THE POWER TO KNOW®

SAS® Enterprise Miner™ 12.3 Reference Help

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Enterprise Miner™ 12.3: Reference Help*. Cary, NC: SAS Institute Inc.

SAS® Enterprise Miner™ 12.3: Reference Help

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

Printing 1, July, 2013

Contents

PART 5 Upgrading and Moving SAS Enterprise Miner Projects 65

PART 6 Analytics 99

PART 7 User Interface 215

PART 8 Data Sources 285

PART 9 Node Reference 317

PART 10 Node Reference: Sample Nodes 323

PART 11 Node Reference: Explore Nodes 411

PART 12 Node Reference: Modify Nodes 611

PART 13 Node Reference: Model Nodes 717

PART 15 Node Reference: Utility Nodes 1071

PART 16 Node Reference: Applications 1193

PART 17 Node Reference: Time Series Nodes 1263

PART 18 Batch Processing 1325

About SAS® Enterprise Miner® 12.3 Reference Help

Audience

This is a hard copy version of the Reference Help that is integrated into the SAS® Enterprise Miner® 12.3 user interface. The SAS® Enterprise Miner® Reference Help is intended for use by novice and experienced licensees of SAS Enterprise Miner 12.3 software.

SAS Enterprise Miner 12.3 Reference Help Requirements

The SAS® Enterprise Miner® 12.3 Reference Help is specifically authored for the SAS® Enterprise Miner® 12.3 workstation or client / server software running on SAS 9.4. Information contained in the SAS® Enterprise Miner® Reference Help may not be accurate for other versions of SAS® Enterprise Miner®.

What’s New in SAS Enterprise Miner 12.3

Overview

SAS Enterprise Miner 12.3 supersedes SAS Enterprise Miner 12.1, released in August 2012. SAS Enterprise Miner 12.3 is the first release of SAS Enterprise Miner on SAS 9.4. SAS Enterprise Miner 12.3 provides improvements and enhancements to many areas of the product including the core user interface, SAS Enterprise Miner Credit Scoring nodes, SAS Enterprise Miner Application nodes, and the SAS Enterprise Miner High Performance Data Mining nodes.

SAS Enterprise Miner Core

- The Enterprise Miner 12.3 client introduces new start-up flexibility. The Enterprise Miner 12.3 client can now be opened to directly load a specific data mining project or diagram, or Enterprise Miner 12.3 users can choose from a Project Navigator tree that contains the most recently worked on projects and process flow diagrams at the top.
- The Enterprise Miner 12.3 Link Analysis node converts relational and transactional data into a data model that can be visualized as a network of effects. The node can detect the linkages between any two variables’ levels for relational data and between two items’ co-occurrence in transactional data. Multiple centrality measures and community information are provided for users to better understand linkage graphs. The Link Analysis node generates cluster scores from raw data that can be used for data reduction and segmentation. The Link Analysis node also uses weighted confidence statistics to provide “next-best-offer” information to customers.
- The Enterprise Miner 12.3 Reporter node improves image and table displays for upgraded PDF and RTF output. The Reporter node report files are smaller in size, but contain improved graphic displays and provide new graphic output font scaling options..
- The Enterprise Miner 12.3 Impute node now supports imputation of special missing values.

SAS Enterprise Miner Applications

- The Enterprise Miner 12.3 Survival node now supports time-varying covariates, as well as user-specified censor and truncation dates.

Part 1

Data Mining Overview

Chapter 1

Data Mining Overview

Data Mining Overview

Data mining is often defined as the process of finding patterns in larger databases. This definition has many implications. One is that the data is largely opportunistic, in the sense that it was not necessarily acquired for the purpose of statistical inference. A significant part of a data mining exercise is spent in an iterative cycle of data investigation: cleansing, aggregation, transformation, and modeling. Another implication is that models are often built on data with large numbers of observations or variables. Statistical methods must be chosen and implemented carefully for scalability. Finally, a data mining model must be actionable. That is, the entire model formula must be executable on separately acquired data and in a separate environment, a process referred to as scoring.

Data miners start with a problem formulation such as, "Rank our current customers by their likelihood to default during the next six months." The problem approach requires historical data that contains actual customer records. The historical customer records will contain some proportion of default occurrences. The task scope requires a minimum of six months of historical data. If customer activity follows seasonal trends, the required historical data must correspond to the season that data miners want to predict. For models that extrapolate into the future, predictor data needs to be selected from a number of months prior to the period in which we measure default occurrences.

For example, data miners might use data acquired from January through June of last year to train models to predict occurrences of default from July through December of last year. The model that was trained and scored with last year's data can use this year's January through June data to predict future default occurrences in the coming months of July through December.

Most often, the available enterprise data has many dimensions, codings, and time periods that will have different relative values in predicting the target. Data miners need to know when the model will be applied, in what operational system, what throughput is required, and what data will be available. The data must be stable. The scales and codings must be consistent throughout model training and scoring. It does no good to build a model that uses data that is too expensive, too volatile, or that is not available at the time of scoring.

Once data miners have the data, they can start building statistical models. Several techniques might be appropriate and the best model for the task might not be known until many model forms have been tested. Often, a business need will determine the model form — many policies might specify a logistic regression model. Data miners must take care to build a model that will generalize by making consistent and reliable predictions in the period of scoring.

SAS Enterprise Miner has been developed to support the entire data mining process. The SAS system provides unparalleled data access to relational and detail data stores. The Base SAS language provides unrivaled power in aggregating and transforming data. Together, SAS/STAT and Enterprise Miner can support virtually any modeling need. Enterprise Miner functions are organized into a process known as SEMMA:

- Sample
- Explore
- Modify
- Model
- Assess

Within each of the SEMMA categories, Enterprise Miner provides a number of tools to promote the process of data mining. For more information about the Enterprise Miner data mining tools in the SEMMA categories, see the reference section on Enterprise Miner Analytics.

Part 2

Starting the SAS Enterprise Miner Client

Chapter 2
Starting the SAS Enterprise Miner Client

Starting SAS Enterprise Miner

Your SAS Enterprise Miner software was installed in one of two configurations: Workstation or Enterprise Client. Either configuration provides complete Enterprise Miner capabilities.

Workstation

The Enterprise Miner Workstation uses SAS services that run on your personal computer or laptop computer. The SAS Deployment Wizard will configure these services during installation.

Enterprise Client

The Enterprise Client connects to remote SAS servers using the SAS Web Infrastructure Platform (WIP). Your organization's SAS system administrator must first install Enterprise Miner on a server, and then ensure that the WIP is running before you can log into a client / server session.

SAS 9 Platform Details

Here are the components of the SAS 9 platform:

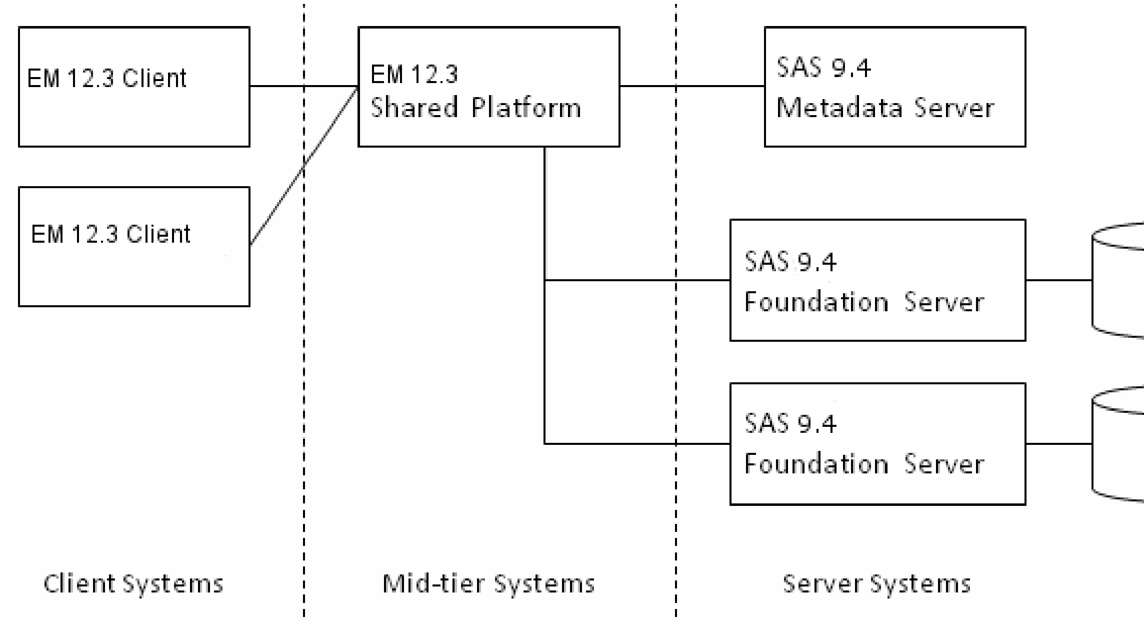
The **SAS Metadata Server** is a central repository for information about SAS such as user names and passwords, user rights, server locations. Administrators use the SAS Management Console to manage the SAS Metadata Server.

The **SAS Foundation Server** is the traditional SAS language and procedures server. The data mining functions all run on the Foundation Server. Client systems present a user interface for entering and generating SAS code, managing processes, and visualizing result sets. SAS Enterprise Miner 12.3 clients are implemented using Java technology.

The **Web Infrastructure Platform (WIP)** functions as a single point of access for multiple clients, and it maintains data mining connections even when you end your Enterprise Miner client session. Multiple users can connect to a single instance of the Enterprise Miner shared components running in the WIP server. The Enterprise Miner shared components are implemented using Java technology.

In the **Workstation** configuration, all processing occurs locally and no connections to SAS Metadata Server and SAS Foundation Server are needed.

In the **Enterprise Client** configuration, the Web Infrastructure Platform (WIP) functions as a bridge between the clients and the servers. Multiple clients can connect to multiple servers. This is the preferred configuration for distributed client/server computing. The WIP must be configured. This task is usually carried out by your SAS system administrators. The clients communicate with the WIP using the HTTP protocol.



Part 3

Getting Started

Chapter 3

Getting Started with SAS Enterprise Miner 12.3

Getting Started with Enterprise Miner 12.3

Overview

This is a very brief overview of using the Enterprise Miner 12.3 software to build and save Enterprise Miner data mining projects, model process flow diagrams, and model packages.

For more detailed information about the Enterprise Miner user interface, see "SAS Enterprise Miner User Interface Help" on page 217.

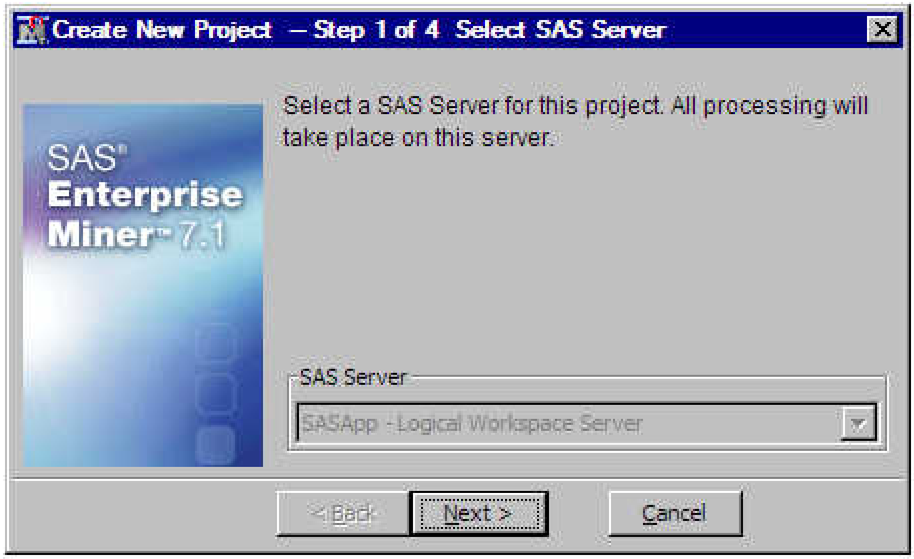
To access the SAS OnlineDoc, open a Web browser and visit **`http://support.sas.com/documentation/onlinedoc/miner/`**. Alternately, you can install the documentation that you want to view from the SAS OnlineDoc CD.

Create a New Project

After you launch your Enterprise Miner 12.3 client session, you can create a new project.

1. From the application main menu, select **File** ⇨ **New** ⇨ **Project**.

   The Create New Project window opens:



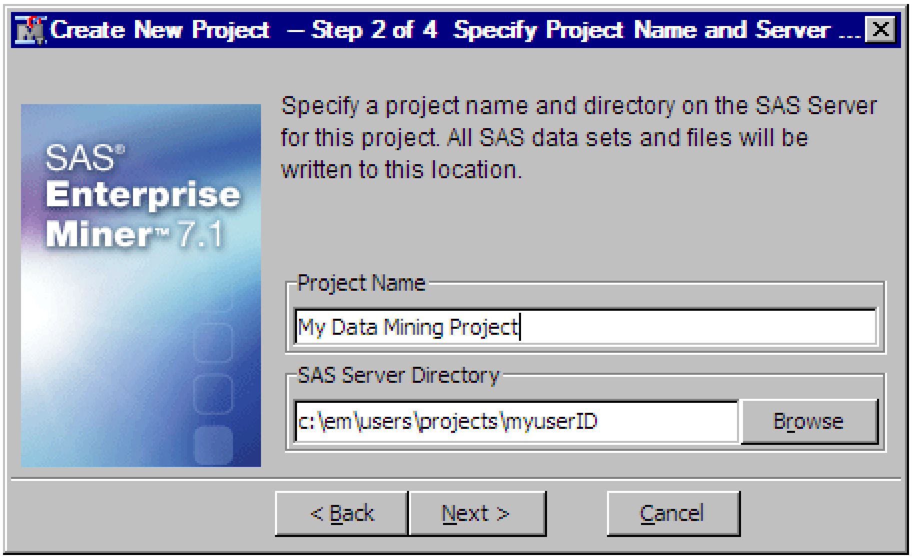
Create New Project – Step 1 of 4 Select SAS Server

Select a SAS Server for this project. All processing will take place on this server.

SAS Server: SASApp - Logical Workspace Server

Select an available server for your project. Click **Next**.

2. Type the name of your project in the **Project Name** field, and specify the server directory that you want to use for your project in the **SAS Server Directory** field.



Create New Project – Step 2 of 4 Specify Project Name and Server ...

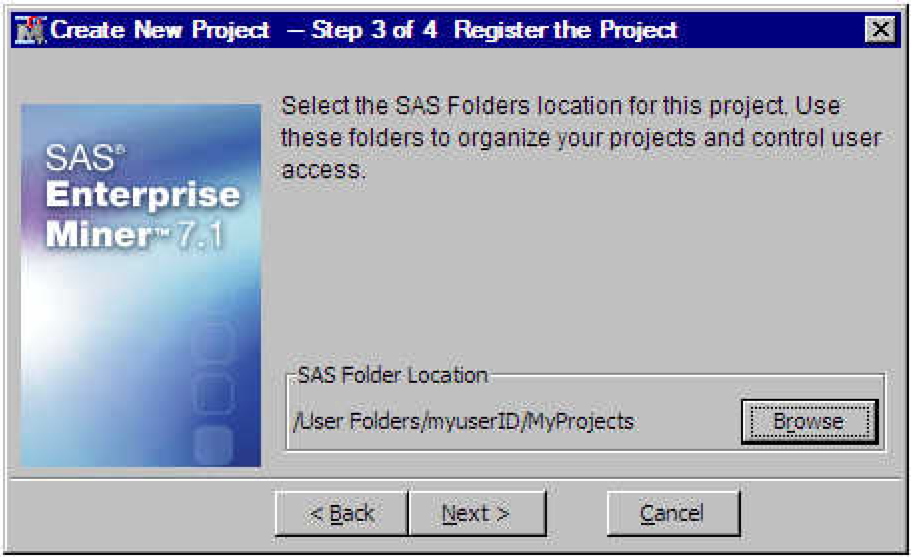
Specify a project name and directory on the SAS Server for this project. All SAS data sets and files will be written to this location.

Project Name: My Data Mining Project

SAS Server Directory: c:\em\users\projects\myuserID

Click **Next**.

3. Use the **Browse** button to select and specify the location for your project's SAS Folders on the server. Your server folders will vary from the folders that are displayed in the example below.



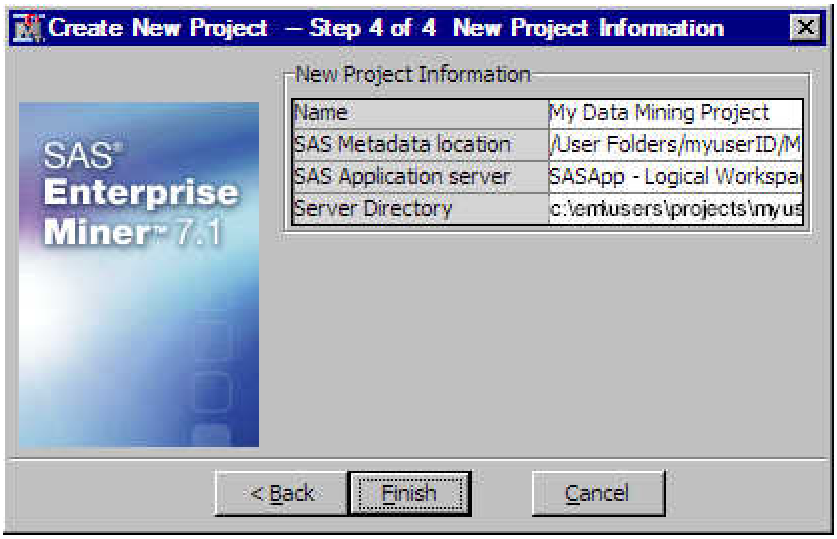
Create New Project – Step 3 of 4 Register the Project

Select the SAS Folders location for this project. Use these folders to organize your projects and control user access.

SAS Folder Location: /User Folders/myuserID/MyProjects

Click **Next**.

4. Review the project information you have just specified in the **New Project Information** table. Use the **Back** button if you need to make changes. When you are ready to create your Enterprise Miner project, click **Finish**.



Create New Project – Step 4 of 4 New Project Information

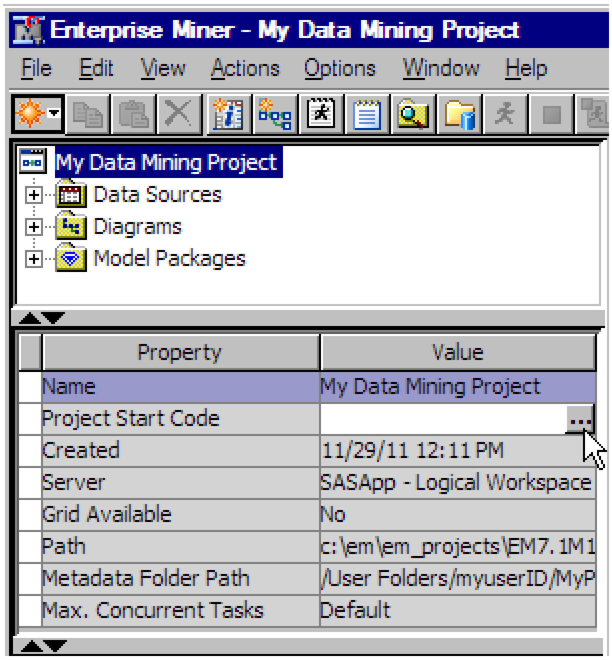
New Project Information

Name	My Data Mining Project
SAS Metadata location	/User Folders/myuserID/M
SAS Application server	SASApp - Logical Workspa
Server Directory	c:\emusers\projects\myus

Project Start Code

You can specify SAS code that you want to run every time you start your Enterprise Miner project. This is called project start code.

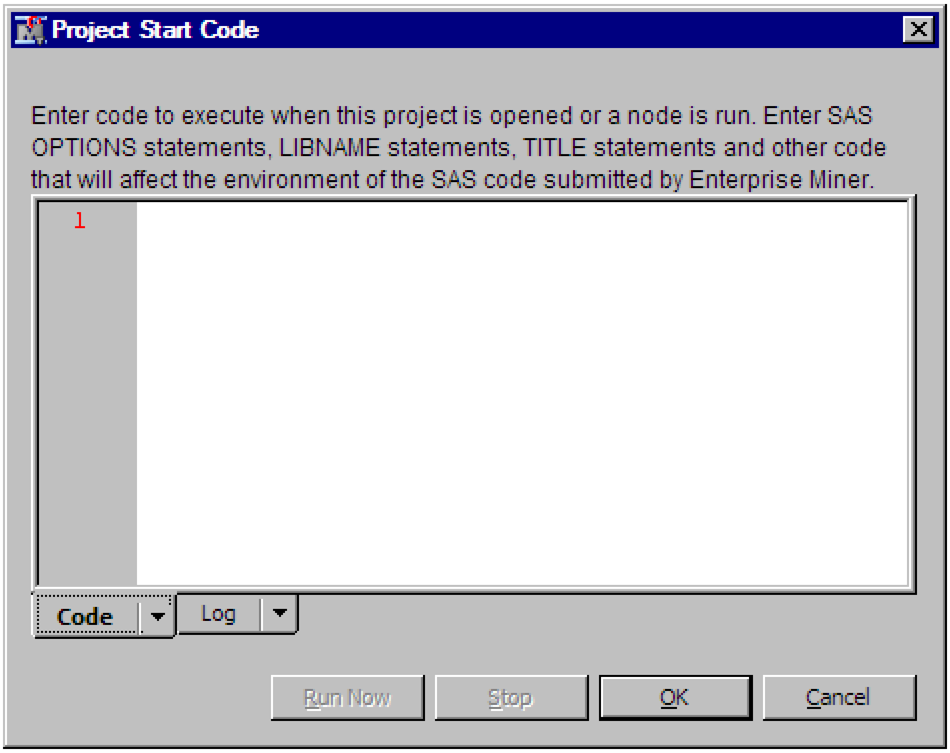
1. Select your newly created project in the Enterprise Miner Project Navigator. The project properties panel displays below.


Enterprise Miner - My Data Mining Project
File Edit View Actions Options Window Help
My Data Mining Project
Data Sources
Diagrams
Model Packages

Property	Value
Name	My Data Mining Project
Project Start Code	...
Created	11/29/11 12:11 PM
Server	SASApp - Logical Workspace
Grid Available	No
Path	c:\em\em_projects\EM7.1M1
Metadata Folder Path	/User Folders/myuserID/MyP
Max. Concurrent Tasks	Default

Click the ellipsis button [...] to the right of the Project Start Code property to open the Project Start Code window.

2. Use the space in this window to enter any SAS code that you want to run each time the project starts, such as LIBNAME statements, grid computing settings, and so on.



Enter code to execute when this project is opened or a node is run. Enter SAS OPTIONS statements, LIBNAME statements, TITLE statements and other code that will affect the environment of the SAS code submitted by Enterprise Miner.

When you have finished entering all of the project start code you want, click **Run Now**. You can select the Log tab to view the results of your start code. When you are satisfied with your start code, click **OK** to close the window.

Note: You can modify your project start code by repeating the previous steps at any time.

Create a New Process Flow Diagram

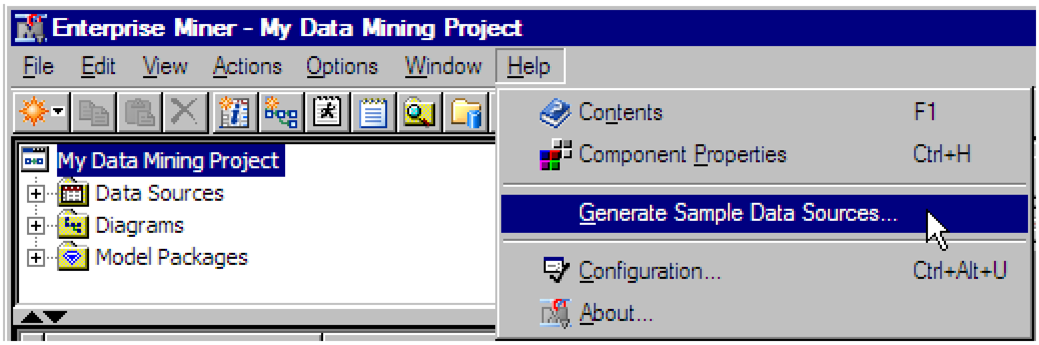
To create a new diagram, select **File** ⇨ **New** ⇨ **Diagram** from the main menu. Type a name for your diagram in the Create New Diagram window and click **OK** to create the diagram. The new diagram opens in the Diagram Workspace.

Create A Data Source

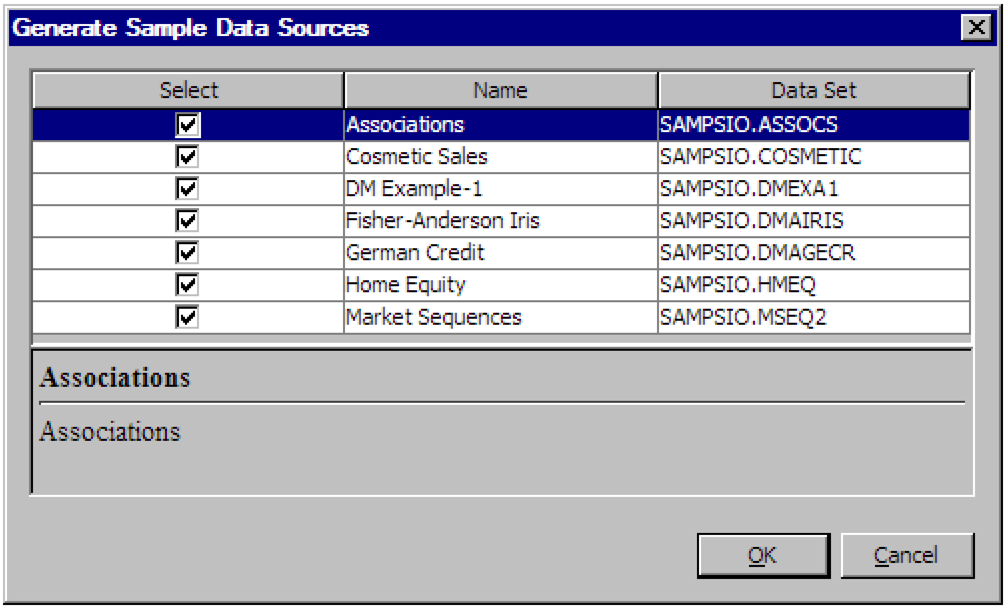
To create a data source, select **File** ⇨ **New** ⇨ **Data Source** from the main menu. The Data Source Wizard opens. You use the Data Source to set up and configure external data tables for use with Enterprise Miner. For more information about using the Data Source Wizard, see "Creating a Data Source Using the Data Source Wizard" on page 293.

SAS Enterprise Miner includes some example data sources that are already set up and ready for you to use. We will add the sample data sources to the **Data Sources** folder in your Project Navigator.

1. From the Enterprise Miner main menu, select **Help** ⇨ **Generate Sample Data Sources**.


Enterprise Miner - My Data Mining Project
File
Edit
View
Actions
Options
Window
Help
My Data Mining Project
Data Sources
Diagrams
Model Packages
Contents F1
Component Properties Ctrl+H
Generate Sample Data Sources...
Configuration... Ctrl+Alt+U
About...

2. The Generate Sample Data Sources window opens. This window contains representative data sets from the SAMPSIO data library that ships with Enterprise Miner. The various data sets are useful for different types of data mining analyses.


Generate Sample Data Sources

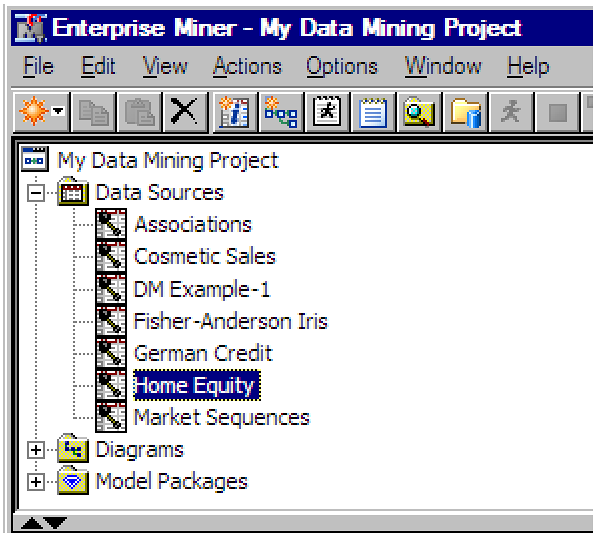
Select	Name	Data Set
☑	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
☑	DM Example-1	SAMPSIO.DMEXA1
☑	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
☑	German Credit	SAMPSIO.DMAGECR
☑	Home Equity	SAMPSIO.HMEQ
☑	Market Sequences	SAMPSIO.MSEQ2

Associations

Associations

By default, all sample data sets are selected for creation. Click **OK** to close the window and to create the sample data sources.

3. Go to your Project Navigator tree and open the **Data Sources** folder. You will see the newly created example Enterprise Miner data sources from the SAMPSIO library inside.



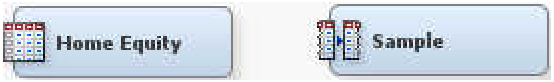
When a data source is selected in the Project Navigator, the data source attributes are displayed in the Properties Panel.

To use a data source in your process flow diagram, select the data source (this example uses the Home Equity data source) in the Project Panel and drag it to the Diagram Workspace.

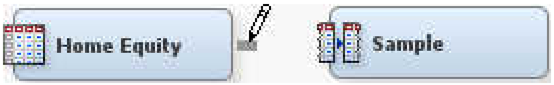
Connect Nodes in Diagram Workspace

Connect data mining tool nodes in the Diagram Workspace to create a logical process flow for your data mining project. When you create a process flow, follow the SEMMA (Sample, Explore, Modify, Model, Assess) data mining methodology. Information flows from node to node in the direction of the connecting arrows. The following example connects the Home Equity data sourdce node to the Sample node.

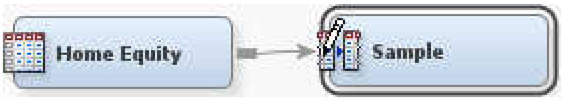
1. Drag the Home Equity data source from the **Data Sources** folder of your Project Navigator onto your Diagram Workspace. Then, from the Sample tab of your Enterprise Miner node tool bar, drag a Sample node onto the workspace.



2. To connect the nodes, move the pointer to the right edge of the Input Data node icon. The pointer icon changes from an arrow to a pencil.



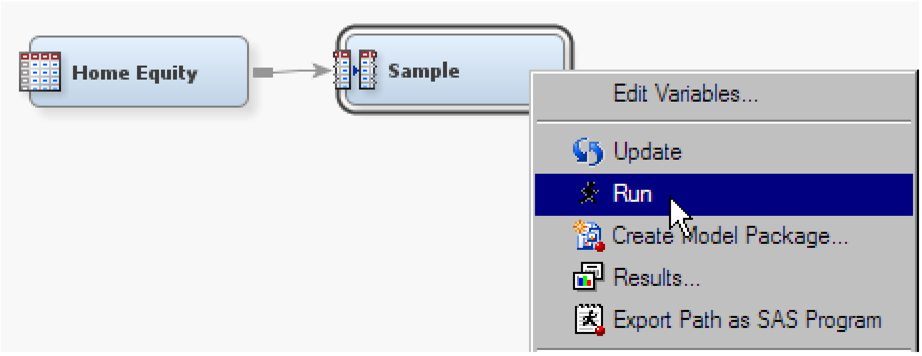
3. Drag the pointer to the left edge of the Sampling node.



Release the pointer, and the nodes are connected.

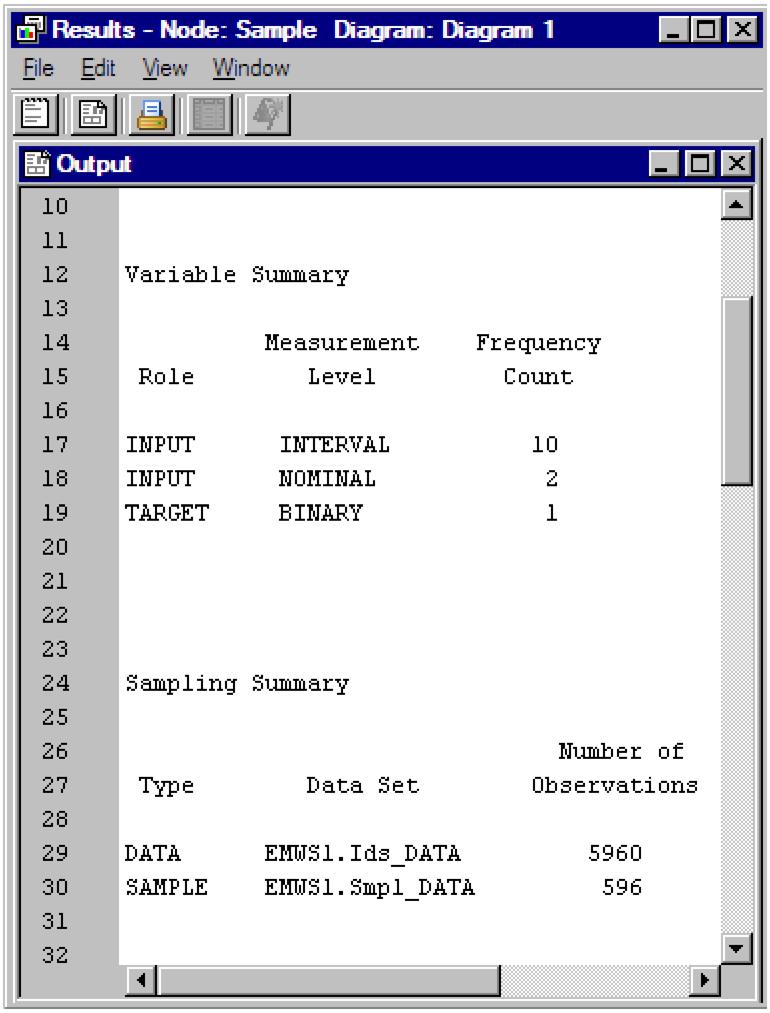
Run the Process Flow Diagram

To generate results from a process flow diagram in your Diagram Workspace, you must run the process flow path to execute the code that is associated with each node. Right-click the node from which you want to run the flow (normally, a terminal node) and select Run from the pop-up menu.



View Results

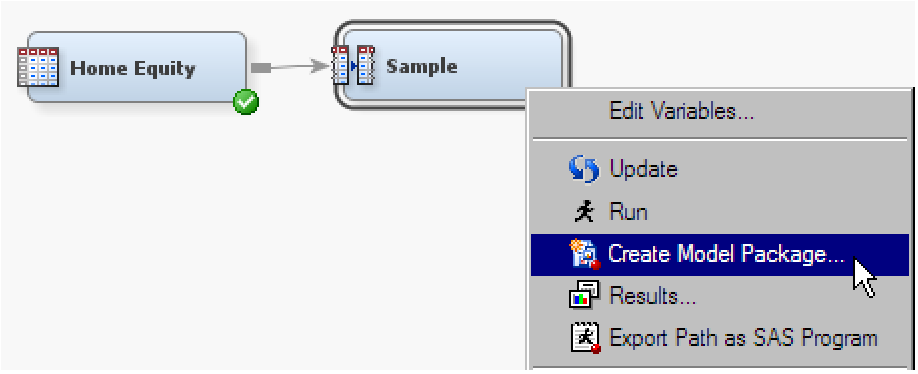
To view the results of a completed run, right-click the node that you ran and select Results from the pop-up menu. You can view the results of the node run from the Results window that opens.



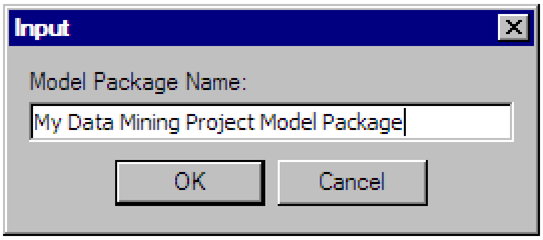
```


Variable Summary

            Measurement    Frequency
 Role          Level         Count

INPUT        INTERVAL          10
INPUT        NOMINAL            2
TARGET       BINARY             1




Sampling Summary

                                     Number of
 Type           Data Set           Observations

DATA         EMWS1.Ids_DATA           5960
SAMPLE       EMWS1.Smpl_DATA           596


```

Create A Model Package

You can export the contents of your model to comparable Enterprise Miner environments using an Enterprise Miner model package. To save your process flow diagram as a model package, right-click the last node in your model, and from the menu, select **Create Model Package**.

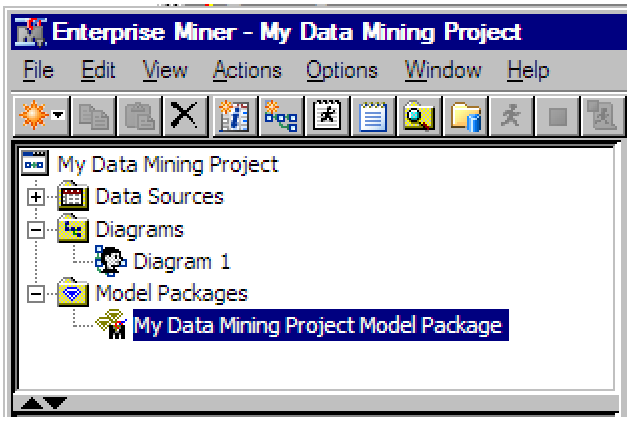


When the Input window opens, specify a name for your model package and click **OK**.



My Data Mining Project Model Package

Your model is now saved inside the Model Packages folder of your Project Navigator.



For more information about model package files, see “About SAS Package Files” on page 1446.

Part 4

Administering SAS Enterprise Miner 12.3

Chapter 4

SAS Enterprise Miner 12.3 Planning and Preparation

SAS Enterprise Miner 12.3 Planning and Preparation

Network Performance

All SAS computations are performed on the server and are unaffected by the client/ server network performance. All project information such as functional settings and intermediate data sets are stored on the SAS server. GUI operations (such as editing process flow diagrams, property sheets, and variables tables) depend on transferring data from the SAS server to the client GUI and are affected by both the server's availability and client/server network performance. SAS Enterprise Miner 12.3 is designed to tolerate network disconnections, and will clean up resources that include SAS sessions. A reliable 1-GB or greater network bandwidth is recommended for baseline client/server performance.

Mapped Network Drives

SAS Workspace and Stored Process servers are not accessible through Windows mapped network drives. You cannot use a Windows mapped drive to create or to access Enterprise Miner projects on a network or access remote libraries if a Windows mapped drive is specified in the path. If an Enterprise Miner Windows client needs access to SAS resources or project libraries on a networked location, the path to the resource should be specified using the network Universal Naming Convention (for example, \ \mypc\myshare\libraries).

Multitasking

The SAS Enterprise Miner 12.3 client starts a SAS session for interacting with the server and submitting SAS code. Each node in a process flow diagram also runs in its own SAS session. Each user can therefore start many SAS sessions on the data mining server.

Note: It is possible for you to start so many SAS sessions on a single CPU desktop system that overall system performance is seriously degraded.

Threading

Some tasks in SAS Enterprise Miner 12.3 (such as data sorting, variable selection, and regression modeling) have been built to distribute their work over multiple CPUs on the same system. Refer to the SAS System documentation for more information about how to configure SAS server options for multithreading.

SAS Servers

The SAS Enterprise Miner 12.3 server is responsible for project storage and computations. To use the SAS Enterprise Miner 12.3 server, you must have installed both a SAS Object Spawner and a SAS Metadata server. Most installation plans will create these services.

Java Requirements

SAS Enterprise Miner 12.3 client requires Java 7 in Windows operating environments and Java 7 in UNIX operating environments. Installed versions of SAS 9.4 include Java 7 current. No further version of Java is required. The SAS Enterprise Miner Client can be delivered to users through Java WebStart. In this case, you will need to download and install the correct version of Java to your local system.

Midtier Platform

System architecture changes in SAS Enterprise Miner 12.3 aim to simplify the single user experience as well as to increase the scalability and conformity to standards of the multi-user experience. The foremost change regards the mid-tier technology: the SAS Analytics Platform server has been deprecated and has replaced by the Web Infrastructure Platform (WIP). The Web Infrastructure Platform is a collection of middle tier services and applications that provides basic integration services. It is delivered as part of the Integration Technologies package. As such, all BI applications, DI applications, and SAS Solutions have access to the Web Infrastructure Platform as part of their standard product bundling. The SAS Analytics Platform service is not used for any SAS 9.4 products or solutions. Existing deployments may disable and remove this service once the new installation is complete.

Workstation and Client / Server Modes

SAS Enterprise Miner 12.3 may be installed and configured in one of two modes.

- In *workstation* mode, SAS Foundation 9.4 and SAS Enterprise Miner 12.3 are deployed on a Microsoft Windows system in a single user configuration. This configuration is indicated for SAS Enterprise Miner Desktop, SAS Enterprise Miner

Classroom, and SAS Enterprise Miner Workstation licenses. This deployment does not require the configuration step of the SAS Deployment Wizard and installing users should not select a configuration plan option. The workstation mode configuration does not require the SAS Metadata Server or the SAS Application Server. Installations based on SAS 9.2 and earlier did require those services; however, they may be removed if they are not required for any other SAS software.

- In *client / server* mode, SAS Foundation 9.4 and SAS Enterprise Miner 12.3 Server can be installed on a local or remote system for multi-user access. The SAS Web Infrastructure Platform is installed as mid-tier server. The SAS Enterprise Miner 12.3 client may be installed on a Microsoft Windows system, or may be started through Java Web Start by connecting your internet browser to the SAS mid-tier.

SAS Enterprise Miner Client

You should be aware of the following considerations when using the SAS Enterprise Miner 12.3 client.

- **Memory** — The maximum Enterprise Miner Java client memory is set to 512 MB. Client systems should have 512 MB of memory for best performance.
- **Property Sheets and Variables Tables** — The data for property sheets and variables table editors is retrieved from the server. If the server is running a resource-intensive diagram, the property sheet or the variables table might be slow to appear.
- **Program Editor** — User-entered code that is submitted through the Program Editor runs in the same SAS session as other GUI operations. Submitting code through the Program Editor blocks other GUI operations, such as concurrently editing a process flow diagram or a property sheet, and interacts in the same way as the traditional SAS Editor session.
- **Graphics and Table Views** — Data for interactive graphics and table views is transferred on demand from the server to the client. The maximum number of rows of data that can be transferred from the server to the client depends on the data. For interactive graphics, you can select a sampling method and size. SAS variable formats are not currently supported in interactive graphics and table views.

Batch Processing

SAS Enterprise Miner 12.3 supports SAS language-based batch processing by using macros. You can use all these macros together in one or in many SAS jobs that conduct complete data mining model building without using the GUI in SAS Enterprise Miner 12.3, or you can use the macros with the GUI in a complementary cycle.

- **%EMDS** — supports the creation of SAS Enterprise Miner 12.3 data source definitions and management of table and column metadata. This capability is useful in automating routine data source registration from batch data preparation jobs. For example, you might have a SAS code job that extracts data from relational tables in a data warehouse and creates a table that is ready for predictive modeling. You can add the %EMDS macro to that job to create the data source definition that is needed for use in the SAS Enterprise Miner 12.3 user interface or in batch computing jobs.
- **%EMTP** — supports the creation of SAS Enterprise Miner 12.3 target profile definitions that are properties of a data source definition. These target profile definitions contain information about the decision matrix, cost values, and prior probabilities, which are used by Enterprise Miner modeling functions.

- **%EM5BATCH** — supports the building, running, and reporting of SAS Enterprise Miner 12.3 process flow diagrams. An exact diagram can be run from either the SAS Enterprise Miner 12.3 user interface or from a batch job. The results can be viewed in the SAS Enterprise Miner 12.3 user interface or included in a reporting SAS program. The SAS Enterprise Miner 12.3 user interface includes a function to build batch code. This is useful when you want to automate creation and execution of a data mining analysis.

Chapter 5
SAS Enterprise Miner 12.3 Client/ Server Sizing

SAS Enterprise Miner 12.3 Client/Server Sizing

Introduction

This document discusses required resources for data mining using SAS Enterprise Miner 12.3. Understanding the resource requirements helps you to size a Windows or UNIX server for your projects. This document assumes that the server is dedicated to SAS Enterprise Miner 12.3 only. If other applications will be running on the server, then the information mentioned needs to be added to the needs of the other applications. Please note that SAS Enterprise Miner 12.3 requires SAS 9.4.

What Size Server Do I Need?

The answer to this question is it depends on the following:

- How many SAS Enterprise Miner 12.3 projects will be running at any given time?
- How much data will these SAS Enterprise Miner 12.3 projects be accessing?
- How many variables will be analyzed?
- What is the expected response time of the users?
- Will there be other applications, such as other SAS sessions or RDBMSs running on the server?

The multithreaded procedures in SAS Enterprise Miner 12.3 will use all the CPUs on the server. The multitasking process flow diagrams will use all the CPUs on the system. SAS Enterprise Miner 12.3 users can also start multiple training runs concurrently. If caution is not exercised, users can start too many concurrent processes. If the concurrent process load is too high, the system spends too much time swapping data, which significantly slows down computation times.

The following factors affect SAS Enterprise Miner 12.3 server sizing:

- Large numbers of input variables require more memory.
- Large numbers of categorical variables, especially those with many levels (such as ZIP codes) require extra memory during modeling. Categorical variables are memory intensive because some data mining functions create separate dummy variables for each unique level of the categorical input.
- Some modeling methods for training data can be memory intensive.
- If you have an RDBMS running on the server (especially if it is supporting an OLTP system), then there might be file system resource conflicts between the RDBMS and SAS Enterprise Miner 12.3. The operating system and RDBMS should be configured to operate on large page sizes. Using small page sizes with SAS Enterprise Miner 12.3 bottlenecks the throughput of mining data and adds significant computing time to model training.

The remaining sections provide recommendations for the amount of memory, the number of processors, and the amount of disk space needed to perform data mining with SAS Enterprise Miner 12.3.

How Much Memory?

Memory for Client and Server Machines

The SAS Enterprise Miner 12.3 product is designed to run in a client/server environment. Here are some recommendations for memory on SAS Enterprise Miner 12.3 client and server machines.

- **Client** — a minimum of 512 MB of physical RAM dedicated to the SAS Enterprise Miner 12.3 product running on the client machine.
- **Server** — a minimum of 512 MB of physical RAM per processor.

Configuring the MEMSIZE Parameter

After sizing and configuring memory for your server, you must allocate memory for the SAS Enterprise Miner 12.3 application. Use the MEMSIZE parameter in your config.sas file to specify the memory allocation for SAS Enterprise Miner 12.3. See the Base SAS documentation for the location of the config.sas file in your SAS 9.4 installation.

The MEMSIZE option is the upper limit to the amount of memory the SAS System can use on a per-session basis. Each session uses memory as required up to the specified MEMSIZE limit. The configuration file is located in the root directory of your SAS Enterprise Miner 12.3 installation.

- **Windows Server** — The MEMSIZE parameter is set to 0 by default, which allows each SAS session to take all available memory on the server. The MEMSIZE parameter recommendation is 512 MB.
- **UNIX Server** — The MEMSIZE parameter is set to a given value by default. The value varies according to different UNIX versions of SAS. Depending on the memory requirements for your project, it might be necessary to increase the value specified for the MEMSIZE option. The MEMSIZE parameter recommendation is the amount of RAM equal to 80% of available host memory. Refer to the host documentation for more information about this option.

Never set the MEMSIZE option to a value that is too high. If multiple SAS sessions are running and every SAS session consumes the maximum memory allotted by MEMSIZE, it is possible to exceed the amount of physical RAM on some servers. On Windows servers, the sum of SAS Enterprise Miner 12.3 MEMSIZE allocations for the peak concurrent users should never exceed the physical amount of RAM, as such conditions result in severe I/O bottlenecking.

Refer to Appendix B: Tuning Parameters for more information related to this topic.

Configuring the CATCACHE Parameter

The CATCACHE parameter is a SAS system parameter that controls whether SAS catalogs are kept open in cached memory for repeated use instead of opening and closing the SAS catalogs each time SAS calls them. Enterprise Miner Java clients are not compatible with SAS system CATCACHE settings that are greater than zero. For best Enterprise Miner performance, the CATCACHE parameter should be set to CATCACHE=0 or CATCACHE=MIN in the config.sas file for the SAS session that launches Enterprise Miner.

How Many Processors

The modeling phase of an SAS Enterprise Miner 12.3 data mining project can require intense memory, CPU, and I/O resources. A good rule of thumb for Windows and UNIX servers is one CPU is capable of supporting between 2 and 5 concurrent SAS Enterprise Miner 12.3 projects.

The underlying procedures of the Association node can be even more disk-space and memory-intensive than the modeling nodes, especially where thousands of items exist in the transaction data set.

The nature of the data that is typically mined influences the processor needs:

- If a data set has a large number of input variables, more resources are consumed and the server will support fewer concurrent sessions.
- If class variables have many unique values, more resources are consumed and the server will support fewer concurrent sessions.
- If a data set uses a small number (such as 10-15) of input variables that have relatively few unique values, less resources are consumed and the server will support more concurrent sessions.

How Much Disk Space?

Allocating Disk Space

Disk space requirements should be analyzed for both the client and server machines. In addition to the amount of disk space needed to install SAS Enterprise Miner 12.3 (see Appendix A), you will need adequate disk space to store the following:

- The actual data to be mined. You must allocate space for the source data, whether it is SAS data sets or RDBMS tables.
- The work data that is generated during SAS Enterprise Miner 12.3 processing on the server. A general rule of thumb is for each project to allow for 4-7 times the amount of the data file being analyzed.
- The SAS Enterprise Miner 12.3 Sample, Data Partition, and Memory-Based Reasoning nodes create work files that are as large as the input data files during

execution. If your data analyses use these nodes regularly, plan your disk space allocation accordingly.

If you use RAID 1 or RAID 5, your SAS Enterprise Miner 12.3 disk space requirements are over and above any space required for the operating system and the RAID file system.

Performance Tips for Disk I/O

To get the best performance with your disks, you need to balance the I/O subsystem on your server. Some general tips for balancing the I/O subsystem are as follows:

- Change the default location for the SASWORK temporary directory to a file system other than the one that the operating system uses for swap files. The default location for SASWORK uses the same file system for both temporary disk space and for the swap file.
- Place all the heavy I/O activity directories (SASWORK, Enterprise Miner Project Libraries) on separate file systems, using independent I/O paths when possible. Some modeling nodes create enormous temporary files in the SASWORK directory. Make sure that your SASWORK directory is at least 1.5 – 2 times as large as your data set.

Computer Resources Needed

The processor and memory resources required to perform a data mining analysis depend on the number of observations and variables in the input data, in the complexity of the data model, and in the training algorithm used. For many modeling algorithms, there is a trade-off between time and memory.

In order for a SAS Enterprise Miner 12.3 modeling node to run, there must be enough free memory to support the operating system, the SAS Supervisor, and the Enterprise Miner software while maintaining a free overhead of about 20 to 30 megabytes.

Computer resources can be calculated using a formula:

Let:

N

be the number of cases (anywhere from 100 to 100,000,000 cases are typical).

V

be the number of input variables.

I

be the number of input terms or units, including dummy variables, interactions, and polynomials.

W

be the number of weights in a neural network.

Q

be the number of output units.

D

be the average depth of a tree.

R

be the number of times the training data are read in logistic regression or neural nets, which depends on the training technique, the termination criteria, the model, and the data. R is typically much larger for neural nets than for logistic regression. In regard to training techniques, R is usually smallest for Newton- Raphson or Levenberg-Marquardt, larger for quasi- Newton, and still larger for conjugate gradients.

S
: be the number of steps in stepwise regression, or 1 if stepwise regression is not used.

- **Association Node** — Memory usage is proportional to [N^M], where N is the number of unique items in the data, and M is the maximum number of items in an association. Both lowering M and/or using higher confidence and/or higher support values can reduce memory and disk usage requirements when performing an associations analysis in SAS Enterprise Miner 12.3.
- **Decision Tree Node** — The minimum additional memory required is 8N bytes. Decision Tree training is considerably faster if the entire data set can be loaded into available RAM. You should have about [8N*(V+1)] bytes of free RAM. If the data will not fit in RAM memory, it is written to a utility file. Memory is also required to hold summary statistics and other tables generated as part of a node's output, but the space required for summary statistics and tables is much smaller than the amount required to train the decision tree.
- **Regression Node** — The required memory depends on the model type and on the selected training technique. During linear regression, the SSCP matrix dominates memory usage. SSCP memory usage is estimated as [$8*I^2$] bytes. During logistic regression, the selected training technique drives memory usage. See the *SAS/OR Software Technical Report: The NLP Procedure* to view memory allocation by training technique. Memory requirements range from approximately [40*I] bytes for the Conjugate Gradient technique to about [$8*I^2$] bytes for the Newton-Raphson technique.
- **Neural Network Node** — The memory usage depends on the selected training technique. See the SAS/OR Software Technical Report: The NLP Procedure to view memory allocation by training technique. Memory requirements range from approximately [40*W] bytes needed for the Conjugate Gradient technique, to as many as [$4*W^2$] bytes needed for the Quasi-Newton and Levenberg-Marquardt techniques.

  You can calculate W for a neural network with biases and H hidden units in one layer:

  W=[H*(I+1)]+[Q*(H+1)].

  For both logistic regression and neural nets, the Conjugate Gradient technique, which requires the least memory, must usually read the training data many more times than the Newton-Raphson and Levenberg-Marquardt techniques.

Assuming that the number of training cases is greater than the number of inputs or weights, the time required for training is approximately proportional to the following:

NI^2
: for linear regression

SRNI
: for logistic regression using conjugate gradients

$SRNI^2$
: for logistic regression using quasi-Newton or Newton-Raphson. Note that R is usually considerably less for these techniques than for conjugate gradients

DNI
: for tree-based models

RNW
: for neural nets using conjugate gradients

RNW^2
for neural nets using quasi-Newton or Levenberg- Marquardt. Note that R is usually considerably less for these techniques than for conjugate gradients

Conclusion

This information is a guideline to understanding resource requirements for SAS Enterprise Miner 12.3. There are several key sizing factors. One is the size of the typical input data set used for data mining. Data set size is affected by the number of observations, the number of variables, and the number of unique levels. The chosen modeling algorithm and its underlying complexity also help to quantify the amount of resources needed to run a SAS Enterprise Miner 12.3 data mining project.

If you have any additional questions about hardware or software configuration, please contact your SAS Account Executive. Your SAS Account Executive can arrange a conference call with the SAS Technology Center to address your specific questions.

Appendix A: Tuning Parameters

- **Change the default MEMSIZE and SORTSIZE parameters** — Change the MEMSIZE and SORTSIZE parameters on the client and server installations of SAS Enterprise Miner 12.3. Set the MEMSIZE parameter to 80% of total host RAM and the SORTSIZE parameter to 16 MB. The MEMSIZE and SORTSIZE parameter definitions should be submitted in the config.sas file typically located in the root directory of your SAS Enterprise Miner 12.3 installation.
- **Move the default location of the SASWORK directory** — The default config.sas file locates the SASWORK directory in a directory that exists on all platforms. In its default location, the SASWORK directory is the same directory that the operating system uses to store its swap files. Moving the SASWORK directory to a different disk or file system will result in performance improvements. If possible, avoid using RAID-1 or RAID-5 fault-tolerant file systems, because the RAID algorithms can require additional disk writes.
- **Run the UNIX SAS jobs with the FULLSTIMER option turned on** — When running your SAS Enterprise Miner 12.3 projects on UNIX, you can add the OPTIONS FULLSTIMER statement to the project parameters. When you add the OPTIONS FULLSTIMER statement, SAS gathers statistics on the resources that it needs and uses to run the application. Examining the FULLSTIMER statistics can help you detect I/O bottlenecks and to quantify the memory needed to run the application. After you tune your resources, turn the FULLSTIMER option back off for even greater resource utilization.
- **Give SAS Enterprise Miner 12.3 Projects all the memory they need** — If you run a data mining analysis with the FULLSTIMER option specified, you can determine how much allocated memory you are using. Performance improves if you increase the amount of memory SAS Enterprise Miner 12.3 can access. To increase the allocated memory, modify the MEMSIZE option in the config.sas file. The config.sas file is located in the root of your SAS Enterprise Miner 12.3 installation directory.
- **Monitor CPU for I/O bottleneck**s — Use various UNIX command line tools like vmstat, iostat, ps and sar; as well as GUI based tools like Solstice SYMON and protocol from Sun Microsystems; Performance Manager for Tru64 (Digital UNIX) from Compaq; and Glance from Hewlett-Packard to do performance monitoring on the UNIX server. For Windows servers, you can use a GUI based tool called performance monitor (perfmon) from Intel to gather the same information about

Windows servers. If you are seeing lots of disk activities, you might need to reorganize your file systems on the server to get the best I/O performance.

- **Balance MEMSIZE versus physical memory** — The optimal memory configuration is to ensure that the summed MEMSIZE values of all concurrent SAS Enterprise Miner 12.3 project users at peak times are less than the physical memory configured on the server.

Appendix B: File System Suggestions

- **Configure sequential I/O for large blocks** — Use the SAS BUFSIZE option to specify a large stripe size such as 64K. However, if your typical data mining task uses indexes that have high cardinality to return small result sets, you should leave the BUFSIZE option in its default setting. In this case the default BUFSIZE performs random access I/O at the file system level.
- **Separate input file systems from output file systems when SAS Enterprise Miner 12.3 transforms data with little reduction** — SAS Enterprise Miner 12.3 DATA steps that produce transformations with little reduction of the incoming data, SQL statements that perform similar operations, and APPEND operations that add data to the end of another data file are all disk-intensive tasks. If you can locate SAS Enterprise Miner 12.3 input, output, and work data sets across separate disks and/or disk controllers, you will reduce the potential for I/O bottlenecks.
- **Set up separate file systems for each user** — SAS Enterprise Miner 12.3 creates lots of intermediate files that are stored in the PROJECT library. If you have lots of SAS Enterprise Miner 12.3 users, you might want to set up several file systems and then partition the users between these files systems in as balanced a manner as possible. This should reduce the number of FSYNCs issued by the SAS System.

Appendix C: Nodes that Create Duplicate Copies of Data Files

Most nodes in a SAS Enterprise Miner 12.3 project use a common data file and create only small tables. However, three nodes can make a second copy of the input data file each time they are used. These additional copies are stored in the data library on the server.

The nodes are as follows:

- Data Partition node
- Sampling node (but only the amount sampled)
- Memory Based Reasoning (MBR) node

Chapter 6
SAS Enterprise Miner 12.3 Single User and Multi-User Deployment

SAS Enterprise Miner 12.3 Single User and Multi-User Deployment

Installation, configuration, and administration have been significantly changed in SAS Enterprise Miner 12.3. The most important fact regards the required version of SAS. SAS Enterprise Miner 12.3 is a component of SAS 9.4 and will not function with any other SAS release.

System architecture changes aim to simplify the single user experience as well as to increase the scalability and conformity to standards of the multi-user experience. The foremost change regards the mid-tier technology: the SAS Analytics Platform server has been deprecated. The SAS Analytics Platform service is not used for any SAS 9.4 products or solutions. Existing deployments may disable and remove this service once the new installation is complete.

SAS Enterprise Miner 12.3 may be installed and configured in one of two modes.

- • In *workstation* mode, SAS Foundation 9.4 and SAS Enterprise Miner 12.3 are deployed on a Microsoft Windows system in a single user configuration. This configuration is indicated for SAS Enterprise Miner Desktop, SAS Enterprise Miner Classroom, and SAS Enterprise Miner Workstation licenses. This deployment does not require the configuration step of the SAS Deployment Wizard and installing users should not select a configuration plan option. The workstation mode configuration does not require the SAS Metadata Server or the SAS Application Server. Installations based on SAS 9.2 and earlier did require those services; however, they may be removed if they are not required for any other SAS software.
- • In *client / server* mode, SAS Foundation 9.4 and SAS Enterprise Miner 12.3 Server can be installed on a local or remote system for multi-user access. The SAS Web Infrastructure Platform is installed as mid-tier server. The SAS Enterprise Miner 12.3 client may be installed on a Microsoft Windows system, or may be started through Java Web Start by connecting your internet browser to the SAS mid-tier.

Chapter 7

SAS Enterprise Miner 12.3 Workstation Configuration

SAS Enterprise Miner 12.3 Workstation Configuration

To start the SAS Enterprise Miner Workstation, select **Start** ➔ **All Programs** ➔ **SAS** ➔ **SAS Enterprise Miner Workstation 12.3**. The Workstation is appropriate for a single user installation using a desktop computer.

Chapter 8
SAS Enterprise Miner 12.3 Projects

SAS Enterprise Miner 12.3 Projects

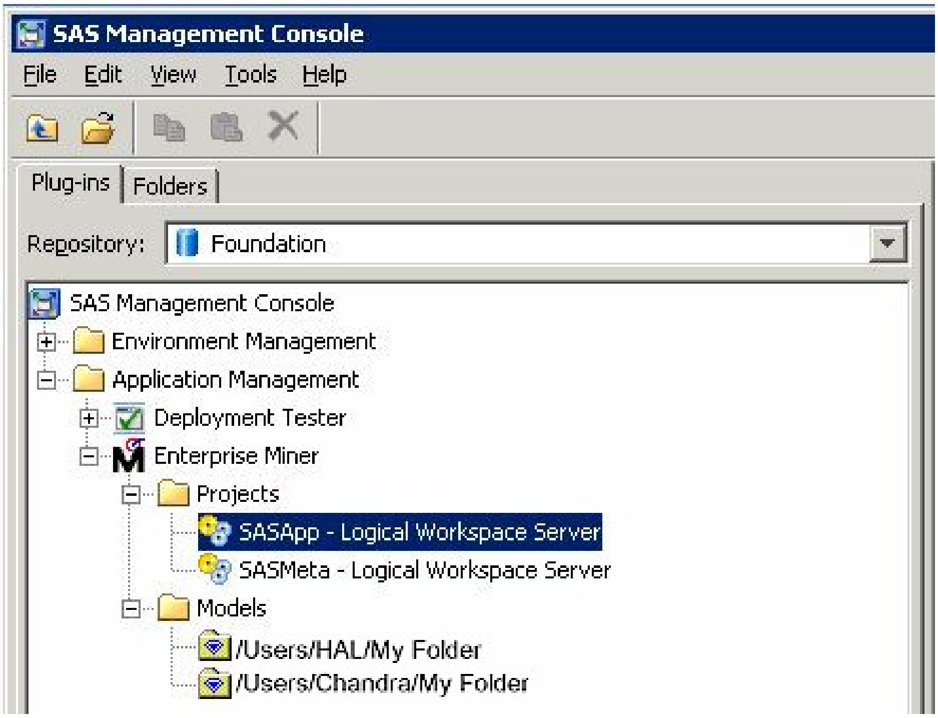
When a SAS Enterprise Miner user creates a new project in the software, a corresponding project object is created in the SAS Metadata server. The project object provides essential project information to the SAS Workspace server. The project objects specify the locations of Enterprise Miner project data sets, catalogs, and files, as well as determining where data mining calculations are executed.

Metadata objects such as file and catalog locations are defined in the SAS Folders of your Enterprise Miner project metadata. You can use the SAS Management Console to view and change permissions and properties of your project metadata objects. The default location for new metadata objects is My Folder.

SAS Enterprise Miner data mining projects that were migrated from a SAS 9.2 installation will be located in the folder named EnterpriseMinerProjects. The EnterpriseMinerProjects folder is located inside the Shared Data folder in the SAS Metadata server.

You can view Enterprise Miner project metadata objects via the Plug-ins tab of the SAS Management Console window. To view all project objects, expand the following folders:

Application Management ⇨ **Enterprise Miner** ⇨ **Projects**.



When you delete a project that you have open in Enterprise Miner, both the SAS metadata entry and the SAS workspace server project files are deleted. When you delete an Enterprise Miner project via the Enterprise Miner View Metadata action, or via the SAS Management Console, only the SAS metadata entry is deleted. In such cases, you will need to manually delete the SAS workspace server project files as well.

When you open a project in SAS Enterprise Miner, you see a list of projects that are registered in SAS metadata. If you want to open an Enterprise Miner project that exists on the workspace server file system, but has not been registered in SAS metadata, do the following:

Use your Enterprise Miner menu **File** ⇨ **New** ⇨ **Project** to launch the Create a New Project wizard. When the wizard prompts you for the new project name and project directory, carefully submit the project directory and project name for the project that exists on the workspace server file system (but is not registered in SAS metadata.) Enterprise Miner will notify you that the project files already exist, and prompt you whether you want to continue with the project creation. If you continue with the project creation, all of the information associated with the existing project is preserved.

Chapter 9
SAS Enterprise Miner 12.3 Models

SAS Enterprise Miner 12.3 Models

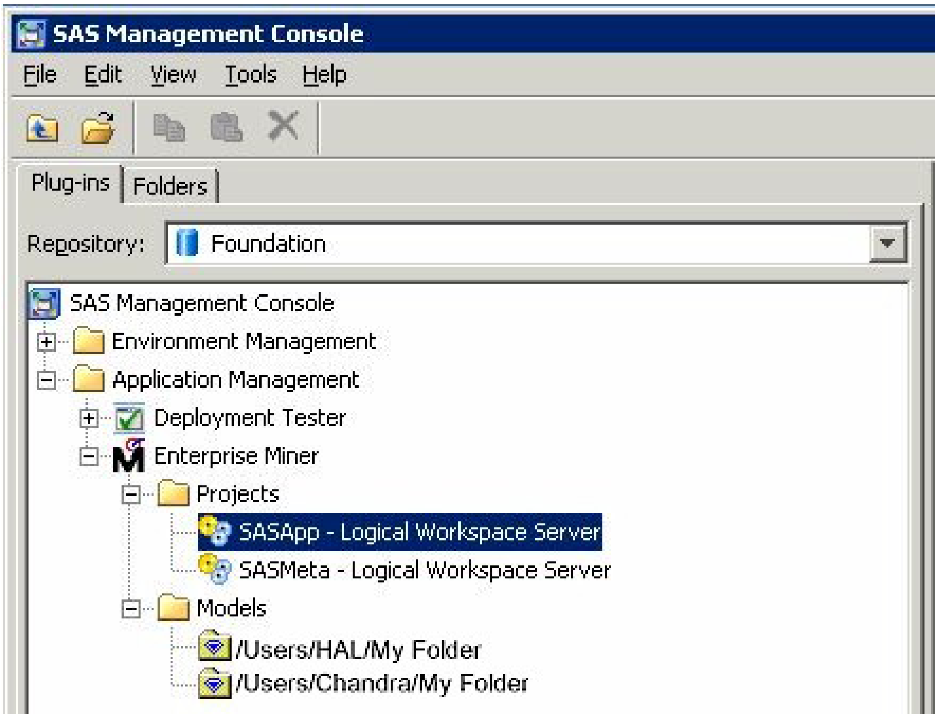
When an Enterprise Miner user registers a model package, a corresponding new model object is created in the SAS Metadata. Model package objects are also known as Mining Result objects. Mining Result Objects contain the following data:

- SAS score code needed for deployment
- lists of input and output variables, types, and formats
- meta information such as model type and model target variables
- name of the algorithm that was used
- name of the data mining analyst
- time of creation

Model package objects are defined in the SAS Folders structures of the SAS Metadata server. You can use the SAS Management Console to change permissions and properties of Model Package objects. The default location for new objects in SAS Management Console is My Folder.

Models that are migrated from a SAS 9.2 installation are stored in a folder called EnterpriseMinerModels. The EnterpriseMinerModels folder is located inside the Shared Data folder in the SAS Metadata server. You can move your model objects to other shared or private locations in the SAS Folders structure.

The Application Management folder in the SAS Management Console contains a list of all metadata objects that are associated with a particular model. To view the list of SAS Enterprise Miner metadata objects, expand the following folders in the Plug-ins tab of SAS Management Console: **Application Management** ⇨ **Enterprise Miner** ⇨ **Models** .



Repository: Foundation

When a model package is registered, a corresponding SAS model package file can be stored in a WebDAV location if desired. SAS provides WebDAV storage through the SAS Content Server product. SAS model package files contain complete data mining results sets, including process flow diagram configuration, model property configuration settings, generated reports, data mining summary tables, SAS log and output listings, graphic plots, score code, and related scoring information.

Chapter 10
SAS Enterprise Miner Plug-in for SAS Management Console

SAS Enterprise Miner Plug-in for SAS Management Console

Overview

The SAS Management Console has a plug-in for Enterprise Miner 12.3. This plug-in enables you to browse and customize some of the metadata objects that Enterprise Miner uses. The Enterprise Miner plug-in provides access to attributes of the Logical Workspace server and a server-specific list of projects. The Enterprise Miner plug-in also provides access to the list of mining result packages that have been registered via Enterprise Miner.

Server Extensions

You can use metadata extensions to Logical Workspace Server definitions to customize the behavior of Enterprise Miner to do the following:

- specify default locations for new projects on a server.
- specify whether users can modify the default location for new projects on a server.
- specify the maximum number of concurrent tasks that are allowed on a particular server. The maximum concurrent tasks setting affects the extent of parallel processing that Enterprise Miner can use in a data mining process flow. You can use the maximum number of concurrent tasks setting to tune your symmetric multiprocessor server environment for optimal performance.
- specify SAS initialization code to be run when a project is opened, or when process flow diagram results or model result packages are generated. SAS initialization code is similar to the project start code feature of Enterprise Miner projects, except that SAS initialization code can initialize all SAS sessions for all projects that are associated with a particular server.

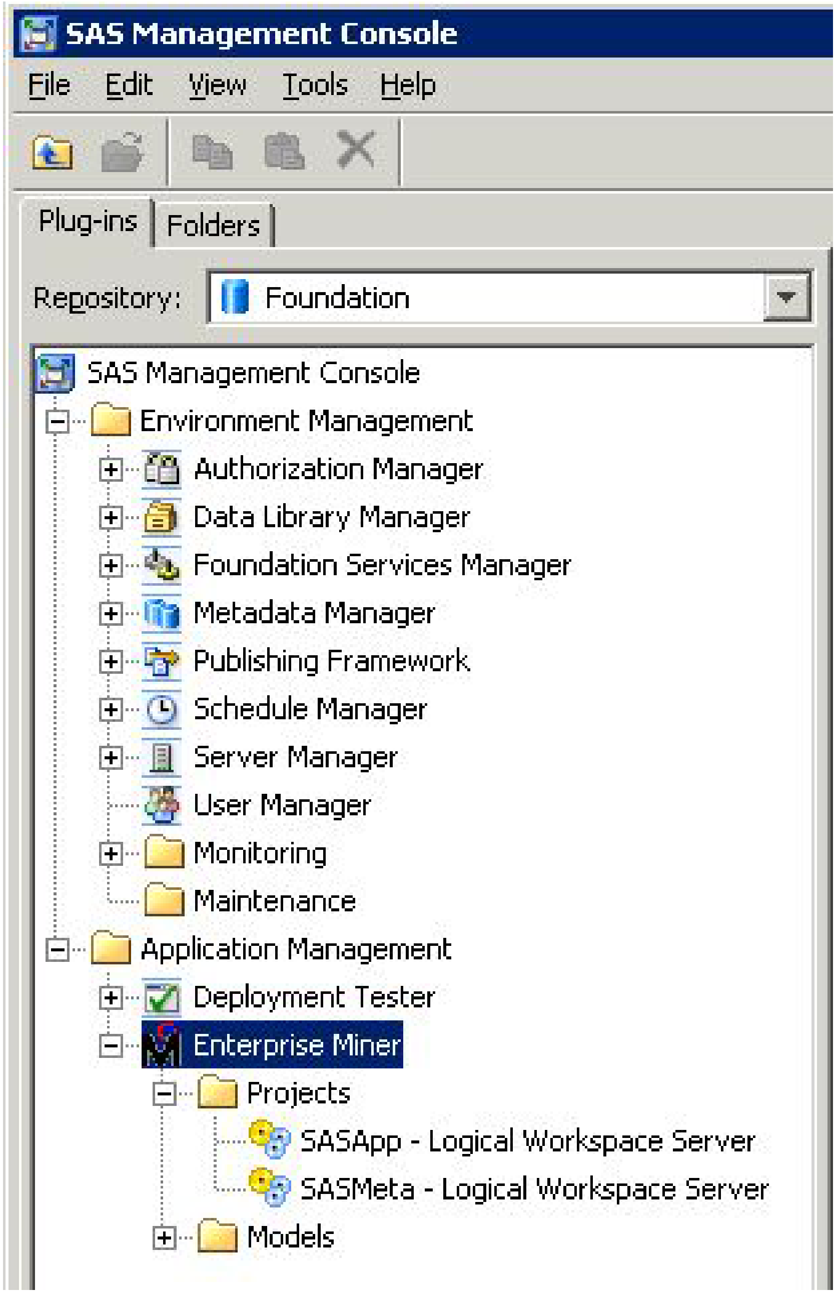
- provide an alternate command to launch any SAS/CONNECT sessions that are called for nodes in a data mining process flow.

Configuring SAS Management Console

Before you can use the Enterprise Miner plug-in for SAS Management Console, you must have started an Enterprise Miner client session at least once. The Enterprise Miner client session initializes the metadata repository with Enterprise Miner metadata. After you successfully start the initial Enterprise Miner client session, you can end the session and start up the SAS Management Console. It is not necessary to define or open any Enterprise Miner projects to initialize the metadata repository.

When you install Enterprise Miner, the Enterprise Miner plug-in is copied to the directory where your SAS Management Console is installed. After you install the Enterprise Miner plug-in, the Enterprise Miner application icon appears in the Application Management section of the left pane of the SAS Management Console window.

In the Application Management section, expand the Enterprise Miner icon to display the Projects and Models folders, as shown in the following display:



Repository: Foundation

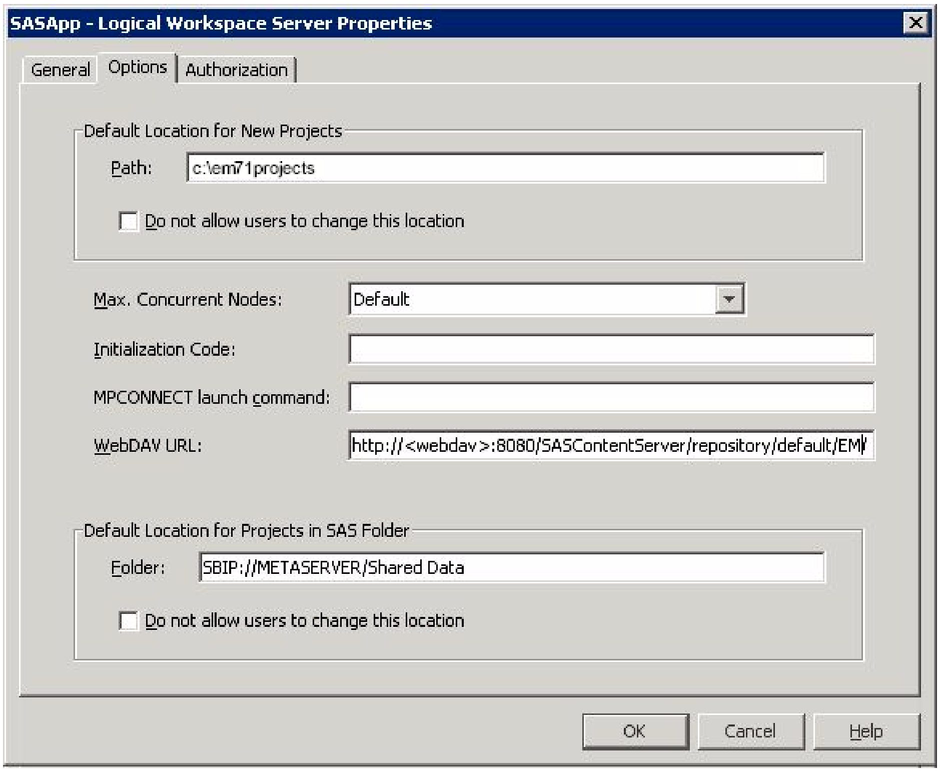
The Projects Folder

The Enterprise Miner Projects folder contains a list of all of the Enterprise Miner servers. When you select the icon for SASApp - Logical Workspace Server, a list of all the projects created by that server displays in the adjacent pane on the right. Expand the Models folder to access the list of model results packages registered from Enterprise Miner to this server.

You can view and customize Logical Workspace Servers, even if you do not have any registered Enterprise Miner projects. Servers will recognize custom project settings when new projects are created on the servers.

Customizing Logical Workspace Servers for Enterprise Miner

To customize the properties for an Enterprise Miner server, perform the following steps:



Default Location for New Projects

Path: c:\em71projects

□ Do not allow users to change this location

Max. Concurrent Nodes: Default

Initialization Code:

MPCONNECT launch command:

WebDAV URL: http://<webdav>:8080/SASContentServer/repository/default/EM/

Default Location for Projects in SAS Folder

Folder: SBIP://METASERVER/Shared Data

□ Do not allow users to change this location

1. Right-click the icon for a server under the Projects folder, and then select Properties from the pop-up menu. The Logical Workspace Server Properties window appears.
2. You can customize the Default Location for New Projects by entering a path. All new Enterprise Miner projects that are created on this server will default to the path that you specify.
3. If you want to prevent users who create projects from changing the default project location, select the Do not allow users to change this location check box.
4. In the Max. Concurrent Nodes list, specify the maximum number of concurrently running branches that you want to allow in project process flow diagrams.
5. In the Initialization Code box, enter the path (on this server) to a project start file, if you want to use one. A project start file is a text file that contains SAS code that you would like to run when a project is opened, or if a process flow diagram is run, or if result reports are generated.
6. In the MPCONNECT launch command box, you can enter an alternate command to use when you launch MPCONNECT sessions. Normally it is safe to leave the MPCONNECT launch command box blank, but there might be cases where you would like to modify some SAS system defaults for sessions that are used when running process flow diagrams. The following default command is used when this box is left blank:

   ```
   !sascmdv -noobjectserver -nosyntaxcheck -noasynchio
   ```

   This command has the same effect as using the SAS command that was used to launch the SAS workspace session at the time the project was opened.

7. In the WebDAV URL box, you have the option to enter the URL of your WebDAV server. If you specify a WebDAV server, the SAS Enterprise Miner model result packages that you save will be copied to the WebDAV server location when the model packages are registered.

Chapter 11
Allocating Libraries for SAS Enterprise Miner 12.3

Allocating Libraries for SAS Enterprise Miner 12.3

Overview: Allocating Libraries

In SAS Enterprise Miner 12.3, there are several places where LIBNAME statements (or other initialization code) can be specified. The library allocations can be specified in these locations:

- SAS autoexec files
- server initialization code
- project start code
- The SAS Management Console Library Manager Plug-In

The general form of the LIBNAME statement is as follows:

```
LIBNAME libref "path";
```

For example, you can specify the following statement:

```
 LIBNAME MYDATA "d:\EMdata\testdata";
```

(Windows path examples are given, but the same principles apply to UNIX systems.)

Allocate Libraries via a SAS Autoexec File

If LIBNAME statements are specified in an autoexec.sas file that resides in the SAS root path, then they execute by default for all SAS processes except those that explicitly specify an autoexec override. You can specify the path to a specific autoexec.sas file by adding the option to the workspace server's SAS launch command or to any sasv9.cfg file:

```
 -autoexec "[full path]"
```

Note: You cannot use a mapped drive specification to indicate the path to an autoexec.sas file.

In most installations, SAS Enterprise Miner uses the configuration file that is located here:

`C:\SAS\EMiner\Lev1\SASApp\sasv9.cfg`.

In this example, EMiner is the installed plan name and might vary from site to site. If you do not designate a plan name, then the default path will be as follows:

```
C:\SAS\Config\Lev1\SASApp\sasv9.cfg
```

The sasv9.cfg file in this directory includes the sasv9.cfg file that is located in the SAS root directory:

```
C:\Program Files\SAS\SASFoundation\9.4\sasv9.cfg
```

The sasv9.cfg file in the SAS root directory points to the last configuration file located in the **`nls\en`** subdirectory:

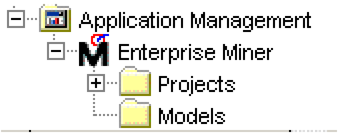
`C:\Program Files\SAS\SASFoundation\9.4\nls\en\sasv9.cfg`.

Allocate Libraries via Server Initialization Code

You can use server initialization code to automatically allocate SAS libraries when users start a SAS Enterprise Miner project. Server initialization code is a server-based text file comprised of SAS statements that execute when a SAS Enterprise Miner project starts up. For example, you can specify LIBNAME statements to allocate one or more libraries when a SAS Enterprise Miner project starts. You can use the SAS Enterprise Miner plug-in to SAS Management Console to customize the properties of SAS Enterprise Miner logical workspace servers, including specifying a path for the logical workspace server to load optional server initialization code that you write.

When you install SAS Enterprise Miner, the SAS Enterprise Miner plug-in for SAS Management Console is copied to the directory where your SAS Management Console is installed. After you install the SAS Enterprise Miner plug-in, the SAS Enterprise Miner application icon appears in the Application Management section of the left pane of the SAS Management Console window.

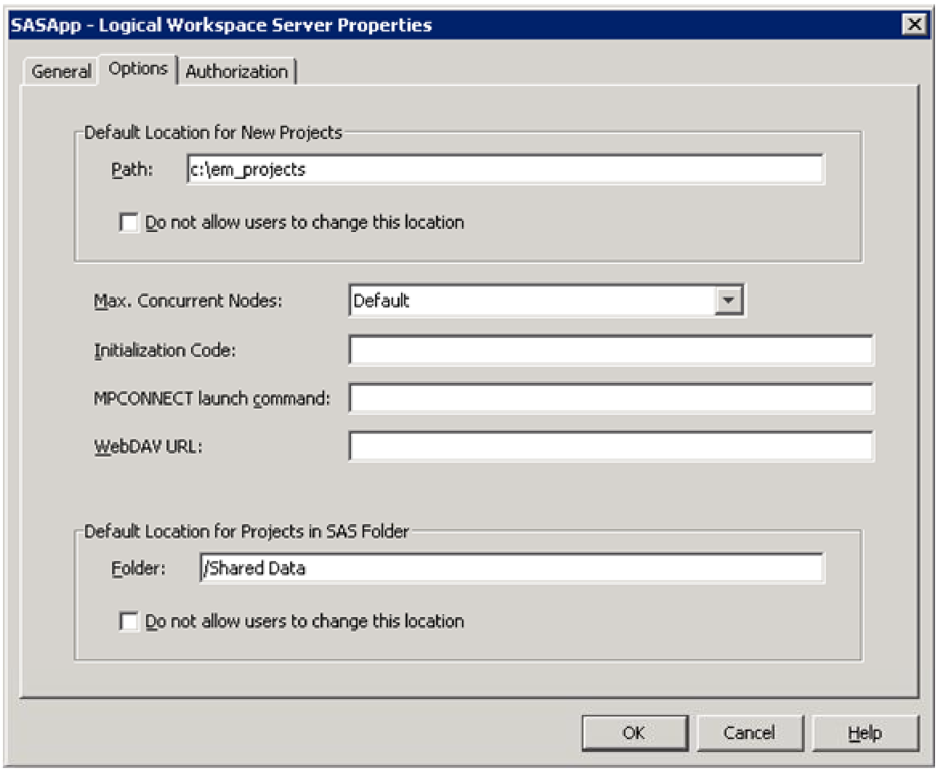
In the Application Management section, expand the SAS Enterprise Miner icon to display the Projects and Models folders, as shown in the following display:



The SAS Enterprise Miner Projects folder contains a list of all of the SAS Enterprise Miner projects that are registered in the SAS Enterprise Miner repository. When you click the Projects folder, a list of all the registered projects displays in the adjacent pane on the right. When you expand the Projects folder, the projects that are displayed are sorted by Logical Workspace Server. Expand the Logical Workspace Server icon in the left pane to subset the list of projects that appear on the right. This enables you to display only the projects that belong to a selected server. You can view and customize Logical Workspace Servers even if you do not have any registered SAS Enterprise Miner projects. Servers that have no registered SAS Enterprise Miner projects will recognize their custom project settings when new projects are created on the servers.

To customize the properties for a SAS Enterprise Miner server, perform the following steps:

1. Right-click the icon for a server under the Projects folder, and then select **Properties** from the pop-up menu. The Logical Workspace Server Properties window appears. Select the **Options** tab.



Default Location for New Projects

Path: c:\em_projects

☐ Do not allow users to change this location

Max. Concurrent Nodes: Default

Initialization Code:

MPCONNECT launch command:

WebDAV URL:

Default Location for Projects in SAS Folder

Folder: /Shared Data

☐ Do not allow users to change this location

2. In the **Path** box of the **Default Location for New Projects** section, enter a default location for new SAS Enterprise Miner projects that are created on this server.

3. If you want to prevent users who create projects from changing the default project location, select the **Do not allow users to change this location** check box .

4. In the **Max. Concurrent Nodes** list, specify the maximum number of concurrently running branches that you want to allow in project process flow diagrams.

5. In the **Initialization Code** box, enter the path (on this server) to a project start file, if you want to use one. A project start file is a text file that contains SAS code that you would like to run when a project is opened, or if a process flow diagram is run, or if result reports are generated.

6. In the MPCONNECT launch command box, you can enter an alternate command to use when you launch MPCONNECT sessions. Normally it is safe to leave the MPCONNECT launch command box blank, but there might be cases where you would like to modify some SAS system defaults for sessions that are used when running process flow diagrams. The following default command is used when this box is left blank:

   ```
   !sascmdv -noobjectserver -nosyntaxcheck -noasynchio
   ```

   This command has the same effect as using the SAS command that was used to launch the SAS workspace session at the time the project was opened.

Allocate Libraries via Project Start Code

You can use SAS Enterprise Miner project start code to issue LIBNAME statements for individual SAS Enterprise Miner projects. To modify the start code for a SAS Enterprise Miner project, open the project in SAS Enterprise Miner, go to the Navigation panel, and select the project name at the top of the navigation tree. With the project highlighted in the Navigation panel, go to the Properties panel, locate the Start Code property, and click the ellipsis [...] button in the Value column. Enter your LIBNAME statement to allocate libraries (or perform other SAS function) in the Start code window and click **OK** to save your new project start code. You can also choose to execute the start code immediately by clicking on the **Run Now** button. A **Log** tab is available so that you can view the SAS log after executing your start code.

Note: You cannot use SAS Enterprise Miner project start code to create an MS Excel library. If you want to use Excel data in SAS Enterprise Miner, you must use the File Import node to do so.

Allocate Libraries via SAS Management Console

SAS Enterprise Miner data libraries that are used frequently can be allocated for use with SAS Enterprise Miner 12.3 using SAS Management Console. First, you must define the library for the SAS Enterprise Miner input data set:

1. Open SAS Management Console.
2. Under the Data Library Manager plug-in, right-click the Libraries folder and select **New Library**.
3. Select **SAS Base Library** and click **Next**.
4. Enter the name of your library and click **Next**. Be sure to check the **Location** and make sure that the selected value is appropriate. The **Location** specifies the metadata folder that contains your library definition.
5. Select an available server from the list on the left and click on the right arrow . This will move the selected server into the adjacent Selected servers pane. Click **Next**.

   Note: If the SASMeta server appears in the list, do not select it as your server.
6. Enter a libref for the library in the LIBREF field. The libref must be 8 characters or less.
7. Click **New** and enter the name of the directory where the library is located. Select the appropriate engine. If the SAS data set is located on the SAS Workspace Server, your engine should be the default Base SAS engine.

   Note: This directory must be accessible to the SAS Workspace Server.
8. Click **Advanced Options**, select the **Library is pre-assigned** check box, and click **OK**.

   Note: If you have SAS tables only, you don't have to pre-assign libraries. The libraries will automatically be available as a **Metadata Repository** from the Create Data Source wizard. If you have database or RDBM libraries (such as Oracle), you must pre-assign RDBMS libraries, and they will be available as a **SAS Table** in the Create Data Source wizard. If you have both SAS and RDBMS tables, you must choose the **Library is pre-assigned** option for both types of tables. Before your libraries will appear in the Create Data Source wizard, you must register them by selecting **Register Tables** from the menu. Now, all of your tables will be available via **SAS Table** in the Create Data Source wizard.

9. Click **Next** and review your entries. Text similar to this should be displayed:

```
Library:
          My Enterprise Miner data
    Libref:
            emdata
    Location:
            /Shared Data
    Assigned to SAS Servers:
            SASApp
    Libref:
            MyData
    Engine:
            BASE
    Path Specification:
            c:\yourdata <specify correct path to data>
    Library is pre-assigned:
         Yes
```

If this looks correct, click **Finish** and then **OK**.

Next, you must grant read permission for the metadata in your new library:

1. In SAS Management Console, click the Data Library Manager icon.
2. Expand the Libraries folder.
3. Right-click the SAS library that you just created and select **Properties** from the pop-up menu.
4. In the Library Properties window, go to the **Authorization** tab and select the **PUBLIC** group.
5. Select the check box in the Grant column for the Read permission row.

ERROR: Data Set LIBREF.TABLENAME Does Not Exist

In SAS Enterprise Miner 12.3, nodes that follow a SAS Code node or custom node in a process flow diagram can produce an error that indicates that the data set that a node attempted to reference does not exist. You might get this error, even when you are able to successfully create the data source, and can explore the data set in your session. In SAS Enterprise Miner 12.3, each node in a process flow diagram spawns a new SAS session. The currently executing node does not have access to libraries that were allocated via the SAS Program Editor or a predecessor SAS Code node. In order for SAS libraries to be available to all tools and nodes in SAS Enterprise Miner 12.3, the LIBNAME statements must be specified in a location that is executed for each spawned session, such as in the project start code, the server initialization code, or SAS Management Console.

Chapter 12

SAS Enterprise Miner 12.3 PMML Support

SAS Enterprise Miner 12.3 PMML Support

PMML Overview

PMML (Predictive Modeling Markup Language) is an XML-based standard for representing data mining results. The PMML standard was developed by the Data Mining Group (DMG), an independent group of software vendors. PMML is designed to enable the sharing and deployment of data mining results between vendor applications and across data management systems. The Data Mining Group released PMML Version 2.1 in March 2003. More information about DMG can be found at

`http://www.dmg.org`

SAS Enterprise Miner produces PMML files for selected data mining functions. SAS Enterprise Miner 12.3 PMML is updated to full compatibility with DMG Version 4.1. The SAS Enterprise Miner PMML files use significant extensions in order to support the data types, transformations, and model definitions that SAS requires.

Enabling PMML Generation

By default, SAS Enterprise Miner nodes do not generate PMML. To enable the creation of PMML code, you must submit the following macro statement in the SAS Enterprise Miner Project Start Code:

```
%let EM_PMML=Y;
```

SAS Enterprise Miner 12.3 Nodes that Support PMML

The table below outlines the SAS Enterprise Miner nodes that generate PMML output.

Table 12.1 SAS Enterprise Miner Nodes that Generate PMML Output

SAS Enterprise Miner Node Name	Scoring Functions	Generates SAS Code?	Generates PMML?
Associations	Probability, Classification, and Decision Assignment	Yes	Yes
Cluster	Segment and Distance Assignment	Yes	Yes
Regression	Probability, Classification, and Decision Assignment	Yes	Yes
Dmine Regression	Probability, Classification, and Decision Assignment	Yes	Yes
Decision Tree	Probability, Classification, Decision Assignment, and Leaf Assignment	Yes	Yes
Neural Network	Probability, Classification, and Decision Assignment	Yes	Yes
AutoNeural	Probability, Classification, and Decision Assignment	Yes	Yes

PMML Document Structures

Header

The PMML header contains timestamp and other system and application information.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<PMML version="4.1">
<Header copyright="Copyright(c) 2002 SAS Institute Inc.,
        Cary, NC, USA. All Rights Reserved.">
<Application name="SAS(r)" version="9.4"/>
<Timestamp>2011-01-10 22:48:30</Timestamp>
```

Data Dictionary

The Data Dictionary contains information about variables in the Data Source. The scoring engine ignores attributes that beginning with the prefix "x-SAS".

```
<DataDictionary numberOfFields="20">
<DataField name="x" optype="continuous" dataType="double"/>
<DataField name="time" optype="continuous" dataType="timeSeconds"
        x-SASinformat="TIME."/>
<DataField name="y" optype="continuous" dataType="double"/>
<DataField name="c" optype="categorical" dataType="string"
        x-SASformat="$CHAR."/>
<DataField name="n" optype="categorical" dataType="double"
        x-SASformat="BEST12."/>
```

Transformation Dictionary

The Transformation Dictionary contains information about any SAS user-defined formats in use. The logic behind the format is captured as a function definition in generic PMML terms. Note that the PMML is intended for non-proprietary representation.

```
<TransformationDictionary>
  <DefineFunction name="COLOR" optype="ordinal" x-SASuserformat="COLOR">
    <ParameterField name="AnyInput" optype="continuous"/>
    <Discretize field="AnyInput" defaultValue="Purple">
      <DiscretizeBin binValue="Red">
        <Interval closure="closedClosed" leftMargin="1" rightMargin="1"/>
      </DiscretizeBin>
      <DiscretizeBin binValue="Green">
        <Interval closure="closedClosed" leftMargin="2" rightMargin="2"/>
```

Model

The rest of the PMML document contains various aspects of the data mining model as described in the following sections.

```
<RegressionModel functionName="regression"
     targetFieldName="y" normalizationMethod="none">
```

Mining Schema

Mining Schema contains information about variable roles in the model, missing value handling, and so on.

```
<MiningSchema>
  <MiningField name="x" usageType="active" optype="continuous"/>
  <MiningField name="color" usageType="active" optype="categorical"/>
  <MiningField name="y" usageType="predicted" optype="continuous"/>
  <MiningField name="customer" usageType="group" optype="categorical"
            outliers="asIs" missingValueTreatment="asIs"/>
```

Output Section

Output section contains information about computed fields such as errors and residuals.

```
<Output>
  <OutputField name="P_y" displayName="Predicted: y" optype="continuous"
        dataType="double" field="y" feature="predictedValue"/>
  <OutputField name="E_y" displayName="Error Function: y" optype="continuous"
        dataType="double" field="y" feature="predictedValue"/>
```

Local Transformation

```
<DerivedField name="n+" optype="categorical">
  <!-- SAS Format: 12.0 -->
  <Apply function="SAS-EM-String-Normalize">
    <Constant>12</Constant>
    <FieldRef field="n"/>
  </Apply>

<DerivedField name="_1_0" optype="continuous">
  <MapValues outputColumn="To">
   <FieldColumnPair field="n+" column="From"/>
   <InlineTable>
     <row>
       <From>1</From>
       <To>1</To>
     </row>
     <row>
        <From>2</From>
        <To>0</To>
        </row>
```

Target Section

Target Section enlists possible target or predicted values and any priors.

```
<Targets>
  <Target field="species" optype="categorical">
    <TargetValue value="VIRGINICA" rawDataValue="virginica "
        priorProbability="0.33333333333333"/>
    <TargetValue value="VERSICOLOR" rawDataValue="versicolor"
        priorProbability="0.33333333333333"/>
    <TargetValue value="SETOSA" rawDataValue="setosa    "
        priorProbability="0.33333333333333"/>
</Targets>
```

Model Details

Model Details on the actual model follow. For example, here is a snippet from a regression model showing intercepts and coefficients.

```
<RegressionTable intercept="6.4">
  <NumericPredictor name="_0_0" coefficient="-1.7033934213238E-15"/>
  <NumericPredictor name="_0_1" coefficient="7.1465186199382E-16"/>
  <NumericPredictor name="_1_0" coefficient="1.6"/>
  <NumericPredictor name="_1_1" coefficient="-0.4"/>
```

Here is a code snippet from a tree model showing a split:

```
<Node id="1" score="7.54230769230769" recordCount="130">
  <True/>
  <Node id="2" score="5.33333333333333" recordCount="60">
    <CompoundPredicate booleanOperator="surrogate">
      <CompoundPredicate booleanOperator="or">
        <SimplePredicate field="color+-" operator="equal" value="1"/>
        <SimplePredicate field="color+-" operator="equal" value="2"/>
      </CompoundPredicate>
  <False/>
  </CompoundPredicate>
  <Node id="5" score="3.225" recordCount="10">
```

SAS Extensions to PMML

SAS Informats

- Match data between database systems and SAS systems
- Datetime, Date, Time, Char, Numeric

SAS Formats

- W, Best, Hour, Downame, Monname, Year, and so on.

User-Defined Formats

- Map Raw Char to Formatted Char
- Map Raw Numeric to Formatted Char

SAS Functions

- Upcase(), Trim(), Left(), Cosine()

Modeling

- Add lift to associations
- General class variable dummy encoding
- Regression interaction terms
- Regression link functions
- Neural Radial Basis Function - width and altitude
- Tree multi-way splits
- Cluster missing value handling
- Output field definitions: Probability, Classification, Residual, Error

SAS Formats and Informats in PMML

SAS system informats are used to map data values into the SAS environment. This mapping is encoded into PMML as data type parameters in the data dictionary.

SAS has proprietary representation for date and time values. Dates are represented as the number of days since January 1, 1960. Time is represented as the number of seconds since midnight. DATETIME stamps are represented as the number of seconds since midnight of January 1, 1960.

When date and time data is brought into SAS from a DBMS, SAS assigns a date/time informat to the specific variables and converts them for SAS internal representation. Subsequent modeling on such data imported into SAS works with the converted internal values. To enable accurate scoring of these models on new data, a similar conversion must be applied.

Three new data types have been introduced in PMML to represent date and time values:

SAS Informat	PMML Data Type
date	dateDaysSince[1960]

SAS Informat	PMML Data Type
time	timeSeconds
datetime	dateTimeSecondsSince[1960]

A PMML scoring engine must convert the incoming data from the DBMS value to the SAS internal values based on the actual date or time values.

The SAS system offers many formats. SAS Enterprise Miner models class variables based on formatted values. If there is no specific format, numeric categorical variables are formatted using the default BEST12 format. Character categorical variables are formatted with the CHAR format.

Formats in PMML are handled as a function application with names of the type SAS-FORMAT-formatname. Formats in PMML are usually seen in the local transformation section of the PMML document. The scoring engine implements the SAS-FORMAT-functions.

The following SAS system formats are currently supported in PMML:

SAS System Formats	PMML Function Names
BEST	SAS-FORMAT-BESTw
$CHAR	SAS-FORMAT-$CHARw
DATE	SAS-FORMAT-DATEp
DATETIME	SAS-FORMAT-DATETIMEp
DOWNAME	SAS-FORMAT-DOWNAMEw
HOUR	SAS-FORMAT-HOURp
MONTH	SAS-FORMAT-MONTHp
MONNAME	SAS-FORMAT-MONNAMEw
MONYY	SAS-FORMAT-MONYYw
QTR	SAS-FORMAT-QTR
W.D	SAS-FORMAT-WD
WEEKDAY	SAS-FORMAT-WEEKDAY
YEAR	SAS-FORMAT-YEARp
YYQ	SAS-FORMAT-YYQp

Example of SAS Format to PMML Conversion

For example, the SAS variable purchase_date here is categorized into day of the week values using DOWNAME format.

```
<DerivedField name="purchase_date+" optype="categorical">
  <!-- SAS Format: DOWNAME9.0 -->
  <Apply function="SAS-EM-String-Normalize">
    <Constant>9</Constant>
    <Apply function="SAS-FORMAT-DOWNAMEw">
      <Constant>9</Constant>
       <FieldRef field="purchase_date"/>
    </Apply>
   </Apply>
  </DerivedField>
```

SAS User-Defined Formats in PMML

SAS User-Defined formats are also handled as a function application in PMML. However, the function is explicitly defined in the Transformation Dictionary section, using SAS PROC FORMAT to define the format. Most value, character, and numeric range mapping formats are supported. User-Defined formats involving picture formats, string ranges, multi-labels and fuzz features are not supported.

Text Normalization in PMML

SAS Enterprise Miner normalizes all category values, which includes trimming leading blanks, truncating a value to a fixed length and then making the string all upper case letters. Models are built with normalized values and reference normalized values during scoring. For example, a value like "Yes" is normalized into “YES”. A PMML scoring engine needs to replicate the normalization process while scoring SAS models. PMML accomplishes string normalization using the SAS-EM-String-Normalize function with a constant parameter for truncation length. An example is shown below.

```
<Apply function="SAS-EM-String-Normalize">
  <Constant>32</Constant>
  <Apply function="SAS-FORMAT-$CHARw">
    <Constant>32</Constant>
     <FieldRef field="URLhit"/>
  </Apply>
```

Decision Processing in PMML

Decision processing is currently not supported in PMML. Prior probabilities for target levels (priors) specified by a user during training are currently ignored as well and no adjustments are made to PMML scores.

Scoring a PMML Document

The process of scoring a PMML document is specific to the database vendor and the scoring engine provider. Several vendors, including IBM and Teradata, have developed modules to score PMML models produced by SAS Enterprise Miner. The following example is taken from IBM DB2 Intelligent Miner Scoring, Version 8.1.

The scoring process is represented in two steps:

1. Register a PMML model in DB2. This is done with an SQL statement such as:

```
insert into IDMMX.ClassifModels values
  ( 'myRegModel102', IDMMX.DM_impClasFile('\\myMachine\myPMMLlib\dmreg.xml'));
```

2. Score a DB2 table with a previously registered model. This is done with statements such as

```
WITH classifView(species, sepallen, sepalwid, petallen, petalwid, classifResult)
AS
(
 SELECT B."species", B."sepallen", B."sepalwid", B."petallen", B."petalwid",
    IDMMX.DM_applyClasModel( C.MODEL, IDMMX.DM_impApplData
    ( rec2xml( 1.0, 'COLATTVAL', '',
    B."sepallen", B."sepalwid", B."petallen", B."petalwid" ) ) )

 FROM "dmairis" B, IDMMX.ClassifModels C
 WHERE C.MODELNAME='myRegModel102'
)
 SELECT sepallen, sepalwid, petallen, petalwid, species,
   IDMMX.DM_getPredClass( classifResult ) as pred_species
 FROM classifView ;
```

Please refer to the scoring engine vendor documentation for more specific details.

PROC PSCORE and PMML

Overview

PMML is an XML markup language that was developed to exchange predictive and statistical models between modeling systems and scoring platforms. Users can import the majority of standard-compliant PMML models and score them within a SAS environment via the SAS PSCORE procedure.

PROC PSCORE Functionality

The SAS PSCORE procedure generates SAS Data Step score code that is functionally equivalent to the PMML model. The generated score code can be executed on all SAS supported platforms to score the data sets. You can submit the score code in SAS Enterprise Miner via the Program Editor, SAS Enterprise Miner Project code, or within a SAS Enterprise Miner Process Flow Diagram, via the SAS Code node. However, the SAS Enterprise Miner UI environment is not necessary to run the score code.

Supported Versions

SAS PROC PSCORE currently supports the use of PMML 4.1. Earlier versions of PMML are not supported for use with PROC PSCORE.

Supported PMML Models

SAS PROC PSCORE supports the following types of PMML models:

- Regression
- Trees
- Neural Networks
- Clustering models

- Scorecard
- Vector Machine
- Naïve Bayes
- Baseline models

The following models are supported on an experimental basis:

- Time Series
- General Regression

Requirements for PROC PSCORE

In order to use PROC PSCORE, you must have SAS 9 or later, a well formed PMML modeling file, and write access to the output directory for the data step score file. A SAS Enterprise Miner license is not necessary to run PROC PSCORE.

PROC PSCORE Usage

```
PROC PSCORE PMML FILE = "<full-pathname-of-PMML-file>"
     DS FILE = "<full-pathname-of-output-DS-file>"
```

PROC PSCORE Example

/*Run the PSCORE procedure on a generated PMML file*/

```
PROC PSCORE PMML FILE = "C:\temp\heart_pmml1.xml"
     DS FILE = "C:\temp\ds_heart_score.sas";
  run;
```

Part 5

Upgrading and Moving SAS Enterprise Miner Projects

Chapter 13
Upgrading and Moving SAS Enterprise Miner Projects

Opening SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 Projects in Later Versions of SAS Enterprise Miner

Do I Need to Perform a Project Conversion?

If you are using SAS Enterprise Miner 12.3, you can open all projects that were created in SAS Enterprise Miner 5.3 or later without any required conversion, as long as the projects are stored on the same type of operating system.

If you are using SAS Enterprise Miner 6.2 or SAS Enterprise Miner 6.1, you can open all projects that were created in SAS Enterprise Miner 5.3 or SAS Enterprise Miner 6.1 without any required conversion, as long as the projects are stored on the same type of operating system.

If you want to open a project from SAS Enterprise Miner 4.3, SAS Enterprise Miner 4.2, or SAS Enterprise Miner 4.1 using SAS Enterprise Miner 5.3 or later, follow the steps below to convert the SAS Enterprise Miner 4.3 project to SAS Enterprise Miner 5.3 format. SAS Enterprise Miner 5.3 and all subsequent releases can open the converted SAS Enterprise Miner 5.3 project without further modification.

If you need to migrate a SAS Enterprise Miner project across different operating systems, you must use the migration macro that is documented here: `http://`

support.sas.com/demosdownloads/setupcat.jsp?cat=Enterprise +Miner.

Overview of Converting SAS Enterprise Miner 4.x Projects to SAS Enterprise Miner 5.3 Project

SAS Enterprise Miner 5.3 and all subsequent versions contain the %EMCONVERT macro. You can use %EMCONVERT to migrate data mining projects that were developed and run in SAS Enterprise Miner 4.x,for use with SAS Enterprise 5.3 or later. When you convert a SAS Enterprise Miner 4.x project into SAS Enterprise Miner 5.3 project format, the process flow diagrams, logs, output, score code, and assessment results from the SAS Enterprise Miner 4.x project are saved in the SAS Enterprise Miner 5.3 project format. All versions of SAS Enterprise Miner 5.3 or later open SAS Enterprise Miner 4.x projects that were converted to SAS Enterprise Miner 5.3 project format, as well as projects that were created and saved in SAS Enterprise Miner 5.3.

You use the %EM_CONVERT macro to convert a SAS Enterprise Miner 4.x project into SAS Enterprise Miner 12.3 format. The source code for the %EM_CONVERT macro is distributed with SAS Enterprise Miner 12.3. The %EM_CONVERT macro converts a SAS Enterprise Miner 4.x project into a project folder structure that SAS Enterprise Miner 12.3 can recognize and open.

You can configure the project conversion macro to create:

- a complete project folder structure that SAS Enterprise Miner 12.3 can open.
- a SAS batch code file (project creation batch code) that creates project folder structures used by SAS Enterprise Miner 12.3 when the SAS batch code file is run on a SAS Enterprise Miner 12.3 server. You can use SAS batch code files to archive SAS Enterprise Miner projects.
- both a complete project folder structure that SAS Enterprise Miner 12.3 can open and a SAS Enterprise Miner project creation batch code file. SAS batch code is platform-independent, so the project creation batch code will run on any SAS Enterprise Miner 12.3 server, regardless of its operating environment.

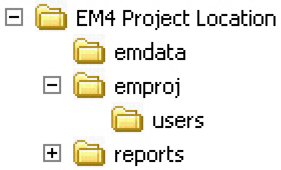
There is a difference between SAS Enterprise Miner 12.3 projects that are created directly by the conversion macro and SAS Enterprise Miner 12.3 projects that are created by running the project creation batch code file that the conversion macro saves.

If you use the conversion macro to create a complete project (instead of the batch code file), you can display the transferred SAS Enterprise Miner 4.x results and compare them to your new SAS Enterprise Miner 5.3 or later results. When you run the project creation batch code that the conversion macro saves, the logs, output, score code, and assessment results from the SAS Enterprise Miner 4.x project are not transferred to the converted SAS Enterprise Miner 5.3 or later project.

Comparing SAS Enterprise Miner 4.x and SAS Enterprise Miner 12.3 Project Directory Structures

SAS Enterprise Miner 4.x Project Directory Structures

SAS Enterprise Miner 4.x stores project directory structures for both local and client/server projects on the SAS Enterprise Miner client machine. For each project, SAS Enterprise Miner 4.x creates these three subdirectories: emdata, emproj, and reports:

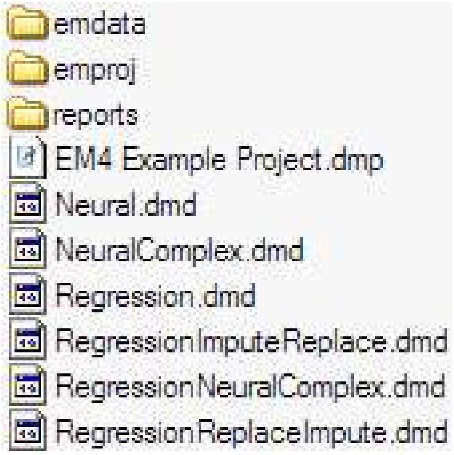


EM4 Project Location
emdata
emproj
users
reports

You can use SAS Enterprise Miner 4.x to find your existing SAS Enterprise Miner 4.x project directory and browse its subdirectories. Open the SAS Enterprise Miner 4.x project, select the Diagrams tab of the Project Navigator, right-click on a named project icon, and then select Explore. A Windows Explorer window appears and displays the project folder and its location in your file system. On Windows systems, the default storage location for SAS Enterprise Miner 4.x projects is as follows:

`C:\Documents and Settings\YourEMUserID\My Documents\My SAS Files\9.1\EM Projects`

The EM4 Project Location directory contains the DMP file for the project, as well as the various DMD files that represent each process flow diagram in the project. In addition to the DMP and DMD files, the project directory also contains the emdata, emproj, and reports subdirectories.



emdata
emproj
reports
EM4 Example Project.dmp
Neural.dmd
NeuralComplex.dmd
Regression.dmd
RegressionImputeReplace.dmd
RegressionNeuralComplex.dmd
RegressionReplaceImpute.dmd

The emdata directory contains Emdata librefs and files that are created when you run process flow diagrams in your project. If you are working on a SAS Enterprise Miner 4.x client/server project, the project's emdata files are written to the data directory on your SAS Enterprise Miner 4.x server. If you are working on a SAS Enterprise Miner 4.x project that can run locally (without connecting to the server), then SAS Enterprise Miner uses the emdata directory on your SAS Enterprise Miner 4.x client.

The emproj directory contains Emproj librefs and project files that contain information for each process flow diagram, its nodes, target profiler settings, and various registries. The users subdirectory in the emproj directory contains files that represent the users, if any, who are currently sharing the project.

The reports subdirectory contains HTML report files, if your SAS Enterprise Miner 4.x project contains a process flow diagram that has a Reporter node. Reports that were generated by SAS Enterprise Miner 4.x Reporter nodes are not transferred to converted SAS Enterprise Miner 12.3 projects. However, the Reporter node is reproduced in converted SAS Enterprise Miner 12.3 projects. You can run the Reporter node in your converted project to generate and save SAS Enterprise Miner 12.3 Reporter node results.

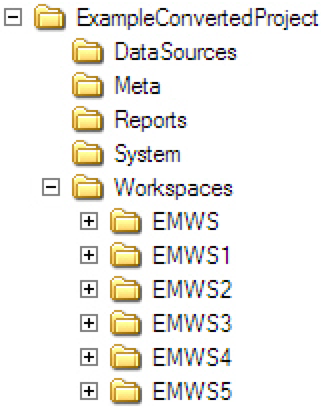
SAS Enterprise Miner 4.x projects are self-contained. You can use file utilities such as Windows Explorer to archive or copy SAS Enterprise Miner 4.x projects. The SAS Enterprise Miner project conversion process does not modify source SAS Enterprise

Miner 4.x project files. If the project that you want to convert is located on a server that has limited user access and rights (such as a production server), you can copy the SAS Enterprise Miner 4.x project files to a work area and then perform the project conversion on the copied project.

If you want to move the project (via ftp, for example) to a different operating system, first put the converted project into a transport file, such as a ZIP or TAR file that will be recognized by the destination operating system.

SAS Enterprise Miner 12.3 Project Directory Structures

Unlike SAS Enterprise Miner 4.x, SAS Enterprise Miner 12.3 stores all project information on the SAS Enterprise Miner server. For each project, SAS Enterprise Miner 12.3 creates a project directory that has five standard subdirectories: DataSources, Meta, Reports, System, and Workspaces. Each folder is briefly discussed as follows so you can understand the SAS Enterprise Miner 12.3 project folder structure and what the folders should contain in a newly converted project:



```
⊟ ExampleConvertedProject
     DataSources
     Meta
     Reports
     System
  ⊟ Workspaces
     ⊞ EMWS
     ⊞ EMWS1
     ⊞ EMWS2
     ⊞ EMWS3
     ⊞ EMWS4
     ⊞ EMWS5
```

The DataSources and Meta folders are SAS Enterprise Miner 12.3 structures that did not exist in SAS Enterprise Miner 4.x. Converted SAS Enterprise Miner 12.3 projects are created with empty DataSources folders. The Meta folder contains assorted project and diagram metadata that is generated by SAS Enterprise Miner 12.3 projects. Converted SAS Enterprise Miner 12.3 projects are created with empty Meta folders.

The Reports folder is also empty in a newly converted SAS Enterprise Miner 12.3 project. The HTML report files that the SAS Enterprise Miner 4.x Reporter node creates are not transferred to converted SAS Enterprise Miner 12.3 projects. If your converted SAS Enterprise Miner 5.3 or later project contains process flow diagrams that use the Reporter node, you can run the process flow diagrams in SAS Enterprise Miner 5.3 or later, and then save the Reporter node output as an LST or PDF file. The newly saved LST and PDF files are stored in the Reports folder of your SAS Enterprise Miner 12.3 project.

The System folder holds SAS Enterprise Miner project system files, such as project start-up and exit scripts. Start-up and exit scripts from SAS Enterprise Miner 4.x projects are currently not transferred by the SAS Enterprise Miner 12.3 project conversion macro. Converted SAS Enterprise Miner 12.3 projects are created with empty System folders. If your converted SAS Enterprise Miner 5.3 or later project requires start and exit code, you can enter the code when you open your converted SAS Enterprise Miner 4.x project in SAS Enterprise Miner 12.3.

The Workspaces directory contains a separate Enterprise Miner workspace subdirectory for each process flow diagram in the project. Each process flow diagram subdirectory is named with an EMWS prefix. The EMWS directories contain the files that are associated with the process flow diagram. The EMWS directory for a process flow diagram also contains a subdirectory for each node in the process flow diagram. The node subdirectories contain files that document each node's property configuration settings, as well as node output files such as results and SAS logs. In newly converted SAS Enterprise Miner 12.3 projects, the Workspaces directory and all of its EMWS subdirectories should contain data.

All SAS Enterprise Miner 12.3 projects must be located on the SAS Enterprise Miner 12.3 server to function, but the location where you store the project on the server is flexible. If you desire, you can distribute projects across multiple directories on your SAS Enterprise Miner 12.3 server. If you want to store your converted SAS Enterprise Miner 4.x projects in the same location as your existing SAS Enterprise Miner 12.3 projects, it is easy to find the location of your existing projects on the server. To find the storage location of an existing SAS Enterprise Miner 12.3 project, open the project and select the project icon in the SAS Enterprise Miner Project Navigator. The Path property in the project's Properties panel displays the path to the project on your SAS Enterprise Miner 12.3 server.

Project Conversion Requirements

The following requirements must be met for the project conversion macro to complete successfully:

- The conversion macro is packaged with SAS Enterprise Miner 12.3 and SAS 9.4. The conversion code must run in a Windows SAS session that has SAS Enterprise Miner 12.3 installed, or a release of SAS Enterprise Miner 4.3 that was shipped with SAS Enterprise Miner 12.3. The Windows environment requirement applies only to the SAS session / Enterprise Miner client workstation, not to the SAS Enterprise Miner server. The conversion macro will not be affected if your SAS 9.4 client connects to a server that uses an operating environment other than Windows (for example, a UNIX server).
- The conversion macro must run on a SAS 9.4 session with a SAS Enterprise Miner 4.3 client that can access both the source project to be converted and the destination where you want to create the converted project. In order to function in SAS Enterprise Miner 12.3, converted projects must be stored on the SAS Enterprise Miner 12.3 server. However, you can create your converted project (or project creation batch code file) in another location, and then transfer the converted project (or project creation batch code file) to a location on the SAS Enterprise Miner 12.3 server. If your SAS Enterprise Miner 12.3 server uses an operating environment other than Windows, you need to use a transport file (such as a TAR file) that the SAS Enterprise Miner 12.3 server operating environment can recognize.

CAUTION:

SAS Enterprise Miner 4.3 should not be open when you submit your conversion code to your SAS 9.4 session. If SAS Enterprise Miner 4.3 is open when you run the project conversion macro, certain files that are required for the conversion might be locked and can prevent the macro from successfully completing.

Project Conversion Macro Syntax

The source code for the project conversion macro is called EM_CONVERT.SOURCE. It is located in the EMCONVRT catalog of the SASHELP library in your SAS installation. The source code for the project conversion macro can be found in SASHELP.EMCONVRT.EM_CONVERT.SOURCE.

The %EM_CONVERT macro has the following general form:

%EM_CONVERT(
EM4PROJECT=*SAS-Enterprise-Miner-4.x-project-path*,

PROJECTPATH=*SAS-Enterprise-Miner-12.3-project-path,*

PROJECTNAME=*SAS-Enterprise-Miner-12.3-project-name,*

CREATE=Y|N *(generate-Enterprise-Miner-12.3-project-folders-with-results),*

FILEREF=*fileref-for-Enterprise-Miner-12.3-project-creation-batch-code***)**;

where

EM4PROJECT=*SAS-Enterprise-Miner-4.x-project-path*
Specifies the path to the directory that contains the SAS Enterprise Miner 4.x project that you want to convert. This parameter is required.

PROJECTPATH=*SAS-Enterprise-Miner-12.3-project-path,*
Specifies the path to the SAS Enterprise Miner 12.3 project that you want to create. If the destination project path that you specify does not exist, it is created by the macro. If you do not specify a value for your SAS Enterprise Miner 12.3 project path, the converted project is created in the WORK folder of your SAS Enterprise Miner 4.3 client session.

PROJECTNAME=*SAS-Enterprise-Miner-12.3-project-name,*
Specifies the name of the SAS Enterprise Miner 12.3 project that you want to create. If you do not specify a value for your SAS Enterprise Miner 12.3 project name, the converted project is created with the name EmProject.

CREATE=Y|N *(generate-Enterprise-Miner-12.3-project-folders-with-results),*
Specifies whether to create the SAS Enterprise Miner 12.3 project folder structure. The SAS Enterprise Miner 12.3 project folder structure contains process flow diagram metadata and results files. You must create the project folder structure if you want to transfer existing SAS Enterprise Miner 4.x process flow diagram results to your converted SAS Enterprise Miner 12.3 project. The default setting is CREATE=Y. If you want the project conversion macro to save the project creation batch code file instead of creating the project folders, you can specify CREATE=N, and then use the FILEREF= argument to specify the path and filename for the project creation batch code file that you want to save. The CREATE= parameter is optional.

FILEREF=*fileref-for-Enterprise-Miner-12.3-project-creation-batch-code)*
Indicates the path and filename to use when you want to save SAS Enterprise Miner 12.3 project creation batch code. Project creation batch code is SAS code that generates the converted SAS Enterprise Miner 12.3 project when it is run on a SAS Enterprise Miner 12.3 server. SAS batch code is platform-independent and works in all SAS Enterprise Miner 12.3 server operating environments. If you do not submit a FILEREF= specification, project creation batch code is not saved. If your project conversion code specifies CREATE=N and has no FILEREF= statement, the macro does not produce output. The SAS Enterprise Miner 12.3 project creation batch code does not transfer existing SAS Enterprise Miner 4.x project results to your converted SAS Enterprise Miner 12.3 project. The FILEREF= parameter is optional.

Steps to Convert a SAS Enterprise Miner 4.x Project to a SAS Enterprise Miner 12.3 Project

1. You must use a Windows workstation with a SAS Enterprise Miner 4.x client that can access the SAS Enterprise Miner 12.3 server and SAS 9.4 software or later to run your project conversion macro.

2. Determine whether you want the project conversion macro to create a complete SAS Enterprise Miner 12.3 project folder structure, a SAS Enterprise Miner 12.3 project creation batch code file, or both:

   - To create only the SAS Enterprise Miner 12.3 complete project folder structure, specify CREATE=Y in the conversion macro code, and do not submit a FILEREF= specification statement.
   - To create only the SAS Enterprise Miner 12.3 project creation batch code file, specify CREATE=N in the conversion macro code, and use a FILEREF= statement to specify the path and filename that you want to use to save the SAS Enterprise Miner 5.3 project creation batch code file.
   - To create both a complete SAS Enterprise Miner 12.3 project folder structure and a SAS Enterprise Miner 5.3 project creation batch code file, specify CREATE=Y, and use the FILEREF= option to specify the path and filename that you want to use to save the SAS Enterprise Miner 12.3 project creation batch code file.
   - If you want to save only the SAS Enterprise Miner 12.3 project creation batch code file, and if the converted project will reside on a SAS Enterprise Miner 12.3 server that uses an operating environment other than Windows, be sure to use the server's operating system syntax when you specify macro paths and fileref arguments.

3. Determine the following information about the SAS Enterprise Miner 4.x project (or project copy) that you plan to convert:

   - the location of the SAS Enterprise Miner 4.x project that you want to convert
   - the path to the location where you want to save the converted SAS Enterprise Miner 12.3 project (or project creation batch file)
   - the name that you want to use for the converted project
   - the filename and path to the location of the project creation batch code file, if you intend to create one.

4. Open a SAS 9.4 session on the Windows workstation that can access both your SAS Enterprise Miner 12.3 server and the SAS Enterprise Miner 4.x project that you want to convert. You must be sure to close SAS Enterprise Miner 4.x before you run your project conversion macro.

5. Enter your project conversion macro code into the SAS Program Editor, and substitute the settings that you determined in steps 3 and 4 for your conversion project. You will need to initialize the SAS catalog that contains the conversion macro code, and if you intend to save a project creation batch code file, you should initialize your fileref variable. The conversion macro examples in this document include useful code that demonstrates how to perform those tasks. Submit your project conversion code to the Program Editor of a SAS 9.4 session. You cannot submit project conversion code using the Program Editor of SAS Enterprise Miner 12.3.

6. After the conversion macro runs, examine your destination project and file paths to verify that the converted SAS Enterprise Miner 12.3 project folder structures or project construction batch code file was created. If you used the conversion macro to create your SAS Enterprise Miner 12.3 project folder structure in a location that is not on your SAS Enterprise Miner 12.3 server, copy or capture the converted project structure in a transport file, and move it to the desired project location on your SAS Enterprise Miner 12.3 server.

7. If you created a SAS Enterprise Miner 12.3 project creation batch code file to build the converted project on a remote SAS Enterprise Miner 12.3 server, or on a SAS Enterprise Miner 12.3 server that uses a different operating system, copy or transport the project creation batch code file to its new location. Open a SAS 9.4 session on the SAS Enterprise Miner 12.3 server, and submit the project creation batch code. The batch code creates the converted project folder structures in the directory where the batch code file was run. Remember, project creation batch code does not transfer earlier project results.

8. Open SAS Enterprise Miner 12.3, and choose New Project. A Create New Project window appears. In the Create New Project window, specify the following information in the order given:

   - Use the Host list to select your SAS Enterprise Miner 12.3 server. You should choose the SAS Enterprise Miner 12.3 server where your converted SAS Enterprise Miner 12.3 project is now located.

   - Use the Path field to specify the server path to the folder that contains the project folder for your converted SAS Enterprise Miner 12.3 project.

   - Use the Name field to specify the name of your converted SAS Enterprise Miner 12.3 project. The name of your SAS Enterprise Miner 12.3 project must match the name of your converted SAS Enterprise Miner 12.3 project folder. Your converted SAS Enterprise Miner 12.3 project folder should be a subdirectory of the folder that you specified in the Path field.

   - If your converted SAS Enterprise Miner 12.3 project requires any start or exit code, use the appropriate tabs in the Create New Project window to enter the information.

   - Select OK to create and open your converted SAS Enterprise Miner 12.3 project.

9. When your converted project opens in SAS Enterprise Miner 12.3, go to the Project Navigator and open the Diagrams folder. You should see an entry for each process flow diagram in the project. After you run a process flow diagram in your converted project, you can open the Results window and compare the results of the converted SAS Enterprise Miner 12.3 process flow diagram and nodes to the results of the SAS Enterprise Miner 4.x process flow diagram and nodes.

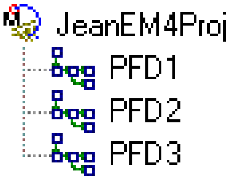
You might observe minor differences when you compare the project process flow diagrams and results of your original and converted projects. For more information about possible differences, see“Project Conversion Differences in Nodes” on page 83..

Project Conversion Code Examples

Project Conversion Scenario

The project conversion code examples use the following scenario:

A SAS Enterprise Miner user named Jean, whose SAS user ID is Jean, is migrating a SAS Enterprise Miner 4.3 client/server data mining project that is named JeanEM4Proj for use with SAS Enterprise Miner 5.3 or later. The project JeanEM4Proj contains three process flow diagrams. The files for the project JeanEM4Proj are stored on Jean's SAS Enterprise Miner 4.3 Windows client machine. This is how the JeanEM4Proj project appears in the Project Navigator of the SAS Enterprise Miner 4.3 software:



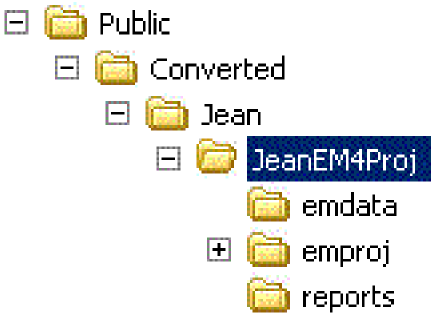
JeanEM4Proj
PFD1
PFD2
PFD3

Jean uses the project properties information in SAS Enterprise Miner 4.3 to determine the location of the JeanEM4Proj project files. Jean knows that project files aren't affected by the conversion process, but Jean prefers to work with a copy of the project in a local work directory. Jean closes SAS Enterprise Miner 4.3, and then uses a file utility to copy the entire project file structure to a work directory in the following location on the client PC:

```
C:\Public\Converted\Jean\JeanEM4Proj
```

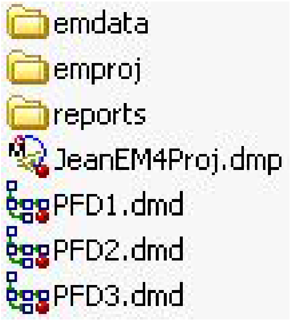
You must close SAS Enterprise Miner 4.3 before copying the Enterprise Miner 4.3 project file structure to another location. When open, SAS Enterprise Miner 4.3 places locks on some project files that can interfere with file copy operations.

This is how the copied SAS Enterprise Miner 4.3 version of the JeanEM4Proj project folders appear in Jean's work directory:



Public
Converted
Jean
JeanEM4Proj
emdata
emproj
reports

Opening the JeanEM4Proj directory reveals the JeanEM4Proj.dmp project file and a DMD file for each of the three process flow diagrams that belong to the JeanEM4Proj project.



emdata
emproj
reports
JeanEM4Proj.dmp
PFD1.dmd
PFD2.dmd
PFD3.dmd

Jean's Windows PC has network access to a projects folder on the SAS Enterprise Miner 5.3 server. The SAS Enterprise Miner 5.3 server uses the Windows operating environment. The network path to the projects folder that Jean has chosen to use on the SAS Enterprise Miner 5.3 server is as follows:

```
\\EM5Server\EM5Projects\ConvertedEM4\Jean
```

If Jean's conversion creates a complete SAS Enterprise Miner 5.3 or later project folder structure, the converted SAS Enterprise Miner project will be named JeanEM5Proj. The target location for the JeanEM5Proj project folder structure on the SAS Enterprise Miner 5.3 server is

```
\\EM5Server\EM5Projects\ConvertedEM4\Jean\JeanEM5Proj
```

If Jean's conversion creates a SAS Enterprise Miner 5.3 or later project creation batch code file, the project creation batch code file will be named JeanEM5ProjBatch.sas. The project creation batch code file can be saved to any location, but Jean prefers to save the JeanEM5ProjBatch.sas file to the following project folder on the SAS Enterprise Miner 5.3 server:

```
\\EM5Server\EM5Projects\ConvertedEM4\Jean\JeanEM5ProjBatch.sas
```

When the project creation batch code is run from the above location, the project will be created in the following location:

```
\\EM5Server\EM5Projects\ConvertedEM4\Jean\JeanEM5Proj
```

Jean organizes the following information before creating the conversion macro code:

- the path on Jean's workstation to the SAS Enterprise Miner 4.x project copy that Jean will convert
- the path to the location where Jean's converted complete project folder structure will be saved, if created
- the path to the location where Jean's converted project creation batch code file will be saved, if created
- the name that Jean will use for the converted project or project creation batch code file.

Jean is now ready to create the conversion macro code.

Create a SAS Enterprise Miner 5.3 or Later Complete Project Folder Structure

Creating a SAS Enterprise Miner 5.3 or later complete project folder structure is the most direct method for converting SAS Enterprise Miner 4.x projects into SAS Enterprise Miner 5.3 or later projects while preserving past project logs, charts, and assessment results. This is the best method to use when the SAS Enterprise Miner 4.x client that is used to perform the conversion has network access to the designated project storage folder on the SAS Enterprise Miner 5.3 or later server. In this situation, Jean can choose to create the converted SAS Enterprise Miner 5.3 or later project in the designated project storage folder on the server, or Jean can choose to create the converted SAS Enterprise Miner 5.3 or later project in the project conversion work directory on the SAS Enterprise Miner 4.3 client workstation and then copy the complete converted project folder structure to the designated project storage folder on the server. Either choice creates identical results, a converted project that transfers saved project results, charts, and logs.

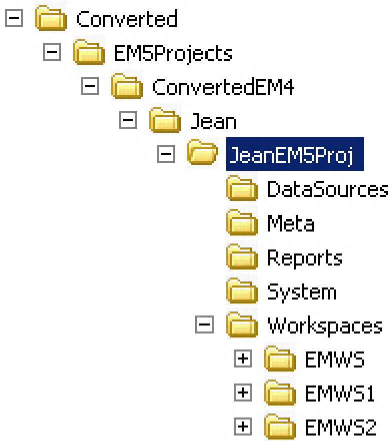
Jean is creating a complete project folder structure, so the CREATE= macro option is set to Y. No project conversion batch code file will be saved, so the FILEREF= option will not be used. Jean decides to let the conversion macro create the converted project in the designated project storage folder that is located on the SAS Enterprise Miner 5.3 server.

Jean launches a SAS 9.1.3 session on a workstation that has access to both the SAS Enterprise Miner 5.3 server and the SAS Enterprise Miner 4.3 project, and submits the following code to the SAS Program Editor:

```
/* use x to allocate and include the catalog entry */
/* that contains the conversion macro source code, */
/* then de-allocate x to clean up                  */
```

```
filename x catalog 'SASHELP.EMCONVRT.EM_CONVERT.SOURCE';
%inc x;
filename x;

/* specify path and name values for placeholder    */
/* variables em4project, em5path, em5name          */

%let em4project=C:\Public\Converted\Jean\JeanEM4Proj;
%let em5path=\\EM5Server\EM5Projects\ConvertedEM4\Jean;
%let em5name=JeanEM5Proj;

/* submit macro using placeholder variables and    */
/* specify CREATE=Y                                */

%em_convert(em4project=<em4project>, projectPath=<em5path>,
  projectName=<em5name>, create=Y);
```

After the code runs, Jean uses a file utility to examine the newly converted project in the destination directory on the SAS Enterprise Miner 5.3 server. The converted SAS Enterprise Miner 5.3 or later project folder structure was created in the correct location. Expanding the JeanEM5Proj project folder reveals the standard SAS Enterprise Miner 5.3 or later project directory structure.



Expanding the Workspaces folder reveals the EMWS subdirectories for the three process flow diagrams that belong to the project. Expanding one of the EMWS folders reveals the SAS catalog files for the process flow diagram, data sets that were created in the diagram workspace, and catalog files for the process flow diagram, logs, results, and plots. Each EMWS folder also contains a subfolder for each node in the process flow diagram.

Jean closes the SAS session on the SAS Enterprise Miner 4.3 client. Jean is ready to start a client SAS Enterprise Miner 5.3 or later session to open and run the process flow diagrams in the newly converted JeanEM5Proj project. After running each process flow diagram, Jean uses the SAS Enterprise Miner 5.3 or later Results window to compare the results that were generated by each process flow diagram in SAS Enterprise Miner 4.3 and SAS Enterprise Miner 5.3 or later.

Create a SAS Enterprise Miner 5.3 or Later Project Creation Batch Code File

Creating a SAS Enterprise Miner 5.3 or later project creation batch code file is useful when it is not important to transfer the existing SAS Enterprise Miner 4.x project logs, charts, and assessment results to the converted SAS Enterprise Miner 5.3 or later project.

The SAS Enterprise Miner 5.3 or later project creation batch code file is also useful in situations where network connectivity between the SAS Enterprise Miner 4.3 client and the SAS Enterprise Miner 5.3 or later server is restricted or unavailable.

SAS Enterprise Miner 5.3 or later project creation batch code is small in size, portable, operating environment-independent, and easily transported by file transfer protocol, disk, CD, thumb drive, or similar media. The project creation batch code file contains all of the information that is required to create SAS Enterprise Miner 5.3 or later complete project folder structures. The tradeoff for the small size and portability of the project creation batch code file is that the SAS Enterprise Miner 5.3 or later projects that they create do not transfer the SAS Enterprise Miner 4.x project logs, charts, or assessment results.

For this example, assume that Jean does not have network connectivity between the existing SAS Enterprise Miner 4.x client and the SAS Enterprise Miner 5.3 or later Windows server. Jean will generate the project creation batch code file in the project work directory that she created on the SAS Enterprise Miner 4.3 client PC.

Jean does not want to create a SAS Enterprise Miner 5.3 or later project folder structure, so the CREATE= option in the macro will be set to N. Jean wants to save the project creation batch code file, so a FILEREF= argument must be supplied to provide the filename and path specification. After the project creation batch code file is successfully saved, Jean will use media to manually transfer the batch code file to the designated project storage location on the SAS Enterprise Miner 5.3 or later server, where the code can be run to reconstruct the converted project, without its pre-existing logs, charts, or assessment results.

Jean launches a SAS 9.1.3 (or later) session on a workstation that has access to both the SAS Enterprise Miner 5.3 or later server and the SAS Enterprise Miner 4.3 project, and submits the following code to the SAS Program Editor:

```
/* use x to allocate and include the catalog entry */
/* that contains the conversion macro source code, */
/* then de-allocate x to clean up                  */

filename x catalog 'SASHELP.EMCONVRT.EM_CONVERT.SOURCE';
%inc x;
filename x;

/* specify path and name values for placeholder    */
/* variables em4project, em5path, em5name          */

%let em4project=C:\Public\Converted\Jean\JeanEM4Proj;
%let em5path=C:\Public\Converted\Jean;
%let em5name=JeanEM5Proj;

/* fileref specifies filename and path for project */
/* creation batch code file                        */

filename x 'C:\Public\Converted\Jean\JeanEM5ProjBatch.sas';

/* submit macro using placeholder variables,       */
/* specify CREATE=N, provide fileref for project   */
/* creation batch code file to be generated        */

%em_convert(em4project= <&em4project>, projectPath= <&em5path>,
            projectName= <&em5name>, create=N, fileref=x);
```

After the conversion code runs, Jean uses a file utility to examine the newly converted project creation batch file in the destination directory on the SAS Enterprise Miner 4.3 client. The batch code file, named JeanEM5ProjBatch.sas, was created in the desired location. Jean copies the file to portable media and transports the file to Jean's project folder on the SAS Enterprise Miner 5.3 or later server:

```
C:\EM5Projects\ConvertedEM4\Jean\JeanEM5Proj
```

Note: Project creation batch files can be stored and run in any location that is accessible to SAS. Jean's project folder locations in this example are arbitrary. You can freely specify a custom destination for your own project creation batch files.

Once the file JeanEM5ProjBatch.sas is located in Jean's project folder, Jean uses a SAS 9.1.3 (or later) session on the server to run the SAS batch file JeanEM5ProjBatch.sas. The batch file creates the complete project folder structure beneath the JeanEM5Proj project folder. After the project folder structures are created, Jean can open the newly converted project using SAS Enterprise Miner 5.3 or later.

Create a SAS Enterprise Miner 5.3 or Later Complete Project Folder Structure and a SAS Enterprise Miner 5.3 or Later Project Creation Batch Code File

The %EM_CONVERT macro is typically used either to create a complete SAS Enterprise Miner 5.3 or later project folder structure or to save the SAS Enterprise Miner 5.3 or later project creation batch code file. With the exception of transferring SAS Enterprise Miner 4.x project results (which the project creation batch code cannot do), both macro output types generate the same converted SAS Enterprise Miner 5.3 or later project.

Suppose, however, that Jean plans to deploy the converted SAS Enterprise Miner 5.3 or later project on a remote server by transporting the project creation batch code to the server, and then running the batch code there. Also suppose that Jean would like to preview the converted SAS Enterprise Miner 5.3 or later project folder structures before transporting and running the project creation batch code in its new location on the SAS Enterprise Miner 5.3 server. Jean wants to create both types of output in the work area, preview the created project folder structures to ensure they meet requirements, and then transport and run the project creation batch code in its new location with confidence. The path to Jean's temporary work area is

```
C:\Public\Converted\EM5Projects\Temporary
```

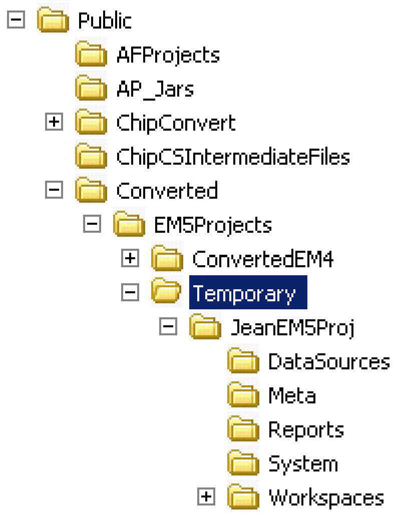
To create both types of output, Jean's macro specifies both CREATE=Y and provides a path and filename argument for the FILEREF= option.

```
/* use x to allocate and include the catalog entry */
/* that contains the conversion macro source code, */
/* then de-allocate x to clean up                  */

filename x catalog 'SASHELP.EMCONVRT.EM_CONVERT.SOURCE';
%inc x;
filename x;

/* specify path and name values for placeholder    */
/* variables em4project, em5path, em5name          */
```

```
%let em4project=C:\Public\Converted\Jean\JeanEM4Proj;
%let em5path=C:\Public\Converted\EM5Projects\Temporary;
%let em5name=JeanEM5Proj;

/* fileref specifies filename and path for project */
/* creation batch code file                         */

filename x 'C:\Public\Converted\EM5Projects\Temporary\JeanEM5ProjBatch.sas';

/* submit macro using placeholder variables,        */
/* specify CREATE=Y, provide fileref for project    */
/* creation batch code file to be generated         */

%em_convert(em4project= <&em4project>, projectPath= <&em5path>,
            projectName= <&em5name>, create=Y, fileref=x);
```

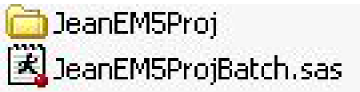
Jean submits the macro code to the Program Editor of a SAS 9.1.3 (or later) session on a Windows workstation that can access the SAS Enterprise Miner 5.3 or later server and the SAS Enterprise Miner 4.3 project to be converted.

After the code runs, Jean uses a file utility to examine the newly converted project folder structure in the project work area.



```
⊟ Public
    AFProjects
    AP_Jars
  ⊞ ChipConvert
    ChipCSIntermediateFiles
  ⊟ Converted
    ⊟ EM5Projects
      ⊞ ConvertedEM4
      ⊟ Temporary
        ⊟ JeanEM5Proj
            DataSources
            Meta
            Reports
            System
          ⊞ Workspaces
```

The Temporary folder and the JeanEM5Proj project structure folders were created correctly. Jean selects the Temporary folder and views the file list to also verify that the project creation batch code file was saved.



JeanEM5Proj
JeanEM5ProjBatch.sas

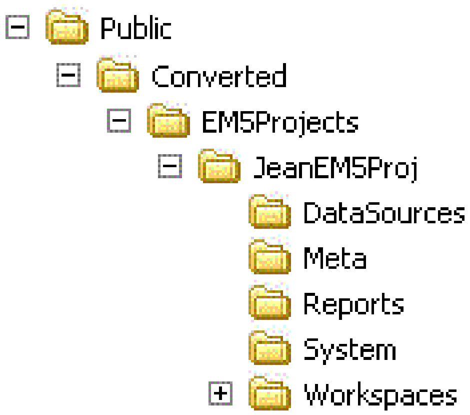
Jean copies the project creation batch code file JeanEM5ProjBatch.sas to a portable media device such as a disk, CD, DVD, or thumb drive. The batch code file is ready to be copied to its destination folder on the remote SAS Enterprise Miner 5.3 or later server, where it can be run to create new project folder structures.

Alternately, Jean could ignore the saved project batch code creation file, and use a ZIP or TAR file utility to compress and save the JeanEM5Proj project folder structure to portable media. The media can be transported to the SAS Enterprise Miner 5.3 or later server and the files restored (unzipped or untarred) to their original form. SAS Enterprise

Miner 5.3 and later can recognize and open the transported project file structure. Unlike a project that is created by running the project batch code file, the transported project folder structure transfers the process flow diagram results from the SAS Enterprise Miner 4.x project.

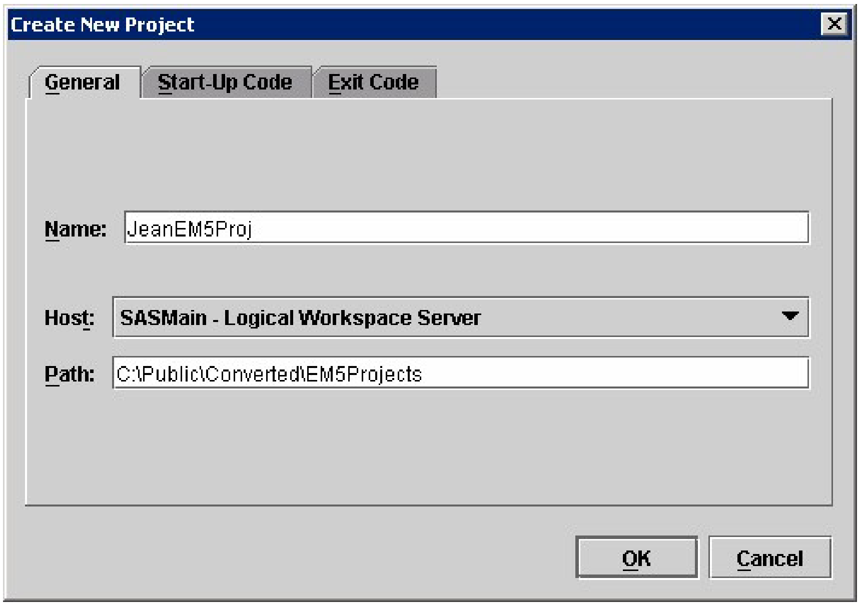
Opening Converted SAS Enterprise Miner 4.x Projects in SAS Enterprise Miner 5.3

The following instructions explain how to open a newly converted SAS Enterprise Miner 5.3 or later project using SAS Enterprise Miner 5.3 or later software. The instructions use the converted project folder structure here:



```
⊟ Public
    ⊟ Converted
        ⊟ EM5Projects
            ⊟ JeanEM5Proj
                  DataSources
                  Meta
                  Reports
                  System
              ⊞ Workspaces
```

Open a SAS Enterprise Miner 5.3 or later session on your SAS Enterprise Miner 5.3 or later server. In the SAS Enterprise Miner introductory window, choose **New Project** to open a **Create New Project** window. In the **Create New Project** window, provide the following information in the following order:



Create New Project

General | Start-Up Code | Exit Code

Name: JeanEM5Proj

Host: SASMain - Logical Workspace Server

Path: C:\Public\Converted\EM5Projects

1. Select your SAS Enterprise Miner 5.3 or later server from the Host list. The SAS Enterprise Miner 5.3 or later server should be the server where your converted SAS

Enterprise Miner 5.3 or later project was copied to or created. In this example, the Host is SASMain - Logical Workspace Server.

2. Use the **Path** field to specify the server path to the folder that contains the project folder for your converted SAS Enterprise Miner 5.3 or later project. In this example, the server path to the project is `C:\Public\Converted\EM5Projects`

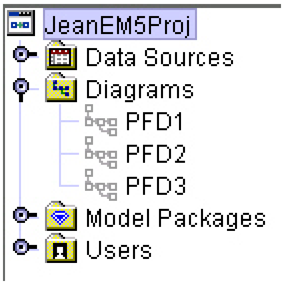
   Note: If the text in the **Path** field of your **Create New Project** window is dimmed and cannot be edited, your SAS Enterprise Miner 5.3 or later administrator has configured the server to use a fixed project path. In this case, consult with your Enterprise Miner administrator, or copy your converted project folder structure to the location specified in the fixed project path.

3. Use the **Name** field to specify the name of your converted SAS Enterprise Miner 5.3 or later project. The name of your SAS Enterprise Miner 5.3 project must *exactly* match the name of the top-level folder in your converted SAS Enterprise Miner 5.3 or later project, which is also the value that was supplied for the %LET EM5NAME= argument in the project conversion macro. Your converted SAS Enterprise Miner 5.3 or later project folder must be a subdirectory of the folder that you specified in the **Path** field.

4. If your converted SAS Enterprise Miner 5.3 or later project requires start or exit code, use the **Start Code** and **Exit Code** tabs of the **Create New Project** window to enter your project's start or exit code. The project conversion macro does not transfer project start code or project exit code for SAS Enterprise Miner 4.x projects.

5. Select **OK** in the **Create New Project** window to open your converted SAS Enterprise Miner 5.3 or later project in the software.

6. When the SAS Enterprise Miner 5.3 or later software opens up your converted project, go to the Project Navigator and expand the Diagrams folder to verify that all of the process flow diagrams that belong to your converted project were created.

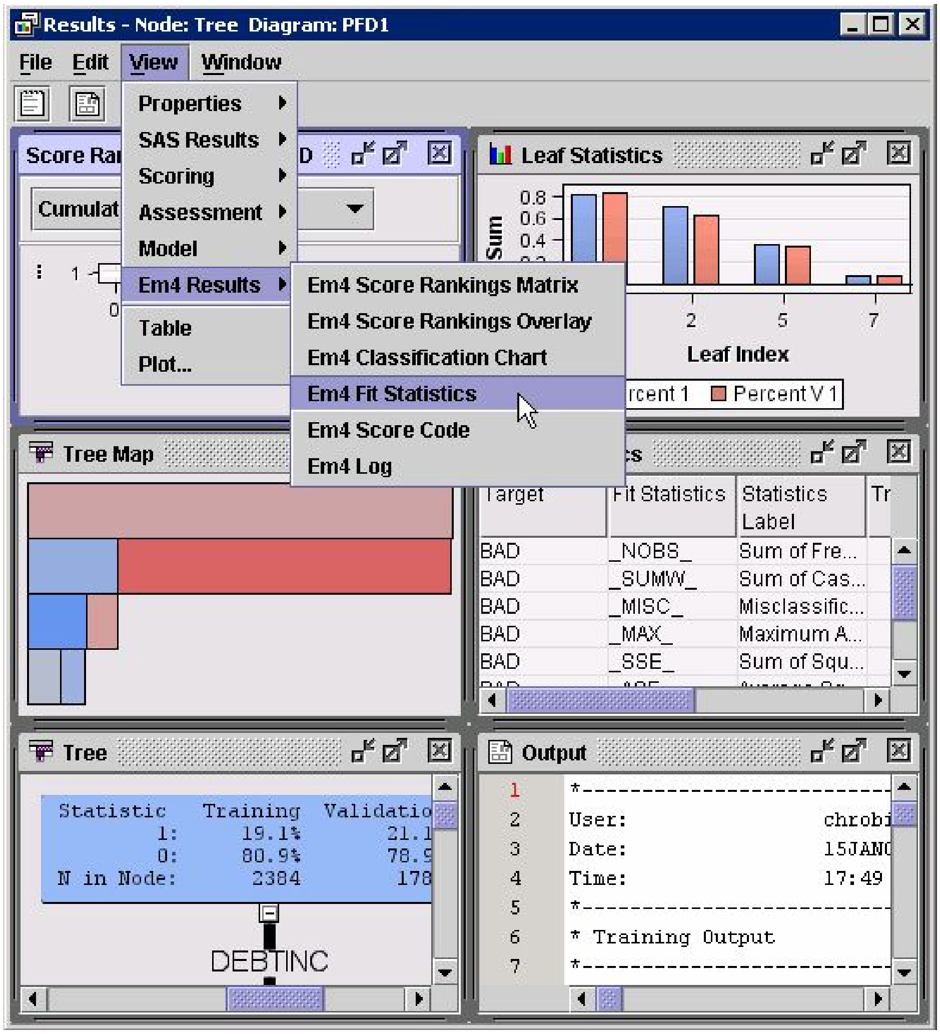


7. You will need to open and run each process flow diagram in your converted project before you can compare the project results that were transferred from SAS Enterprise Miner 4.x to the project results that were produced by the converted SAS Enterprise Miner 5.3 or later process flow diagrams.

8. To compare transferred SAS Enterprise Miner 4.x project process flow diagram results with converted SAS Enterprise Miner 5.3 or later project process flow diagram results, run the converted project's process flow diagrams in SAS Enterprise Miner 5.3 or later, and then open the Results window of any node in the process flow diagram.

   The Results window displays the converted SAS Enterprise Miner 5.3 or later project results by default. For example, you might want to compare the SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 or later assessment results of a Decision Tree node. To view the project results that were transferred from SAS Enterprise Miner 4.x in the SAS Enterprise Miner 5.3 or later Results window, select the following from the main menu:

   View ➔ **Em4Results** ➔ ...



Target	Fit Statistics	Statistics Label	Tr
BAD	_NOBS_	Sum of Fre...	
BAD	_SUMW_	Sum of Cas...	
BAD	_MISC_	Misclassific...	
BAD	_MAX_	Maximum A...	
BAD	_SSE_	Sum of Squ...	

```
Statistic    Training  Validatio
       1:       19.1%       21.1
       0:       80.9%       78.9
N in Node:       2384        178
```

```
*--------------------------
User:                chrobi
Date:                15JAN0
Time:                17:49
*--------------------------
* Training Output
*--------------------------
```

Project Conversion Differences in Nodes

Overview

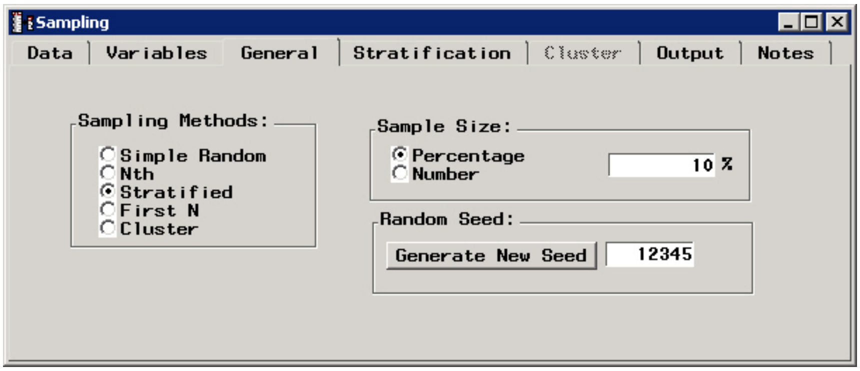
Differences can exist between original SAS Enterprise Miner 4.x and converted SAS Enterprise Miner 5.3 or later process flow diagram results that are not caused by the project conversion process.

The data mining tools that are provided with SAS Enterprise Miner 4.x and with SAS Enterprise Miner 5.3 and later are not identical. The set of data mining tools in SAS Enterprise Miner 5.3 and later is an improved and expanded version of the tools in SAS Enterprise Miner 4.x. Over time, numerous SAS Enterprise Miner 4.x data mining nodes have been refined or redesigned for enhanced configuration and computation. In more than one case, the functionality that was performed by a single node in SAS Enterprise Miner 4.x is performed by two separate, more configurable nodes in SAS Enterprise Miner 5.3 and later. There might also be enhancements and changes in the SAS procedure code that underlies a SAS Enterprise Miner data mining node.

When comparing original and converted project process flow diagrams and their results, you should expect to see differences in the implementation of the following SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 or later nodes.

Sample Node

When performing stratified sampling in a process flow diagram, the default settings of SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later Sample nodes vary. In SAS Enterprise Miner 4.x, you use the **General** tab of the Sampling node to specify the Sampling Method as **Stratified**.

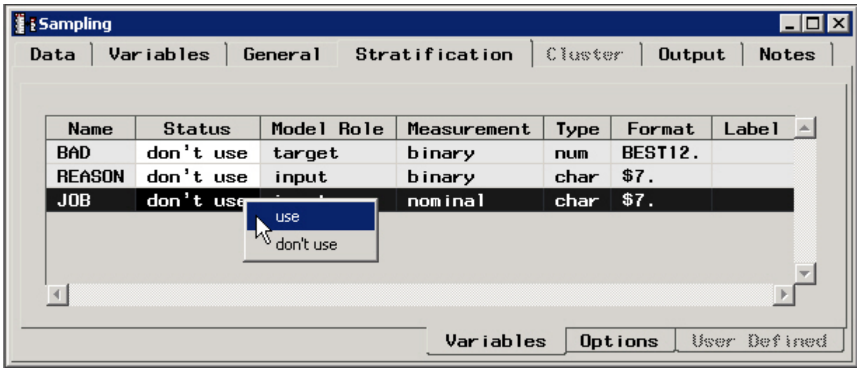


Sampling Methods:
- ○ Simple Random
- ○ Nth
- ● Stratified
- ○ First N
- ○ Cluster

Sample Size:
- ● Percentage 10 %
- ○ Number

Random Seed: 12345

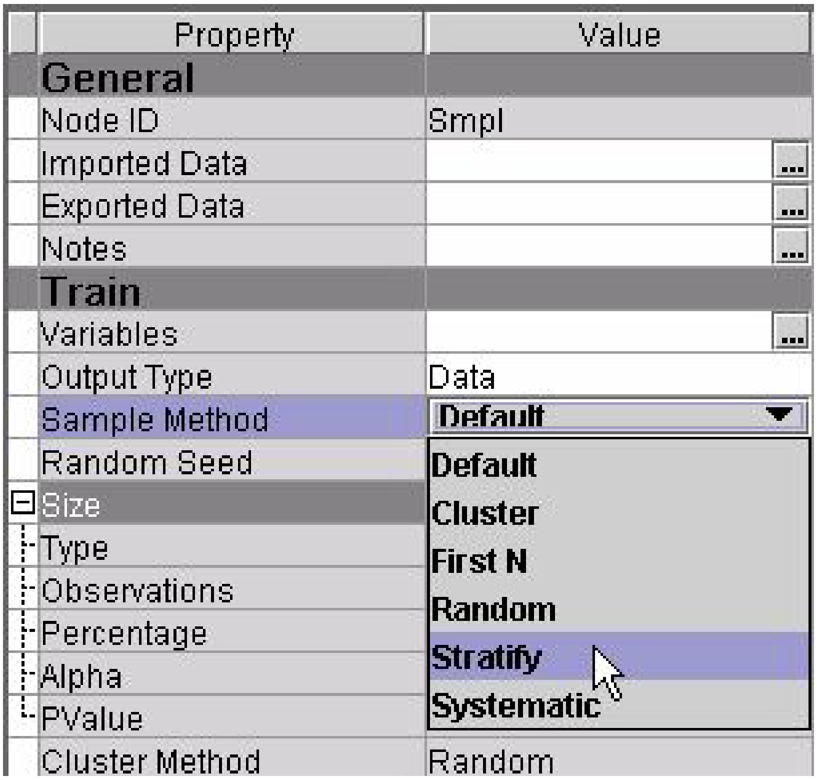
Then, you use the **Stratification** tab to choose your stratification variable from the available nominal and ordinal variables in your data set:



Name	Status	Model Role	Measurement	Type	Format	Label
BAD	don't use	target	binary	num	BEST12.	
REASON	don't use	input	binary	char	$7.	
JOB	don't use		nominal	char	$7.	

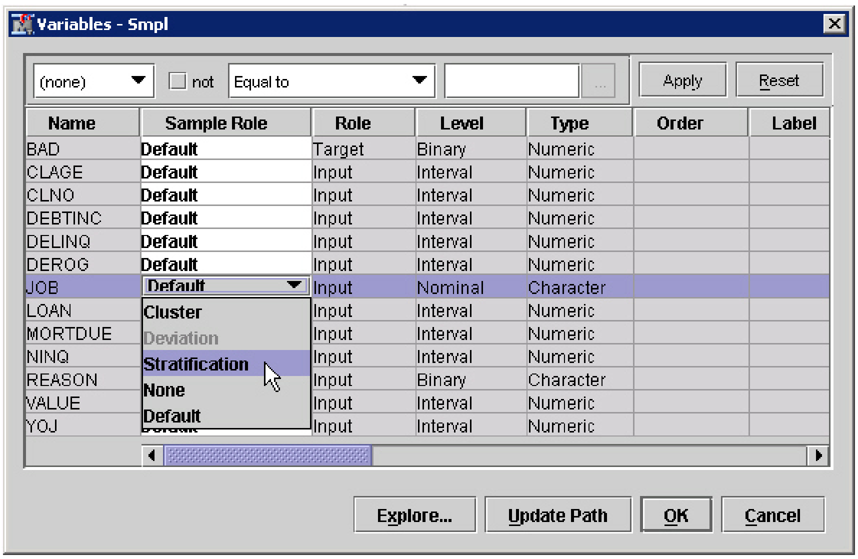
The default setting for the status of SAS Enterprise Miner 4.x class variables in the **Stratification** tab of the Sampling node is **don't use**. If no stratification variable status is set to **use**, then no stratification sampling will be performed. Instead, the Sampling node reverts to simple random sampling, because data cannot be sampled by strata without a declared stratification variable. You must change the status of a class variable from its default setting of **don't use** to **use** to perform stratified sampling.

In SAS Enterprise Miner 5.x and later, you use the Properties panel for the Sample node to choose **Stratify** as the **Sample Method**:



Property	Value
General	
Node ID	Smpl
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Output Type	Data
Sample Method	Default
Random Seed	
Size	
Type	
Observations	
Percentage	
Alpha	
PValue	
Cluster Method	Random

Then, you must open the **Variables** property window to choose your stratification variable by assigning the **Sample Role**:



Variables - Smpl

(none) □ not Equal to

Name	Sample Role	Role	Level	Type	Order	Label
BAD	Default	Target	Binary	Numeric		
CLAGE	Default	Input	Interval	Numeric		
CLNO	Default	Input	Interval	Numeric		
DEBTINC	Default	Input	Interval	Numeric		
DELINQ	Default	Input	Interval	Numeric		
DEROG	Default	Input	Interval	Numeric		
JOB	Default	Input	Nominal	Character		
LOAN		Input	Interval	Numeric		
MORTDUE		Input	Interval	Numeric		
NINQ		Input	Interval	Numeric		
REASON		Input	Binary	Character		
VALUE		Input	Interval	Numeric		
YOJ		Input	Interval	Numeric		

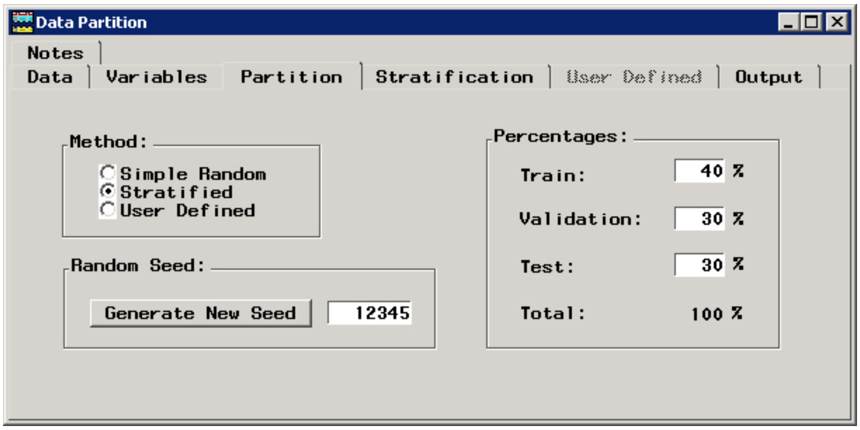
You must assign the **Sample Role** of **Stratification** to your desired stratification variable, and you must also assign the **Sample Role** of **None** to the remaining variables. The default **Sample Role** for variables in SAS Enterprise Miner 5.x and later is **Default**, which means that if a class variable is not explicitly configured as the stratification variable, the node will use all class (binary, nominal, and ordinal) variables by default to perform stratified sampling. If there are no class targets, then the node will perform random sampling by default.

You must use care to ensure that Sample node settings are consistent in both versions of your SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later process flow diagrams. Otherwise, the default settings of the Sample node might differ. Different data samples might change the analytical results of your converted SAS Enterprise Miner 5.3 and later process flow diagram.

Data Partition Node

The default settings for stratified data partitioning vary between nodes in SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later. Variances between the default settings of the Data Partition node in the SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later process flow diagrams can disturb the data partitioning results in some cases.

When performing stratified data partitioning in a process flow diagram, the default settings of the SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later Data Partition nodes vary. In SAS Enterprise Miner 4.x, use the **Partition** tab of the Data Partition node to specify **Stratified** as the data partitioning method.

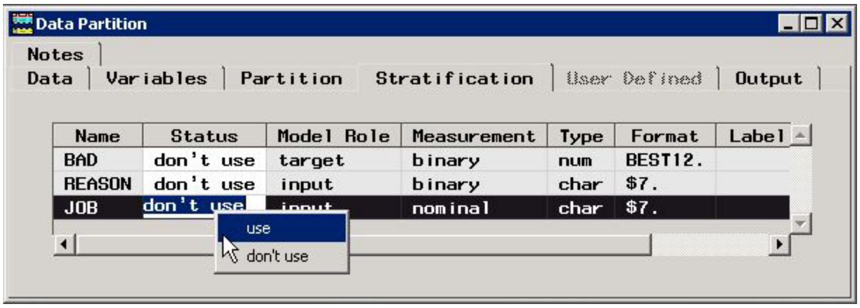


Method:
○ Simple Random
● Stratified
○ User Defined

Random Seed: 12345

Percentages:
Train: 40 %
Validation: 30 %
Test: 30 %
Total: 100 %

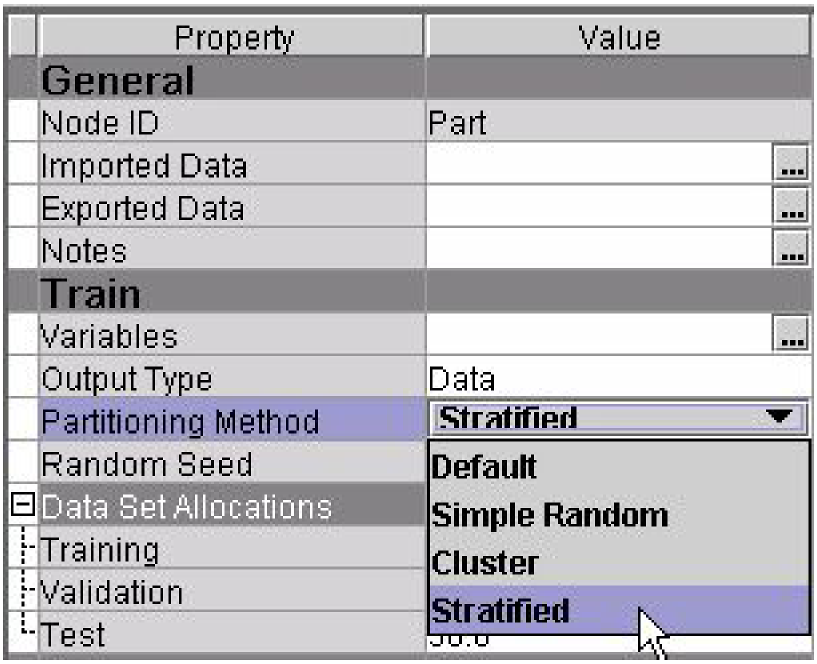
Then, use the **Stratification** tab to choose your stratification variable from the available class variables in your data set:



Name	Status	Model Role	Measurement	Type	Format	Label
BAD	don't use	target	binary	num	BEST12.	
REASON	don't use	input	binary	char	$7.	
JOB	don't use	input	nominal	char	$7.	

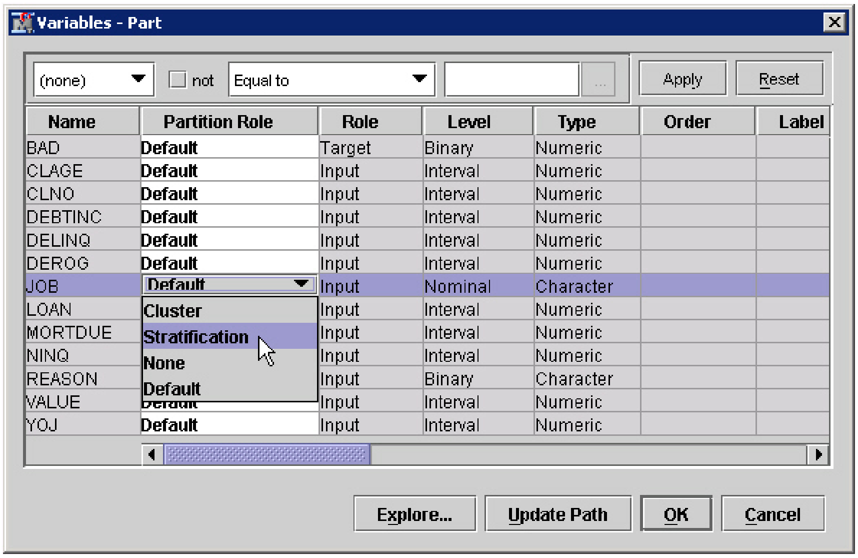
The default setting for the status of SAS Enterprise Miner 4.x class variables in the **Stratification** tab of the Data Partition node is **don't use**. If no stratification variable status is set to **use**, then no stratification is performed. Instead, the Data Partition node reverts to performing simple random data partitioning, because data cannot be partitioned by strata without a declared stratification variable. You must change the status of a class variable from its default setting of **don't use** to **use** to perform stratified data partitioning.

In SAS Enterprise Miner 5.3 and later, you use the Data Partition node Properties panel to choose the partitioning method:



Property	Value
General	
Node ID	Part
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Output Type	Data
Partitioning Method	Stratified
Random Seed	
⊟ Data Set Allocations	
Training	
Validation	
Test	

If you specify **Stratified** as the **Partitioning Method**, you must open the Variables property window to choose your stratification variable by specifying the **Partition Role** for the stratification variable:



Name	Partition Role	Role	Level	Type	Order	Label
BAD	Default	Target	Binary	Numeric		
CLAGE	Default	Input	Interval	Numeric		
CLNO	Default	Input	Interval	Numeric		
DEBTINC	Default	Input	Interval	Numeric		
DELINQ	Default	Input	Interval	Numeric		
DEROG	Default	Input	Interval	Numeric		
JOB	Default	Input	Nominal	Character		
LOAN		Input	Interval	Numeric		
MORTDUE		Input	Interval	Numeric		
NINQ		Input	Interval	Numeric		
REASON		Input	Binary	Character		
VALUE		Input	Interval	Numeric		
YOJ	Default	Input	Interval	Numeric		

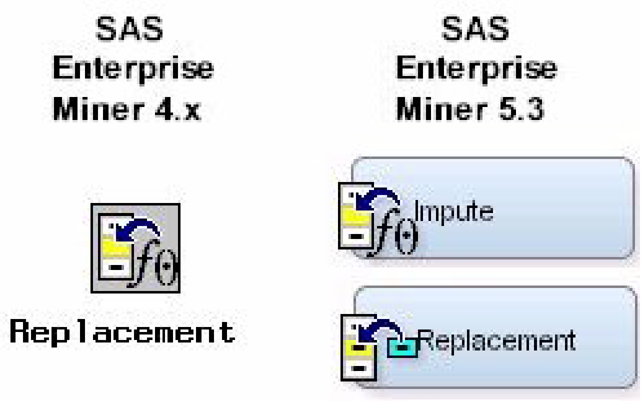
Assign the **Partition Role** of **Stratification** to your desired stratification variable, and then assign the Partition Role of **None** to the remaining variables. The default Partition Role for Data Partition variables in SAS Enterprise Miner 5.3 and later is **Default**, which means that if a class variable is not explicitly configured as the stratification variable, the node will use the class target variable as the stratification variable by default. If no class target is found, all class (binary, nominal, and ordinal) variables will be used to perform stratified data partitioning.

You must use care to ensure that Data Partition node configurations match in both versions of your SAS Enterprise Miner 4.x and SAS Enterprise Miner 5.3 and later process flow diagrams. Otherwise, the default settings of the Data Partition node can differ between SAS Enterprise Miner 4.x to SAS Enterprise Miner 5.3 and later.

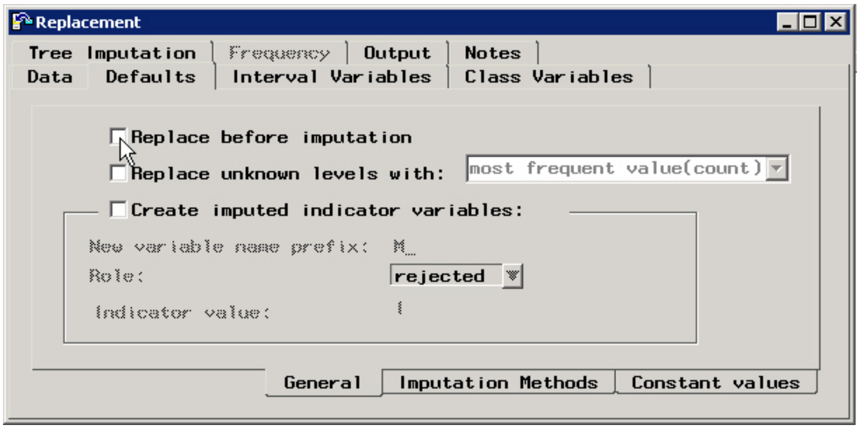
Differently partitioned data can alter the analytical results of your converted SAS Enterprise Miner 5.3 or SAS Enterprise Miner 6.2 process flow diagram.

Replacement and Impute Nodes

The Replacement node in SAS Enterprise Miner 4.x performs both data replacement and data imputation functions. SAS Enterprise Miner 5.3 and later uses two nodes to perform those functions: the SAS Enterprise Miner 5.3 and later Replacement node performs data replacement, and the SAS Enterprise Miner 5.3 and later Impute node performs data imputation. SAS Enterprise Miner 4.x process flow diagrams that contain a Replacement node, when converted for use with SAS Enterprise Miner 5.3 and later, will contain both a Replacement node and an Impute node in the new diagram.



The order of data replacement and data imputation are configurable in the SAS Enterprise Miner 4.x Replacement node. The order of the Replacement and Impute nodes in converted SAS Enterprise Miner 5.3 and later process flow diagrams depends on how the SAS Enterprise Miner 4.x Replacement node was configured.



□ Replace before imputation

□ Replace unknown levels with: most frequent value(count)

□ Create imputed indicator variables:

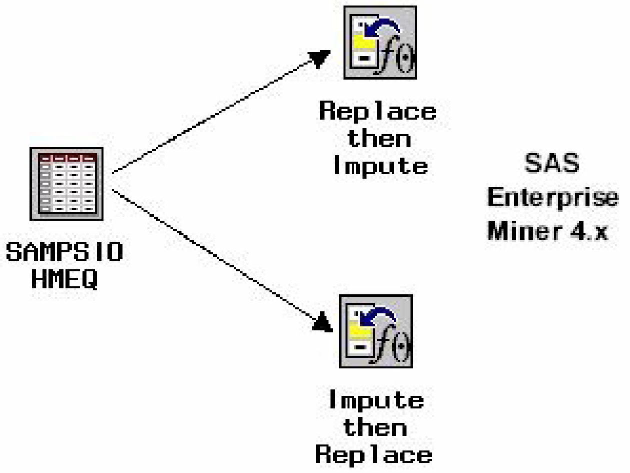
New variable name prefix: M_

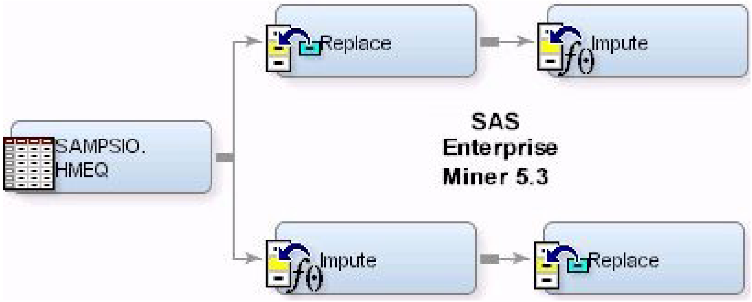
Role: rejected

Indicator value: 1

The order of data replacement and data imputation in the SAS Enterprise Miner 4.x Replacement node is configured in the **General** subtab of the Replacement node's **Defaults** tab. The node performs data imputation before performing data replacement, unless the **Replace before imputation** check box is selected.

The process flow diagrams here illustrate how the order of replace and impute operations affects the conversion process for SAS Enterprise Miner 4.x process flow diagrams:





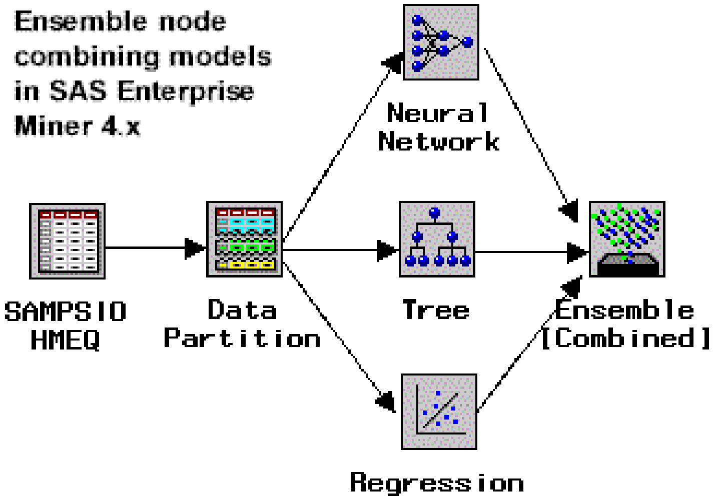
A significant difference exists between the SAS Enterprise Miner 4.x Impute node and the SAS Enterprise Miner 5.3 and later Impute node that must be considered when comparing analytical results produced by the two project versions. The Impute node in SAS Enterprise Miner 4.x bases its imputation values on a sample of the submitted data set. The Impute node in SAS Enterprise Miner 5.3 uses the entire submitted data set to base its imputation values on. As a result, the SAS Enterprise Miner 5.3 and later imputation algorithm tends to produce a slightly more accurate variable imputation than an imputation that was produced in SAS Enterprise Miner 4.x.

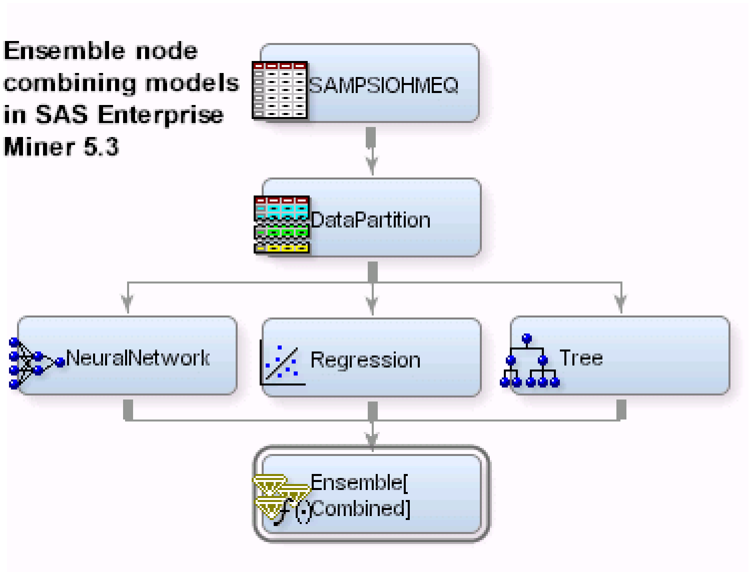
Ensemble and Group Processing Nodes

In SAS Enterprise Miner 4.x, the Ensemble node performs multiple functions. The Ensemble node in SAS Enterprise Miner 4.x can be used to combine different models. The Ensemble node in SAS Enterprise Miner 4.x can also be used with the Group Processing node to accumulate models such as stratification models and bagging or boosting models. When you convert a SAS Enterprise Miner 4.x process flow diagram that contains an Ensemble node for use with SAS Enterprise Miner 5.3 or later, the new process flow diagram contains either an Ensemble node or an End Group node, depending on whether the original Ensemble node performed model combination or model accumulation.

Using the Ensemble node to combine models can improve the stability of disparate nonlinear models. When the individual models use class targets, the combined model averages the posterior probabilities of the individual models. When the individual models use interval targets, the combined model averages the predicted values of the individual models.

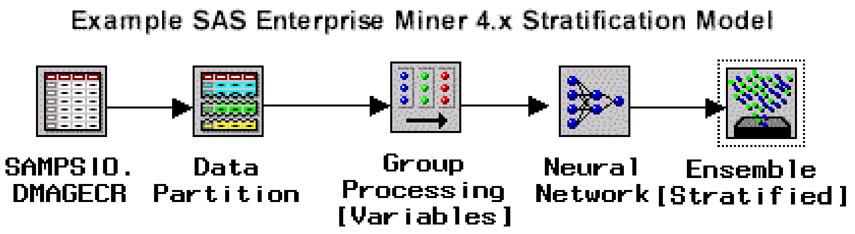
The following example shows an Ensemble node used to combine models in a SAS Enterprise Miner 4.x process flow diagram. The example also shows the process flow diagram after it is converted for use with SAS Enterprise Miner 5.3 or later. The Ensemble node still combines the models in the converted diagram:

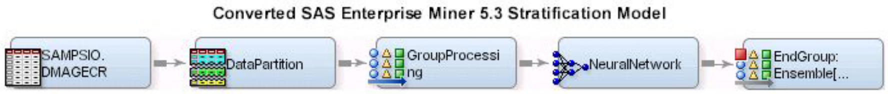




When you perform group processing in SAS Enterprise Miner 4.x, you can use the Ensemble node to accumulate individual group models. Group processing in SAS Enterprise Miner 4.x process flow diagrams is performed using a single Group Processing node. Group processing in SAS Enterprise Miner 5.3 and later process flow diagrams is performed using two nodes, a Group Processing node and an End Group node. The End Group node defines the end of the portion of the process flow diagram that is intended for group processing. The End Group node in SAS Enterprise Miner 5.3 and later also performs the model accumulation function that the Ensemble node performs in SAS Enterprise Miner 4.x.

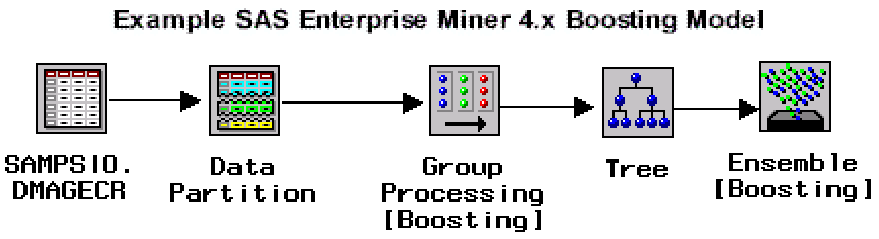
A stratification model is an accumulation model that trains models over segments of the training data and then combines the scoring code from the individual models. Stratification models use a Group Processing node and a modeling node to create a separate model for each level of the stratification variable. The Ensemble node in SAS Enterprise Miner 4.x accumulates the multiple models built using stratified variables. As seen in this example, the End Group node performs the model accumulation task in SAS Enterprise Miner 5.3 and later:

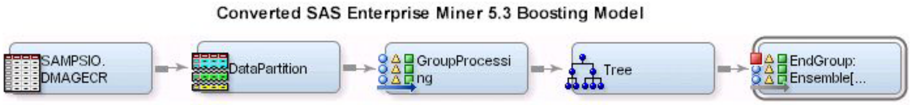

Example SAS Enterprise Miner 4.x Stratification Model
SAMPSIO. DMAGECR
Data Partition
Group Processing [Variables]
Neural Network
Ensemble [Stratified]



Bagging and boosting models are accumulation models that resample the training data and create separate models for each sample. The predicted values (for models with interval targets) or the posterior probabilities (for models with class targets) are then aggregated to form the ensemble model. In SAS Enterprise Miner 4.x, the Ensemble node performs model accumulation for bagging and boosting models. In SAS Enterprise Miner 5.3 and later, the End Group node performs model accumulation for bagging and boosting models.

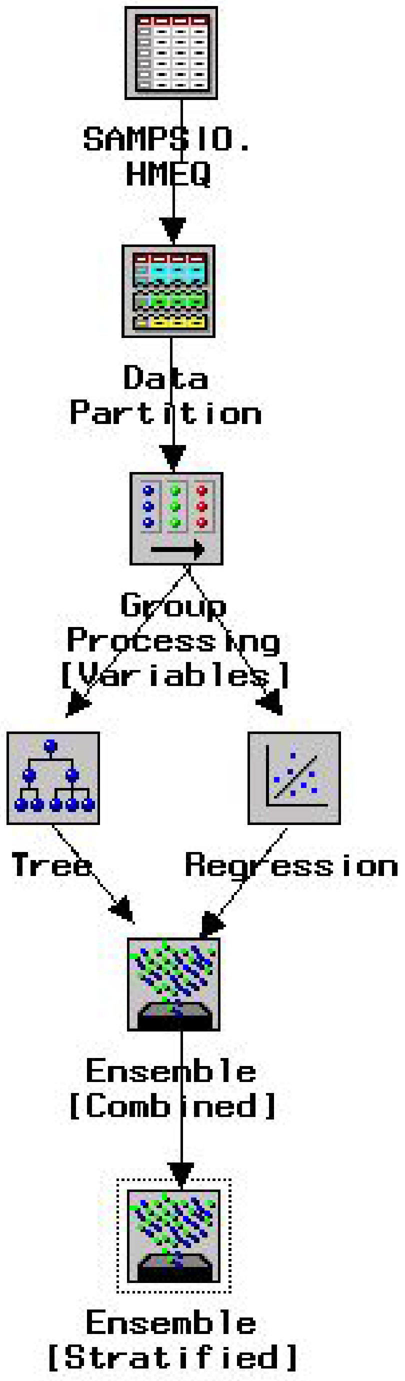
The following example shows a SAS Enterprise Miner 4.x process flow diagram that uses an Ensemble node to perform model accumulation for a boosting model. The example also shows a converted SAS Enterprise Miner 5.3 process flow diagram where the model accumulation and ensemble is performed by the End Group node.

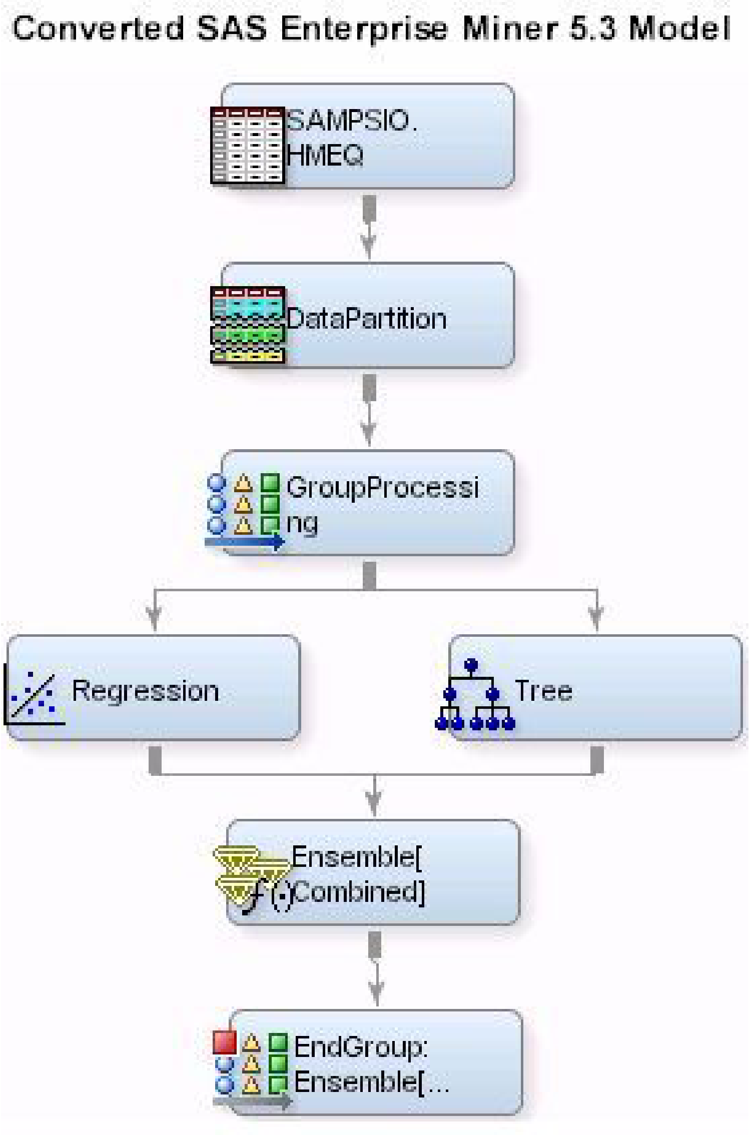

Example SAS Enterprise Miner 4.x Boosting Model
SAMPSIO. DMAGECR
Data Partition
Group Processing [Boosting]
Tree
Ensemble [Boosting]



The last example shows a SAS Enterprise Miner 4.x process flow diagram that uses Ensemble nodes both to combine models and to perform model accumulation for group processing. The converted SAS Enterprise Miner 5.3 and later process flow diagram uses an Ensemble node to perform model combination and an End Group node to perform model accumulation.

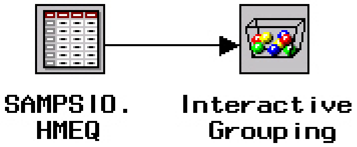
Example SAS Enterprise Miner 4.x Model



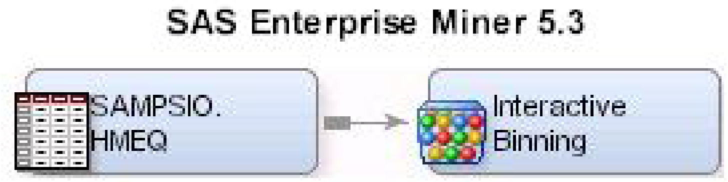


Interactive Grouping Node

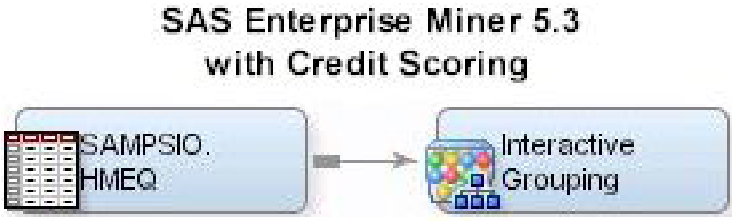
When you convert a SAS Enterprise Miner 4.x process flow diagram that contains an Interactive Grouping node for use with SAS Enterprise Miner 5.3 and later, the converted diagram will contain one of two nodes, as follows:



If your SAS Enterprise Miner 5.3 or later license does not include Credit Scoring for SAS Enterprise Miner, the Interactive Grouping node in your SAS Enterprise Miner 4.x process flow diagram is converted to an Interactive Binning node in the SAS Enterprise Miner 5.3 or later process flow diagram.

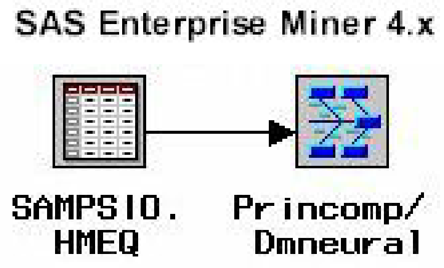


If your SAS Enterprise Miner 5.3 or later license includes Credit Scoring for SAS Enterprise Miner, the Interactive Grouping node in your SAS Enterprise Miner 4.x process flow diagram is converted to an Interactive Grouping node in the SAS Enterprise Miner 5.3 or later process flow diagram.

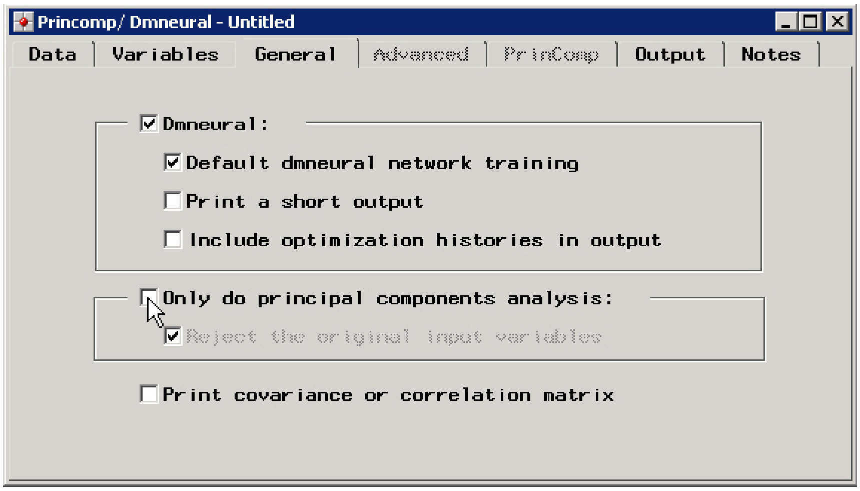


Principal Component and DMNeural Nodes

In SAS Enterprise Miner 4.x, the Princomp/DMNeural node can be configured to perform either DMNeural network training or principal components analysis. SAS Enterprise Miner 5.3 and later uses two nodes to perform these functions. The SAS Enterprise Miner 5.3 and later DMNeural node performs DMNeural network training, and the SAS Enterprise Miner 5.3 and later Principal Components node performs principal component analysis.



SAS Enterprise Miner 4.x process flow diagrams that contain a Princomp/DMNeural node, when converted for use with SAS Enterprise Miner 5.3 and later, will contain either a DMNeural node or a Principal Components node, depending on how the SAS Enterprise Miner 4.x Princomp/DMNeural node was configured.



☑ Dmneural:

☑ Default dmneural network training

□ Print a short output

□ Include optimization histories in output

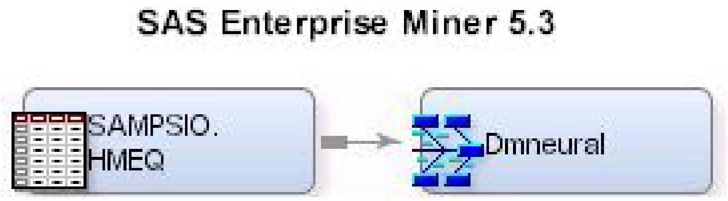
□ Only do principal components analysis:

☑ Reject the original input variables

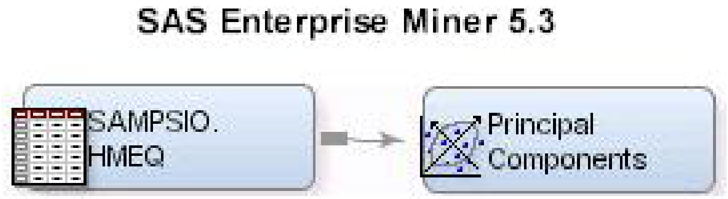
□ Print covariance or correlation matrix

Use the **General** tab of the SAS Enterprise Miner 4.x Princomp/DMNeural node to specify whether to perform DMNeural network training or principal components analysis. DMNeural network training is performed when the **Dmneural** box is checked, and principal components analysis is performed when the **Only do principal components analysis** box is checked. Both boxes cannot be checked at the same time.

If the Princomp/DMNeural node in the SAS Enterprise Miner 4.x process flow diagram is configured to perform DMNeural network training, the converted SAS Enterprise Miner 5.3 and later process flow diagram will replace the Princomp/DMNeural node with a DMNeural node:



If the Princomp/DMNeural node in the SAS Enterprise Miner 4.x process flow diagram is configured to perform principal components analysis, the converted SAS Enterprise Miner 5.3 and later process flow diagram will replace the Princomp/DMNeural node with a Principal Components node:



Tree Node

The SAS Enterprise Miner 5.3 and later Decision Tree node uses an improved version of the splitting algorithm that was used in SAS Enterprise Miner 4.x. As a result, the improved Decision Tree algorithm in SAS Enterprise Miner 5.3 or later might produce results that vary slightly from the results of the SAS Enterprise Miner 4.x Tree node.

Link Analysis Node

SAS Enterprise Miner 4.x uses the Link Analysis node to examine the linkage between effects in complex systems. SAS Enterprise Miner 5.3 and later do not support Link Analysis. When you convert a SAS Enterprise Miner 4.x process flow diagram that contains a Link Analysis node, the resulting SAS Enterprise Miner 5.3 or later process flow diagram displays a nonfunctional placeholder for the Link Analysis process. The placeholder Link Analysis node in SAS Enterprise Miner 5.3 and later do not perform any analytical calculations. The placeholder node is able to display only the results that were generated by the Link Analysis node in the original SAS Enterprise Miner 4.x process flow diagram.

Reporter Node

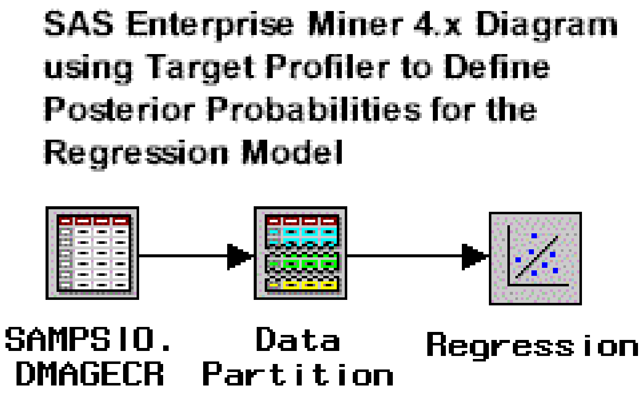
The HTML report files that the SAS Enterprise Miner 4.x Reporter node creates are not transferred to converted SAS Enterprise Miner 5.3 and later projects. If your converted SAS Enterprise Miner 5.3 and later project contains process flow diagrams that use the Reporter node, you can run the process flow diagrams in SAS Enterprise Miner 5.3 or later and save the Reporter node output as an LST or PDF file in the project's Reports folder.

Subdiagrams

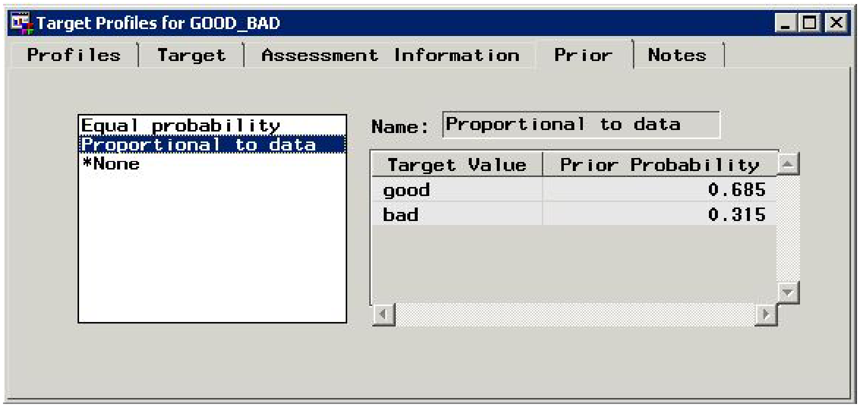
The Subdiagram node in SAS Enterprise Miner 4.x is not supported in SAS Enterprise Miner 5.3 and later. If you convert a SAS Enterprise Miner 4.x process flow diagram that uses a Subdiagram node, the converted SAS Enterprise Miner 5.3 and later process flow diagram will not have a Subdiagram node. Instead, the converted SAS Enterprise Miner process flow diagram will accurately reproduce the detailed diagram substructure that was represented by the Subdiagram node in SAS Enterprise Miner 4.x. The subdiagram portion of the process flow diagram will be enclosed between Enter and Exit nodes.

Target Profiler

In SAS Enterprise Miner 4.x, you could use the Target Profiler to define or modify a target profile. Good target profiles produce optimal decisions that are based on a user-supplied decision matrix and output from a subsequent modeling procedure. In SAS Enterprise Miner 4.x you could access the Target Profiler through the Variables tab of any modeling node. In SAS Enterprise Miner 5.3 and later, you can only access Target Profiler information through an Input Data node or a Decision node. If you convert a SAS Enterprise Miner 4.x process flow diagram that has one or more modeling nodes that use the Target Profiler, the converted SAS Enterprise Miner 5.3 and later process flow diagram will add a Decision node before each modeling node that defined or modified the Target Profiler in SAS Enterprise Miner 4.x. For example, consider the simple SAS Enterprise Miner 4.x process flow diagram here. The Target Profiler is used to specify prior probabilities for the target before modeling, as shown in the Prior tab of the Target Profiles for the GOOD_BAD window:



SAS Enterprise Miner 4.x Diagram using Target Profiler to Define Posterior Probabilities for the Regression Model



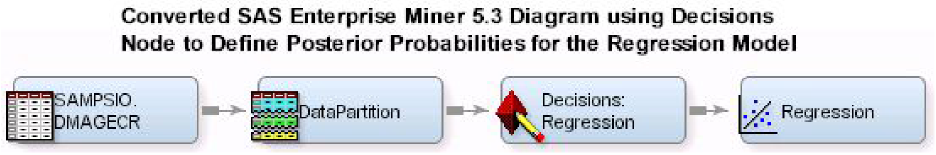
Target Profiles for GOOD_BAD

Equal probability
Proportional to data
*None

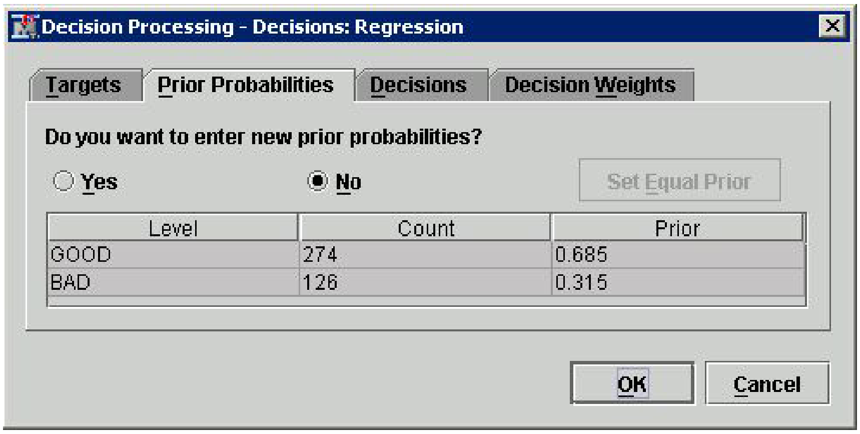
Name: Proportional to data

Target Value	Prior Probability
good	0.685
bad	0.315

When the SAS Enterprise Miner 4.x project that contains this process flow diagram is converted for use with SAS Enterprise Miner 5.3 or later, the converted diagram contains a Decisions node that precedes the Regression node. The Decisions node in the SAS Enterprise Miner 5.3 and later process flow diagram contains the prior probabilities that were specified for the target variable GOOD_BAD in the Target Profiler of the SAS Enterprise Miner 4.x Regression node.



Converted SAS Enterprise Miner 5.3 Diagram using Decisions Node to Define Posterior Probabilities for the Regression Model



Decision Processing - Decisions: Regression

Do you want to enter new prior probabilities?

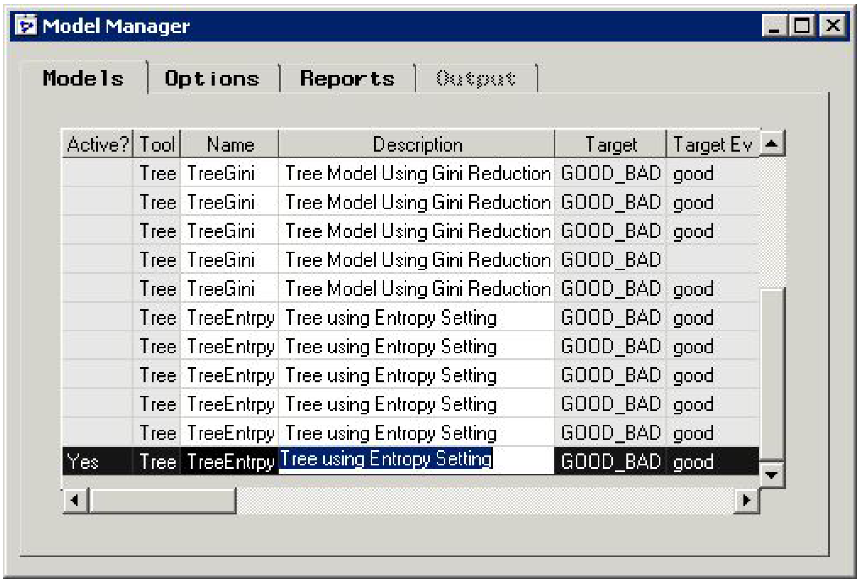
○ Yes ● No

Level	Count	Prior
GOOD	274	0.685
BAD	126	0.315

Model Manager

All modeling nodes in SAS Enterprise Miner 4.x contain a utility called Model Manager. The Model Manager utility acts as a common framework that can be used to compare models and predictions, create assessment charts, and configure model reports.

In modeling nodes that aggregate models through stratification or looping, the SAS Enterprise Miner 4.x Model Manager enables you to view individual model results and fit statistics.



Active?	Tool	Name	Description	Target	Target Ev
	Tree	TreeGini	Tree Model Using Gini Reduction	GOOD_BAD	good
	Tree	TreeGini	Tree Model Using Gini Reduction	GOOD_BAD	good
	Tree	TreeGini	Tree Model Using Gini Reduction	GOOD_BAD	good
	Tree	TreeGini	Tree Model Using Gini Reduction	GOOD_BAD	
	Tree	TreeGini	Tree Model Using Gini Reduction	GOOD_BAD	good
	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good
	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good
	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good
	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good
	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good
Yes	Tree	TreeEntrpy	Tree using Entropy Setting	GOOD_BAD	good

SAS Enterprise Miner 5.3 and later does not contain a Model Manager utility. If you convert this SAS Enterprise Miner 4.x process flow diagram that contains a modeling node that aggregates models as shown in this example, the modeling node in the converted SAS Enterprise Miner 5.3 and later process flow diagram retains only the model results from the last active model.

This example shows the Model Manager window for a Tree node that is part of a group processing process flow diagram. When this process flow diagram is converted for use with SAS Enterprise Miner 5.3 and later, only the results from the last active model are transferred as SAS Enterprise Miner 4.x results.

Part 6

Analytics

Chapter 14
SAS Enterprise Miner Analytics

SAS Enterprise Miner Analytics

Introduction to SEMMA

SAS Institute defines data mining as the process of Sampling, Exploring, Modifying, Modeling, and Assessing (SEMMA) large amounts of data to uncover previously unknown patterns, which can be used as a business advantage. The data mining process is applicable across a variety of industries and provides methodologies for such diverse business problems as fraud detection, householding, customer retention and attrition, database marketing, market segmentation, risk analysis, affinity analysis, customer satisfaction, bankruptcy prediction, and portfolio analysis.

SAS Enterprise Miner software is an integrated product that provides an end-to-end business solution for data mining. A graphical user interface (GUI) provides a user-friendly front end to the SEMMA data mining process:

- **Sample** the data by extracting and preparing a sample of data for model building using one or more data tables. Sampling includes operations that define or subset rows of data. The samples should be large enough to efficiently contain the significant information.
- **Explore** the data by searching for anticipated relationships, unanticipated trends, and anomalies in order to gain understanding and ideas.
- **Modify** the data by creating, selecting, and transforming the variables to focus the model selection process on the most valuable attributes.
- **Model** the data by using the analytical techniques to search for a combination of the data that reliably predicts a desired outcome.
- **Assess** the data by evaluating the usefulness and reliability of the findings from the data mining process.

Overview of the SAS Enterprise Miner Node Tools

Sample Nodes

- The **Append** node enables you to append data sets that are exported by two or more paths in a single SAS Enterprise Miner process flow diagram. The **Append** node can append data according to the data role, such as joining training data to training data, transaction data to transaction data, score data to score data, and so on. The **Append** node can append data that was previously partitioned in train, test, and validate roles into one large training data set.

- The **Data Partition** node enables you to partition data sets into training, test, and validation data sets. The training data set is used for preliminary model fitting. The validation data set is used to monitor and tune the model weights during estimation and is also used for model assessment. The test data set is an additional hold-out data set that you can use for model assessment. This node uses simple random sampling, stratified random sampling, or user-defined partitions to create partitioned data sets.

- The **File Import** node enables you to import data that is stored in external formats into a data source that SAS Enterprise Miner can interpret. The **File Import** node currently can process CSV flat files, JMP tables, Microsoft Excel and Lotus spreadsheet files, Microsoft Access database tables, and DBF, DLM, and DTA files.

- The **Filter** node enables you to apply a filter to the training data set in order to exclude outliers or other observations that you do not want to include in your data mining analysis. Outliers can greatly affect modeling results and, subsequently, the accuracy and reliability of trained models.

- The **Input Data** node enables you to access SAS data sets and other types of data. The **Input Data** node represents the introduction of a predefined metadata into a Diagram Workspace for processing. You can view metadata information about your source data in the **Input Data** node, such as initial values for measurement levels and model roles of each variable.

- The **Merge** node enables you to merge observations from two or more data sets into a single observation in a new data set. The **Merge** node supports both one-to-one and match merging. In addition, you have the option to rename certain variables (for example, predicted values and posterior probabilities) depending on the settings of the node.

- The **Sample** node enables you to take random, stratified random, and cluster samples of data sets. Sampling is recommended for extremely large databases because it can significantly decrease model training time. If the sample is sufficiently representative, then relationships found in the sample can be expected to generalize to the complete data set.

- The **Time Series** node enables you to understand trends and seasonal variations in the transaction data that you collect from your customers and suppliers over the time, by converting transactional data into time series data. Transactional data is time-stamped data that is collected over time at no particular frequency. By condensing the information into a time series, you can discover trends and seasonal variations in customer and supplier habits that might not be visible in transactional data.

Explore Nodes

- The **Association** node enables you to identify association relationships within the data. For example, if a customer buys a loaf of bread, how likely is the customer to also buy a gallon of milk? The node also enables you to perform sequence discovery if a sequence variable is present in the data set.

- The **Cluster** node enables you to segment your data by grouping observations that are statistically similar. Observations that are similar tend to be in the same cluster, and observations that are different tend to be in different clusters. The cluster identifier for each observation can be passed to other tools for use as an input, ID, or target variable. It can also be used as a group variable that enables automatic construction of separate models for each group.
- The **DMDB** node creates a data mining database that provides summary statistics and factor-level information for class and interval variables in the imported data set. The DMDB is a metadata catalog used to store valuable counts and statistics for model building.
- The **Graph Explore** node is an advanced visualization tool that enables you to explore large volumes of data graphically to uncover patterns and trends and to reveal extreme values in the database. For example, you can analyze univariate distributions, investigate multivariate distributions, and create scatter and box plots and constellation and 3-D charts. Graph Explore plots are fully interactive and are dynamically linked to highlight data selections in multiple views.
- The **Link Analysis** node transforms unstructured transactional or relational data into a model that can be graphed. Such models can be used to discover fraud detection, criminal network conspiracies, telephone traffic patterns, website structure and usage, database visualization, and social network analysis. Also, the node can be used to recommend new products to existing customers.
- The **Market Basket** node performs association rule mining over transaction data in conjunction with item taxonomy. This node is useful in retail marketing scenarios that involve tens of thousands of distinct items, where the items are grouped into subcategories, categories, departments, and so on. This is called item taxonomy. The **Market Basket** node uses the taxonomy data and generates rules at multiple levels in the taxonomy.
- The **MultiPlot** node is a visualization tool that enables you to explore larger volumes of data graphically. The **MultPlot** node automatically creates bar charts and scatter plots for the input and target variables without making several menu or window item selections. The code created by this node can be used to create graphs in a batch environment.
- The **Path Analysis** node enables you to analyze Web log data to determine the paths that visitors take as they navigate through a website. You can also use the node to perform sequence analysis.
- The **SOM/Kohonen** node enables you to perform unsupervised learning by using Kohonen vector quantization (VQ), Kohonen self-organizing maps (SOMs), or batch SOMs with Nadaraya-Watson or local-linear smoothing. Kohonen VQ is a clustering method, whereas SOMs are primarily dimension-reduction methods.
- The **StatExplore** node is a multipurpose node that you use to examine variable distributions and statistics in your data sets. Use the **StatExplore** node to compute standard univariate statistics, to compute standard bivariate statistics by class target and class segment, and to compute correlation statistics for interval variables by interval input and target. You can also use the **StatExplore** node to reject variables based on target correlation.
- The **Variable Clustering** node is a useful tool for selecting variables or cluster components for analysis. Variable clustering removes collinearity, decreases variable redundancy, and helps reveal the underlying structure of the input variables in a data set. Large numbers of variables can complicate the task of determining the

relationships that might exist between the independent variables and the target variable in a model. Models that are built with too many redundant variables can destabilize parameter estimates, confound variable interpretation, and increase the computing time that is required to run the model. Variable clustering can reduce the number of variables that are required to build reliable predictive or segmentation models.

- The **Variable Selection** node enables you to evaluate the importance of input variables in predicting or classifying the target variable. The node uses either an R-square or a Chi-square selection (tree based) criterion. The R-square criterion removes variables that have large percentages of missing values, and remove class variables that are based on the number of unique values. The variables that are not related to the target are set to a status of rejected. Although rejected variables are passed to subsequent tools in the process flow diagram, these variables are not used as model inputs by modeling nodes such as the Neural Network and Decision Tree tools.

Modify Nodes

- The **Drop** node enables you to drop selected variables from your scored SAS Enterprise Miner data sets. You can drop variables that have roles of Assess, Classification, Frequency, Hidden, Input, Rejected, Residual, and Target from your scored data sets. Use the **Drop** node to trim the size of data sets and metadata during tree analysis.

- The **Impute** node enables you to replace missing values for interval variables with the mean, median, midrange, mid-minimum spacing, distribution-based replacement, or use a replacement M-estimator such as Tukey's biweight, Hubers, Andrew's Wave, or by using a tree-based imputation method. Missing values for class variables can be replaced with the most frequently occurring value, distribution-based replacement, tree-based imputation, or a constant.

- The **Interactive Binning** node is used to model nonlinear functions of multiple modes of continuous distributions. The interactive tool computes initial bins by quintiles, and then you can split and combine the initial quintile-based bins into custom final bins.

- The **Principal Components** node enables you to perform a principal components analysis for data interpretation and dimension reduction. The node generates principal components that are uncorrelated linear combinations of the original input variables and that depend on the covariance matrix or correlation matrix of the input variables. In data mining, principal components are usually used as the new set of input variables for subsequent analysis by modeling nodes.

- The **Replacement** node enables you to replace selected values for class variables. The **Replacement** node summarizes all values of class variables and provides you with an editable variables list. You can also select a replacement value for future unknown values.

- The **Rules Builder** node enables you to create ad hoc sets of rules for your data that result in user-definable outcomes. For example, you might use the **Rules Builder** node to define outcomes named Deny and Review based on rules such as the following:

```
IF P_Default_Yes > 0.4 then do
    EM_OUTCOME-"Deny";
    IF AGE > 60 then
      EM_OUTCOME="Review";
END;
```

- The **Transform Variables** node enables you to create new variables that are transformations of existing variables in your data. Transformations can be used to stabilize variances, remove nonlinearity, improve additivity, and correct nonnormality in variables. You can also use the **Transform Variables** node to transform class variables and to create interaction variables.

Model Nodes

- The **AutoNeural** node can be used to automatically configure a neural network. The **AutoNeural** node implements a search algorithm to incrementally select activation functions for a variety of multilayer networks.
- The **Decision Tree** node enables you to fit decision tree models to your data. The implementation includes features found in a variety of popular decision tree algorithms (for example, CHAID, CART, and C4.5). The node supports both automatic and interactive training. When you run the Decision Tree node in automatic mode, it automatically ranks the input variables based on the strength of their contribution to the tree. This ranking can be used to select variables for use in subsequent modeling. You can override any automatic step with the option to define a splitting rule and prune explicit tools or subtrees. Interactive training enables you to explore and evaluate data splits as you develop them.
- The **Dmine Regression** node enables you to compute a forward stepwise least squares regression model. In each step, the independent variable that contributes maximally to the model R-square value is selected. The tool can also automatically bin continuous terms.
- The **DMNeural** node is another modeling node that you can use to fit an additive nonlinear model. The additive nonlinear model uses bucketed principal components as inputs to predict a binary or an interval target variable with automatic selection of an activation function.
- The **Ensemble** node enables you to create new models by combining the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple predecessor models.
- The **Gradient Boosting** node uses tree boosting to create a series of decision trees that together form a single predictive model. Each tree in the series is fit to the residual of the prediction from the earlier trees in the series. The residual is defined in terms of the derivative of a loss function. For squared error loss with an interval target, the residual is simply the target value minus the predicted value. Boosting is defined for binary, nominal, and interval targets.
- The **LARS** node enables you to use Least Angle Regression algorithms to perform variable selection and model fitting tasks. The **LARs** node can produce models that range from simple intercept models to complex multivariate models that have many inputs. When using the **LARs** node to perform model fitting, The **LARs** node uses criteria from either least angle regression or the LASSO regression to choose the optimal model.
- The **MBR** (Memory-Based Reasoning) node enables you to identify similar cases and to apply information that is obtained from these cases to a new record. The **MBR** node uses k-nearest neighbor algorithms to categorize or predict observations.
- The **Model Import** node enables you to import models into the SAS Enterprise Miner environment that were not created by SAS Enterprise Miner. Models that were created by using SAS PROC LOGISTIC, for example, can now be run, assessed, and modified in SAS Enterprise Miner.

- The **Neural Network** node enables you to construct, train, and validate multilayer feedforward neural networks. Users can select from several predefined architectures or manually select input, hidden, and target layer functions and options.
- The **Partial Least Squares** node is a tool for modeling continuous and binary targets based on SAS/STAT PROC PLS. The **Partial Least Squares** node produces DATA step score code and standard predictive model assessment results.
- The **Regression** node enables you to fit both linear and logistic regression models to your data. You can use continuous, ordinal, and binary target variables. You can use both continuous and discrete variables as inputs. The node supports the stepwise, forward, and backward selection methods. A point-and-click interaction builder enables you to create higher-order modeling terms.
- The **Rule Induction** node enables you to improve the classification of rare events in your modeling data. The **Rule Induction** node creates a Rule Induction model that uses split techniques to remove the largest pure split node from the data. Rule Induction also creates binary models for each level of a target variable and ranks the levels from the most rare event to the most common. After all levels of the target variable are modeled, the score code is combined into a SAS DATA step.
- The **SVM** node uses supervised machine learning to perform binary classification problems, including polynomial, radial basis function and sigmoid nonlinear kernels. The standard SVM problem solves binary classification problems by constructing a set of hyperplanes that maximize the margin between two classes. The **SVM** node does not support multi-class problems or support vector regression.
- The **TwoStage** node enables you to compute a two-stage model for predicting a class and an interval target variables at the same time. The interval target variable is usually a value that is associated with a level of the class target.

Assess Nodes

- The **Cutoff** node provides tabular and graphical information to help you determine the best cutoff point or points for decision making models that have binary target variables.
- The **Decisions** node enables you to define target profiles to produce optimal decisions. You can define fixed and variable costs, prior probabilities, and profit or loss matrices. These values are used in model selection steps.
- The **Model Comparison** node provides a common framework for comparing models and predictions from any of the modeling tools (such as Regression, Decision Tree, and Neural Network tools). The comparison is based on standard model fits statistics as well as potential expected and actual profits or losses that would result from implementing the model. The node produces the following charts that help describe the usefulness of the model: lift, profit, return on investment, receiver operating curves, diagnostic charts, and threshold-based charts.
- The **Score** node enables you to manage, edit, export, and execute scoring code that is generated from a trained model. Scoring is the generation of predicted values for a data set that cannot contain a target variable. The **Score** node generates and manages scoring formulas in the form of a single SAS DATA step, which can be used in most SAS environments even without the presence of SAS Enterprise Miner.
- The **Segment Profile** node enables you to assess and explore segmented data sets. Segmented data is created from data BY-values, clustering, or applied business rules. The **Segment Profile** node facilitates data exploration to identify factors that differentiate individual segments from the population, and to compare the distribution of key factors between individual segments and the population. The **Segment Profile** node outputs a Profile plot of variable distributions across segments

and population, a Segment Size pie chart, a Variable Worth plot that ranks factor importance within each segment, and summary statistics for the segmentation results. The **Segment Profile** node does not generate score code or modify metadata.

Utility Nodes

- The **Control Point** node establishes a nonfunctional connection point to clarify and simplify process flow diagrams. For example, suppose three Input Data nodes are to be connected to three modeling nodes. If no Control Point node is used, then nine connections are required to connect all of the Input Data nodes to all of the modeling nodes. However, if a Control Point node is used, only six connections are required.
- The **End Groups** node terminates a group processing segment in the process flow diagram. If the group processing function is stratified, bagging, or boosting, the **Ends Groups** node will function as a model node and present the final aggregated model. (Ensemble nodes are not required as in SAS Enterprise Miner 4.3.) Nodes that follow the **Ends Groups** node continue data mining processes normally.
- The **ExtDemo** node illustrates the various UI elements that can be used by SAS Enterprise Miner extension nodes.
- The **Metadata** node enables you to modify the columns metadata information at some point in your process flow diagram. You can modify attributes such as roles, measurement levels, and order.
- The **Reporter** node tool uses SAS Output Delivery System (ODS) capabilities to create a single document for the given analysis in PDF or RTF format. The document includes important SAS Enterprise Miner results, such as variable selection, model diagnostic tables, and model results plots. The document can be viewed and saved directly and will be included in SAS Enterprise Miner report package files.
- The **SAS Code** node tool enables you to incorporate SAS code into process flows that you develop using SAS Enterprise Miner. The **SAS Code** node extends the functionality of SAS Enterprise Miner by making other SAS System procedures available in your data mining analysis. You can also write a SAS DATA step to create customized scoring code, to conditionally process data, and to concatenate or to merge existing data sets.
- The **Score Code Export** node tool enables you to extract score code and score metadata to an external folder. The **Score Code Export** node must be preceded by a Score node.
- The **Start Groups** node initiates a group processing segment in the process flow diagram. The **Start Groups** node performs the following types of group processing:

  Stratified group processing that repeats processes for values of a class variable, such as GENDER=M and GENDER=F.

  Bagging, or bootstrap aggregation via repeated resampling.

  Boosting, or boosted bootstrap aggregation, using repeated resampling with residual-based weights.

  Index processing, which repeats processes a fixed number of times. Index processing is normally used with a Sampling node or with user's code for a sample selection.

Credit Scoring Nodes

Note: The SAS Credit Scoring feature is not included with the base version of SAS Enterprise Miner. If your site has not licensed SAS Credit Scoring, the credit scoring node tools will not appear in your SAS Enterprise Miner software.

Analysts can use SAS Enterprise Miner and its credit scoring tools to build scorecard models that assign score points to customer attributes, to classify and select characteristics automatically or interactively using Weights of Evidence and Information Value measures, and to normalize score points to conform with company or industry standards.

- The **Credit Exchange** node enables you to exchange the data that is created in SAS Enterprise Miner with the SAS Credit Risk Solution. The **Credit Exchange** node creates a statistics table, a target table, flow score code, and a required input variables matrix.
- The **Interactive Grouping** node groups variable values into classes that you can use as inputs for predictive modeling, such as building a credit scorecard model. Use the **Interactive Grouping** node to interactively modify classes in order to create optimal binning and grouping.
- The **Reject Inference** node uses the scorecard model to score previously rejected applications. The observations in the rejected data set are classified as inferred "goods" and inferred "bads". The inferred observations are added to the Accepts data set that contains the actual "good" and "bad" records, forming an augmented data set. This augmented data set can then serve as the input data set of a second credit scoring modeling run.
- The **Scorecard** node fits a logistic regression model for the binary target variable. The regression coefficients and scaling parameters are used to calculate scorecard points. Scorecard points are organized in a table that assigns score points to customer attributes. You use the **Scorecard** node to calculate the scaling parameters of a credit scorecard, display the distribution of various score-related statistics, determine an optimal cutoff score, and generate a scored data set.

Applications Nodes

- The **Incremental Response** node directly models the incremental impact of a treatment (such as a marketing action or incentive) and optimizes customer targeting in order to obtain the maximal response or return on investment. You can use incremental response modeling to determine the likelihood that a customer purchases a product, uses a coupon, or to predict the incremental revenue realized during a promotional period.
- The **Ratemaking** node builds a generalized linear model, which is an extension of a traditional linear model. Using class and binned interval input variables, these models enable the population mean to depend on a linear predictor through a nonlinear link function.
- The **Survival** node performs survival analysis on mining customer databases when there are time-dependent outcomes. Some examples of time-dependent outcomes are customer churn, cancellation of all products and services, unprofitable behavior, and server downgrade or extreme inactivity.:

Text Mining Nodes

Note: The SAS Text Miner feature is not included with the base version of SAS Enterprise Miner. If your site has not licensed SAS Text Miner, the text mining node tools will not appear in your SAS Enterprise Miner software.

SAS Text Miner provides tools for discovering and extracting information from a widely varied collection of text documents. The tools uncover verbal patterns and concepts that are embedded within the document collection. SAS Text Miner also enables you to combine quantitative variables with unstructured text in your mining process. SAS Text

Miner nodes run within SAS Enterprise Miner, giving users the ability to combine text mining capabilities with the data mining capabilities that SAS Enterprise Miner offers.

- The **Text Cluster** node enables you to perform a cluster analysis on a document collection. The **Text Cluster** node must be preceded by the Text Parsing and Text Filter node. If you desire, the **Text Cluster** node can also be preceded by the **Text Topic** node.
- The **Text Filter** node enables you to reduce the number of parsed terms by dropping terms with little or no useful information. A filtered and compacted term set allows for faster computations and only meaningful terms to be used.
- The **Text Import** node extracts the text from documents contained in a directory and creates a set of results. You can also use the **Text Import** node to crawl the web beginning at a specific URL and retrieve the web pages that it finds.
- The **Text Miner** node enables you to discover and use the information that exists in a document collection as a whole. It can process volumes of textual data such as e-mail messages, news articles, web pages, research papers, and surveys, even if they are stored in different languages or data formats. The **Text Miner** node behaves like most nodes in SAS Enterprise Miner except that the **Text Miner** node does not generate portable score code.
- The **Text Parsing** node enables you to parse a collection of documents and quantify the information about the terms in that collection. The **Text Parsing** node enables you to ignore different parts of speech, classify multiple terms as synonyms, and adjust the language used for parsing.
- The **Text Topic** node is used to create topics from a document collection. For each topic that is collected, a variable is added to the training table that the node exports. Topics are selected after computing the singular value decomposition of the term-by-document frequency matrix. This is the most memory-intensive task of the text mining process and can require a simple random sample of the documents in order to run successfully. The **Text Topic** node must be preceded by a Text Parsing node. A Text Filter node can also precede the **Text Topic** node.

Time Series Nodes

- The **Time Series Data Preparation** node provides time series data cleaning, summarization, transformation, transposition, and so on.
- The **Time Series Exponential Smoothing** node generates forecasts by using exponential smoothing models with optimized smoothing weights for many time series models.
- The **Time Series Similarity** node computes similarity measures associated with time-stamped data.

Node Usage Rules

Here are some general rules that govern the placement of nodes in a process flow diagram:

- The **Input Data** node cannot be preceded by any other nodes.
- The **Input Data** node and the **SAS Code** node are the only tools that do not require predecessor nodes.
- The **Model Comparison** node must be preceded by one or more modeling nodes.

- The **Score** node must be preceded by a node that produces score code. For example, the Transform Variables, Impute, Cluster, Regression, Neural Network, AutoNeural, Decision Tree, and Variable Selection tools generate score code.

Chapter 15

Cubic Clustering Criterion

Cubic Clustering Criterion

Abstract

The cubic clustering criterion (CCC) can be used to estimate the number of clusters using Ward's minimum variance method, k -means, or other methods based on minimizing the within-cluster sum of squares. The performance of the CCC is evaluated by Monte Carlo methods.

Introduction

The most widely used optimization criterion for disjoint clusters of observations is known as the within-cluster sum of squares, WSS, error sum of squares, ESS, residual sum of squares, least squares, (minimum) squared error, (minimum) variance, (sum of) squared (Euclidean) distances, trace(W), (proportion of) variance accounted for, or R^2 (see, for example, Anderberg 1973; Duran and Odell 1974; Everitt 1980). The following notation is used herein to define this criterion:

n
number of observations

n_k
number of observations in the k^{th} cluster

p
number of variables

q
number of clusters

X
: n by p data matrix

$\overline{X}$
: q by p matrix of cluster means

Z
: cluster indicator matrix with element $Z_{ik} = 1$ if the i^{th} observation belongs to the k^{th} cluster, 0 otherwise.

Assume that without loss of generality each variable has mean zero. Note that Z'Z is a diagonal matrix containing the n_{ks} and that

$$\overline{X} = (Z'Z)^{-1} Z'X.$$

The total-sample sum-of-squares and cross products (SSCP) matrix is

T = X’X.

The between-cluster SCCP matrix is

$$B = \overline{X}' Z'Z\overline{X}.$$

The within-cluster SSCP matrix is

$$\begin{aligned} W &= (X - Z\overline{X})'(X - Z\overline{X}) \\ &= X'X - \overline{X}'Z'Z\overline{X} \\ &= T - B. \end{aligned}$$

The within-cluster sum of squares pooled over variables is thus trace(W). By changing the order of the summations, it can also be shown that trace(W) equals the sum of squared Euclidean distances from each observation to its cluster mean.

Since T is constant for a given sample, minimizing trace(W) is equivalent to maximizing

$$R^2 = 1 - \frac{\text{trace}(W)}{\text{trace}(T)},$$

which has the usual interpretation of the proportion of variance accounted for by the clusters. R^2 can also be obtained by multiple regression if the columns of x are stacked on top of each other to form an np by 1 vector, and this vector is regressed on the Kronecker product of z with an order p identity matrix.

Many algorithms have been proposed for maximizing [untitled graphic] or equivalent criteria (for example, Ward 1963; Edwards and Cavalli-Sforza 1965; MacQueen 1967; Gordon and Henderson 1977). This report concentrates on Ward's method as implemented in the CLUSTER procedure. Similar results should be obtained with other algorithms, such as the k-means method provided by FASTCLUS.

The most difficult problem in cluster analysis is how to determine the number of clusters. If you are using a goodness-of-fit criterion such as R^2, you would like to know the sampling distribution of the criterion to enable tests of cluster significance. Ordinary significance tests, such as analysis of variance F- tests, are not valid for testing differences between clusters. Since clustering methods attempt to maximize the separation between clusters, the assumptions of the usual significance tests, parametric or nonparametric, are drastically violated. For example, 25 samples of 100 observations from a single univariate normal distribution were each divided into two clusters by FASTCLUS. The median absolute t-statistic testing the difference between the cluster means was 13.7, with a range from 10.9 to 15.7. For a nominal significance level of 0.0001 under the usual, but invalid, assumptions, the critical value is 3.4, yielding an actual type 1 error rate close to 1.

The first step in devising a valid significance test for clusters is to specify the null and alternative hypotheses. For clustering methods based on distance matrices, a popular null hypothesis is that all permutations of the values in the distance matrix are equally likely (Ling 1973; Hubert 1974). Using this null hypothesis, you can do a permutation test or a rank test. The trouble with permutation hypothesis is that, with any real data, the null hypothesis is totally implausible even if the data does not contain clusters. Rejecting the null hypothesis does not provide any useful information (Huber and Baker 1977).

Another common null hypothesis is that the data are a random sample from a multivariate normal distribution (Wolfe 1970, 1978; Lee 1979). The multivariate normal null hypothesis is better than the permutation null hypothesis, but it is not satisfactory because there is typically a high probability of rejection if the data is sampled from a distribution with lower kurtosis than a normal distribution, such as a uniform distribution. The tables in Englemann and Hartigan (1969), for example, generally lead to rejection of the null hypothesis when the data is sampled from a uniform distribution.

Hartigan (1978) and Arnold (1979) discuss both normal and uniform null hypotheses, and the uniform null hypothesis seems preferable for most practical purposes. Hartigan (1978) has obtained asymptotic distributions for the within-cluster sum of squares criterion in one dimension for normal and uniform distributions. Hartigan's results require very large sample sizes, perhaps 100 times the number of clusters, and are, therefore, of limited practical use.

This report describes a rough approximation to the distribution of the R^2 criterion under the null hypothesis that the data have been sampled from a uniform distribution on a hyperbox (a p-dimensional right parallelepiped). This approximation is helpful in determining the best number of clusters for both univariate and multivariate data and with sample sizes down to 20 observations. The approximation to the expected value of R^2 is based on the assumption that the clusters are shaped approximately like hypercubes. In more than one dimension, this approximation tends to be conservative for a small number of clusters and slightly liberal for a very large number of clusters (about 25 or more in two dimensions). The cubic clustering criterion (CCC) is obtained by comparing the observed R^2 to the approximate expected R^2 using an approximate variance-stabilizing transformation. Positive values of the CCC mean that the obtained R^2 is greater than would be expected if sampling from a uniform distribution and therefore indicate the possible presence of clusters. Treating the CCC as a standard normal test statistic provides a crude test of the hypotheses:

H_0: the data has been sampled from a uniform distribution on a hyperbox.

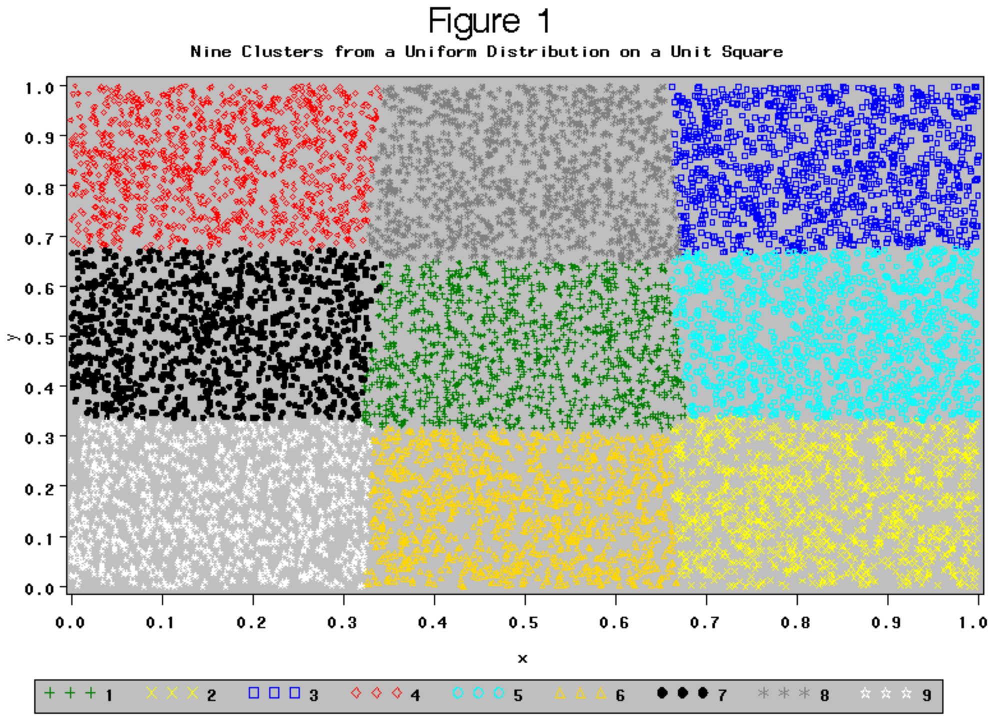
H_a: the data has been sampled from a mixture of spherical multivariate normal distributions with equal variances and equal sampling probabilities.

Under this alternative hypothesis, R^2 is equivalent to the maximum likelihood criterion (Scott and Symons 1971).

Computation of the Cubic Clustering Criterion

The CCC is based on the assumption that clusters obtained from a uniform distribution on a hyperbox are hypercubes of the same size. The hypercube assumption is obviously false in most cases, but is generally conservative unless the number of clusters is very large in two or more dimensions. Wong (1982) has shown that, for many clusters in two dimensions from a uniform sample, the cluster shape tends to be hexagonal. Figure 1 illustrates a case in which the hypercube (or square, since there are only two dimensions) assumption is correct. A sample of 10,000 points from a uniform distribution on the unit square was divided into nine clusters by FASTCLUS. Each cluster is nearly square with edge length 1/3. [Graph of nine clusters from a uniform distribution on a unit square]

Figure 15.1 *Nine Clusters from a Uniform Distribution on a Unit Square*



A first approximation to the value of R^2 for a population uniformly distributed on a hyperbox can be obtained. Assume that the edges of the hyperbox are aligned with the coordinate axes. Let s_j be the edge length of the hyperbox along the j^{th} dimension. Assume further that the s_js are in decreasing order. The volume of the hyperbox is

$$v = \prod_{j=1}^{p} s_j .$$

If the hyperbox is divided into q hypercubes with edge length c, then the volume of the hyperbox equals the total volume of the hypercubes; hence

$$c = \left(\frac{v}{q}\right)^{\frac{1}{p}}.$$

Let

$$u_j = \frac{s_j}{c}$$

be the number of hypercubes along the jth dimension of the hyperbox. The total-sample variance along the jth dimension is proportional to s_j^2, while the within-cluster variance along the jth dimension is proportional to c^2. Thus

$$\begin{aligned} R^2 &\doteq 1 - \frac{\sum_{j=1}^{p} c^2}{\sum_{j=1}^{p} s_j^2} \\ &= 1 - \frac{p}{\sum_{j=1}^{p} u_j^2}. \end{aligned}$$

In Figure 1 above, for example, s1 = s2 = 1, c = 1/3, and u1 = u2 = 3, so the population R^2 is

$$\begin{aligned} R^2 &\doteq 1 - \frac{2}{(3^2+3^2)} \\ &= 0.8888\ldots, \end{aligned}$$

where the sample R^2 is 0.888967+.

The above approximation fails badly if the dimensionality of the between-cluster variation, say p*, is less than p. Obviously, p* must be less than the number of clusters, q. Also, $u_j < 1$ implies p* < j. For a better approximation, assume that the clusters are hyperboxes with edge length c in the first p* dimensions, and edge length s_j in the remaining dimensions. Let

$$v^* = \prod_{j=1}^{p^*} s_j,$$

$$c = \left(\frac{v^*}{q} \right)^{\frac{1}{p^*}},$$

$$u_j = \frac{s_j}{c},$$

where p* is chosen to be the largest integer less than q such that u_{p*} is not less than one. Then we have the following approximation to the population R^2]:

$$R^2 \doteq 1 - \frac{p^* + \sum_{j=p^*+1}^{p} u_j^2}{\sum_{j=1}^{p} u_j^2}.$$

In small samples from a uniform distribution on a hyperbox, the sample R^2s tend to exceed the population R^2 due to the phenomenon widely known as "capitalization on chance." Extensive simulations led to the following heuristic small-sample approximation for the expected value of R^2:

$$E\left(R^2\right) \doteq 1 - \left[\frac{\sum_{j=1}^{p^*} \frac{1}{n+u_j} + \sum_{j=p^*+1}^{p} \frac{u_j^2}{n+u_j}}{\sum_{j=1}^{p} u_j^2} \right] \left[\frac{(n-q)^2}{n} \right] \left[1 + \frac{4}{n} \right].$$

Given a sample X, let s_j be the square root of the jth eigenvalue of T/(n-1), so that under the null hypothesis, the length of the hyperbox in the jth dimension is proportional to the standard deviation of the jth principal component of the data. The CCC is computed from the observed R^2 as

$$CCC = \ln \left[\frac{1 - E\left(R^2\right)}{1 - R^2} \right] \frac{\sqrt{\frac{np^*}{2}}}{\left(0.001 + E\left(R^2\right)\right)^{1.2}}.$$

The above formula was derived empirically in an attempt to stabilize the variance across different numbers of observations, variables, and clusters.

Empirical Examination of the Performance of the CCC

A Monte Carlo study of the null distribution of the CCC was performed by clustering samples from uniform distributions on hypercubes by Ward's minimum variance method as implemented in the CLUSTER procedure. The design involved two factors:

- The number of observations was 20, 40, 80, 160, 320, or 640.
- The number of variables was 1, 2, 4, 8, or 16.

For each combination of these factors, 50 samples were generated from a uniform distribution on a hypercube. Each sample was clustered and E(R^2) and the CCC were computed with the number of clusters ranging from one to one-tenth the number of observations. Figure 2 is a plot of the observed average R^2 against the theoretical approximate expected [untitled graphic]. Each combination of the number of observations, number of variables, and number of clusters is represented by a point giving the mean of 50 values of the observed R^2 and the approximate E(R^2). Points for which the observed R^2 exceeds E(R^2) are labeled "L" for liberal, while the remaining points are labeled "C" for conservative. If both liberal and conservative points fell at a given plotting position, "L" was printed. The vast majority of the points are mildly conservative.

Figure 15.2 *Plot of Observed Average R–Squared Against Theoretical Approximate Expected R–Squared for Uniform Hypercubical Distributions*

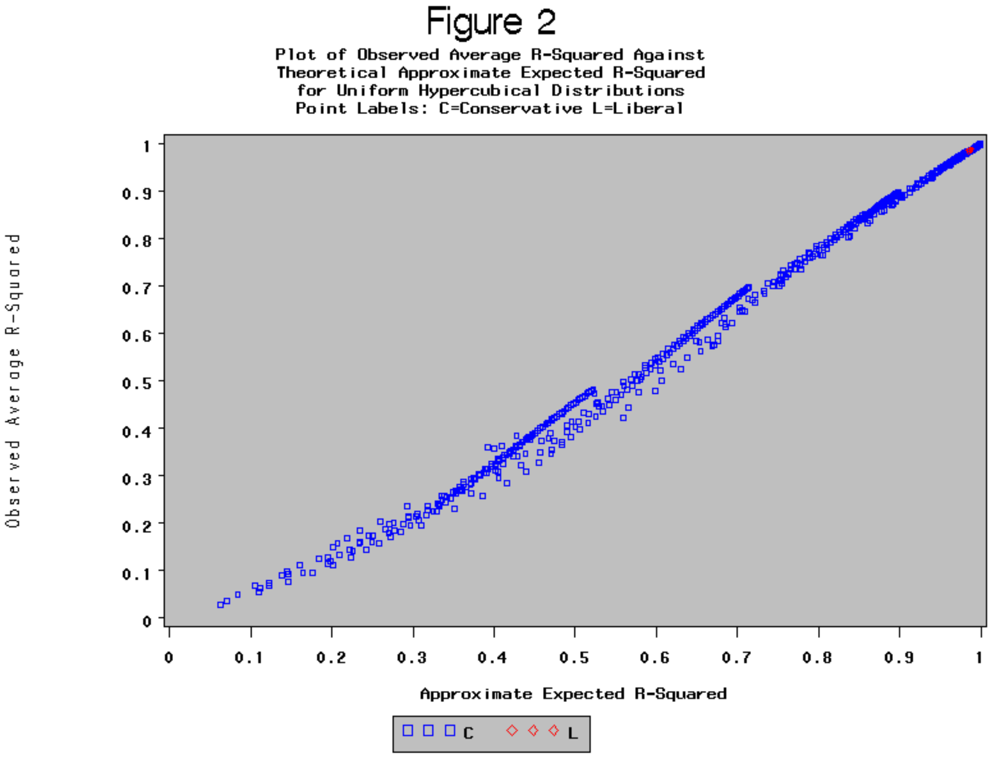


Table 1 gives the mean and standard deviation of the CCC for each combination of number of observations, clusters, and variables. Nearly all the means are negative, showing that the CCC is generally conservative. The only exceptions are for 56 or more clusters with 640 observations and 2 variables. For a given number of observations and variables, the mean CCC reaches a minimum when the number of clusters is close to the number of variables plus one. In that case, the assumption of hypercubical clusters is badly violated. The CCC becomes extremely conservative for 16 variables, especially with a large number of observations. The standard deviations are generally close to 1.0, but are larger for a small number of clusters, especially with the larger sample sizes for two clusters.

Table 15.1 *Table 1: Mean and Standard Deviation of the CCC for Each Combination of Number of Observations, Clusters, and Variables.*

N	Clusters	Variables									
		1		2		4		8		16	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
20	2	-.06	1.2	-0.7	0.6	-1.2	0.4	-1.5	0.3	-1.8	0.2
40	2	-0.7	1.4	-1.0	0.7	-1.5	0.5	-2.0	0.4	-2.5	0.4
40	3	-0.5	1.1	-1.6	1.1	-2.2	0.6	-2.6	0.4	-3.3	0.4
40	4	-0.5	1.1	-1.0	1.1	-2.5	0.7	-3.0	0.4	-3.8	0.3
80	2	-0.7	1.6	-1.3	0.9	-2.3	0.8	-3.2	0.7	-3.8	0.5
80	3	-0.8	1.2	-2.8	1.2	-3.3	0.8	-4.1	0.7	-4.8	0.5
80	4	-0.7	1.2	-1.2	1.4	-4.0	0.9	-4.6	0.8	-5.4	0.5
80	5	-0.5	1.1	-1.1	1.2	-4.4	1.1	-5.0	0.8	-6.0	0.5
80	6	-0.6	1.1	-1.0	1.2	-3.6	1.1	-5.2	0.9	-6.4	0.5
80	7	-0.4	1.3	-0.8	1.1	-2.9	1.1	-5.3	1.0	-6.7	0.5
80	8	-0.4	1.1	-0.6	1.0	-2.4	1.1	-5.3	1.0	-6.9	0.6
160	2	-1.2	2.2	-1.9	1.2	-3.3	1.0	-5.1	0.7	-6.2	0.7
160	3	-1.6	1.4	-4.9	1.3	-5.0	0.9	-6.6	0.8	-7.8	0.7
160	4	-1.4	1.3	-2.7	1.6	-6.3	0.8	-7.5	0.8	-8.8	0.8
160	5	-1.2	1.2	-2.4	1.3	-7.4	0.9	-8.2	0.9	-9.5	0.8
160	6	-1.2	1.1	-2.0	1.1	-6.5	0.8	-8.8	1.0	-10.0	0.8
160	7	-1.1	1.0	-1.8	0.9	-5.5	1.0	-9.0	1.0	-10.4	0.8
160	8	-1.0	1.0	-1.5	0.9	-4.8	1.0	-9.1	1.1	-10.6	0.8
160	9	-1.1	1.0	-1.2	0.9	-4.1	1.1	-9.0	1.2	-10.8	0.8
160	10	-1.1	1.1	-1.1	0.9	-3.5	1.1	-8.4	1.2	-10.8	0.9
160	11	-1.1	1.0	-1.0	0.9	-3.0	1.1	-7.8	1.2	-10.8	0.9
160	12	-1.0	1.0	-0.8	0.9	-2.6	1.1	-7.2	1.2	-10.8	1.0
160	13	-0.9	1.1	-0.7	1.0	-2.3	1.1	-6.8	1.1	-10.6	1.0
160	14	-0.9	1.2	-0.6	1.0	-2.0	1.1	-6.4	1.1	-10.4	1.0
160	15	-0.9	1.2	-0.5	1.0	-1.7	1.1	-6.0	1.1	-10.0	1.0

N	Clusters	Variables									
		1		2		4		8		16	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
160	16	0.8	1.2	-0.4	1.1	-1.5	1.1	-5.6	1.1	-9.7	1.0
320	2	-3.1	2.4	-2.6	1.4	-4.9	1.6	-8.4	0.8	-10.7	0.9
320	3	-1.9	2.0	-7.3	1.4	-7.7	1.3	-10.7	0.8	-13.4	0.9
320	4	-2.2	1.4	-4.5	1.6	-10.2	1.4	-12.4	0.9	-15.1	0.8
320	5	-2.2	1.5	-4.4	1.5	-12.4	1.3	-13.7	0.9	-16.4	0.8
320	6	-1.7	1.3	-4.1	1.3	-10.9	1.2	-14.8	0.9	-17.3	0.8
320	7	-1.7	1.1	-3.7	1.4	-9.5	1.2	-15.6	10	-18.1	0.7
320	8	-1.5	1.1	-3.2	1.3	-8.3	1.2	-16.2	1.1	-18.6	0.7
320	9	-1.5	1.0	-2.9	1.4	-7.5	1.1	-16.5	1.1	-19.0	0.7
320	10	-1.5	0.9	-2.6	1.3	-6.6	1.2	-15.4	1.0	-19.2	0.7
320	11	-1.4	1.0	-2.4	1.3	-5.9	1.2	-14.5	0.9	-19.4	0.8
320	12	-1.3	1.1	-2.2	1.2	-5.3	1.2	-13.7	0.9	-19.5	0.8
320	13	-1.3	1.1	-2.0	1.1	-4.8	1.1	-13.0	0.9	-19.4	0.8
320	14	-1.3	1.1	-1.8	1.0	-4.3	1.1	-12.3	0.8	-19.3	0.9
320	15	-1.3	1.0	-1.6	1.0	-4.0	1.1	-11.7	0.8	-19.2	0.9
320	16	-1.2	0.9	-1.5	1.0	-3.7	1.1	-11.1	0.8	-18.8	1.0
320	17	-1.2	0.9	-1.4	1.0	-3.5	1.0	-10.6	0.9	-18.1	1.0
320	18	-1.2	1.0	-1.3	1.0	-3.2	1.0	-10.1	0.9	-17.4	1.0
320	19	-1.2	1.0	-1.2	1.0	-3.0	1.0	-9.6	0.9	-16.8	1.0
320	20	-1.2	1.0	-1.1	1.0	-2.8	0.9	-9.1	1.0	-16.2	1.0
320	21	-1.1	1.0	-1.0	1.1	-2.6	0.9	-8.7	1.0	-15.8	1.0
320	22	-1.1	1.0	-0.9	1.1	-2.5	0.8	-8.2	1.0	-15.3	1.0
320	23	-1.0	1.0	-0.8	1.1	-2.3	1.0	-7.8	1.0	-14.8	1.0
320	24	-1.0	1.0	-0.7	1.1	-2.1	1.0	-7.5	1.0	-14.3	1.0
320	25	-1.0	1.0	-0.7	1.1	-1.9	1.0	-7.1	1.0	-13.9	1.0
320	26	-1.0	1.0	-0.6	1.1	-1.8	1.0	-6.8	1.0	-13.5	0.9

		Variables									
		1		2		4		8		16	
N	Clusters	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
320	27	-1.0	0.9	-0.5	1.1	-1.6	1.0	-6.5	1.0	-13.1	0.9
320	28	-1.0	0.9	-0.4	1.1	-1.5	1.0	-6.2	1.0	-12.8	1.0
320	29	-1.0	0.9	-0.4	1.1	-1.4	1.0	-6.0	1.0	-12.4	1.0
320	30	-0.9	0.9	-0.3	1.1	-1.2	1.0	-5.7	1.0	-12.1	1.0
320	31	-0.9	1.0	-0.3	1.1	-1.1	1.0	-5.5	1.0	-11.8	0.9
320	32	-1.0	1.0	-0.2	1.1	-1.0	1.0	-5.2	0.9	-11.5	0.9
640	2	-3.5	3.2	-2.8	1.4	-7.2	1.9	-12.8	1.1	-17.3	0.9
640	3	-3.1	2.0	-10.4	1.7	-11.0	1.6	-16.8	1.2	-21.6	0.9
640	4	-3.2	1.9	-5.4	2.4	-15.2	1.5	-19.7	1.4	-24.6	0.9
640	5	-2.8	1.7	-5.7	1.8	-19.2	1.4	-22.1	1.3	-26.9	0.9
640	6	-2.9	1.4	-5.6	1.5	-17.3	1.2	-24.1	1.3	-28.7	0.9
640	7	-2.5	1.4	-5.2	1.3	-15.6	1.1	-25.9	1.3	-30.2	1.0
640	8	-2.1	1.2	-4.9	1.2	-14.0	1.1	-27.4	1.2	-31.4	1.0
640	9	-1.9	1.2	-4.6	1.3	-12.7	1.3	-28.6	1.2	-33.2	1.0
640	10	-2.0	1.0	-4.3	1.4	-11.5	1.4	-27.2	1.2	-33.2	1.0
640	11	-2.1	0.9	-4.1	1.2	-10.5	1.4	-26.0	1.2	-33.9	1.0
640	12	-2.2	0.9	-3.9	1.1	-9.6	1.4	-24.8	1.1	-34.5	1.1
640	13	-2.2	1.0	-3.8	1.0	-8.8	1.4	-23.8	1.1	-34.9	1.1
640	14	-2.1	1.0	-3.6	0.9	-8.2	1.3	-22.9	1.1	-35.1	1.1
640	15	-2.0	1.1	-3.4	0.9	-7.7	1.3	-22.0	1.1	-35.3	1.2
640	16	-1.8	1.1	-3.2	0.9	-7.3	1.3	-21.2	1.1	-35.3	1.2
640	17	-1.8	1.1	-3.1	0.9	-7.0	1.3	-20.3	1.1	-35.3	1.2
640	18	-1.8	1.1	-2.9	0.9	-6.7	1.3	-19.6	1.4	-34.1	1.1
640	19	-1.8	1.0	-2.8	0.9	-6.5	1.2	-18.9	1.1	-33.0	1.1
640	20	-1.8	1.0	-2.7	0.9	-6.2	1.2	-18.2	1.1	-32.0	1.1
640	21	-1.8	1.0	-2.6	0.9	-6.0	1.2	-17.5	1.2	-31.0	1.1

N	Clusters	Variables									
		1		2		4		8		16	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	22	-1.8	1.0	-2.5	0.9	-5.8	1.1	-16.9	1.2	-30.1	1.1
640	23	-1.8	0.9	-2.4	0.9	-5.6	1.1	-16.3	1.1	-29.3	1.1
640	24	-1.8	1.0	-2.3	0.9	-5.3	1.1	-15.7	1.1	-28.5	1.1
640	25	-1.8	1.0	-2.2	0.9	-5.1	1.1	-15.2	1.1	-27.7	1.1
640	26	-1.7	1.1	-2.1	0.9	-4.9	1.0	-14.7	1.1	-26.9	1.1
640	27	-1.7	1.1	-2.0	0.8	-4.7	1.0	-14.1	1.1	-26.2	1.1
640	28	-1.6	1.1	-1.9	0.9	-4.5	1.0	-13.6	1.1	-25.5	1.1
640	29	-1.5	1.1	-1.8	0.9	-4.3	1.0	-13.2	1.1	-24.9	1.1
640	30	-1.5	1.0	-1.8	0.9	-4.1	1.1	-12.8	1.1	-24.2	1.1
640	31	-1.4	1.0	-1.7	0.9	-3.9	1.1	-12.3	1.0	-23.6	1.0
640	32	-1.4	1.0	-1.6	0.9	-3.8	1.1	-12.0	1.0	-23.1	1.0
640	33	-1.5	1.0	-1.5	0.9	-3.6	1.1	-11.6	1.0	-22.5	1.0
640	34	-1.5	1.0	-1.4	0.9	-3.4	1.1	-11.2	1.0	-22.0	1.0
640	35	-1.5	1.0	-1.3	0.9	-3.3	1.1	-10.9	1.0	-21.5	1.0
640	36	-1.5	1.0	-1.2	0.9	-3.1	1.1	-10.5	1.0	-21.0	1.0
640	37	-1.5	1.0	-1.1	0.9	-3.0	1.1	-10.2	1.1	-20.5	1.0
640	38	-1.6	1.0	-1.0	0.9	-2.9	1.1	-9.9	1.1	-20.1	1.0
640	39	-1.6	1.0	-0.9	0.9	-2.7	1.1	-9.6	1.1	-19.6	1.0
640	40	-1.6	1.0	-0.8	0.9	-2.6	1.0	-9.3	1.1	-19.2	1.0
640	41	-1.6	1.0	-0.8	0.9	-2.5	1.0	-9.0	1.1	-18.8	1.0
640	42	-1.6	1.0	-0.7	0.9	-2.3	1.0	-8.7	1.1	-18.4	1.0
640	43	-1.6	1.0	-0.6	0.9	-2.2	1.0	-8.5	1.1	-18.1	1.0
640	44	-1.6	1.0	-0.5	0.9	-2.1	1.0	-8.2	1.0	-17.7	1.0
640	45	-1.6	1.0	-0.5	0.9	-2.0	1.0	-7.9	1.0	-17.4	0.9
640	46	-1.5	1.0	-0.4	0.9	-1.9	1.0	-7.7	1.0	-17.0	17.0
640	47	-1.5	1.0	-0.4	0.9	-1.7	1.0	-7.4	1.0	-16.7	0.9

N	Clusters	Variables									
		1		2		4		8		16	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	48	-1.5	1.0	-0.3	0.9	-1.6	1.0	-7.2	1.0	-16.4	0.9
640	49	-1.5	1.0	-0.3	0.9	-1.5	1.0	-7.0	1.0	-16.1	0.9
640	50	-1.4	1.0	-0.3	0.9	-1.4	1.0	-6.8	1.0	-15.8	0.9
640	51	-1.4	1.0	-0.2	0.9	-1.3	1.0	-6.5	1.0	-15.5	0.9
640	52	-1.4	1.0	-*0.2	0.9	-1.2	1.0	-6.3	1.0	-15.2	0.9
640	53	-1.4	1.0	-0.1	0.9	-1.1	1.0	-6.1	1.0	-14.9	0.9
640	54	-1.4	1.0	-0.1	0.9	-1.1	1.0	-5.9	1.0	-14.7	0.9
640	55	-1.4	1.0	0.0	0.9	-1.0	1.1	-5.8	1.0	-14.4	0.9
640	56	-1.4	1.0	0.0	0.9	-0.9	1.1	-5.6	1.0	-14.2	0.9
640	57	-1.4	1.0	0.1	0.9	-0.8	1.1	-5.4	1.0	-13.9	0.9
640	58	-1.4	1.0	0.1	0.9	-0.7	1.1	-5.2	1.0	-13.7	0.9
640	59	-1.4	1.0	0.2	0.9	-0.6	1.1	-5.1	1.0	-13.5	0.9
640	60	-1.3	1.0	0.2	0.8	-0.5	1.1	-4.9	0.9	-13.2	0.9
640	61	-1.3	1.0	0.3	0.8	-0.4	1.1	-4.8	0.9	-13.0	0.9
640	62	-1.3	0.9	0.3	0.8	-0.4	1.1	-4.6	0.9	-12.8	0.9
640	63	-1.3	0.9	0.4	0.8	-0.3	1.1	-4.5	0.9	-12.6	0.9
640	64	-1.3	0.9	0.5	0.8	-0.2	1.1	-4.3	0.9	-12.4	0.9

Note: Each mean and standard deviation is based on 50 samples.

Figures 3.1 – 3.5 plot the probability of the CCC exceeding 2.0 for each combination of number of observations, clusters, and variables. All probabilities are less than 0.10 and most are less than 0.05. Table 2 shows the probability of the maximum CCC exceeding 2.0, where the maximum is taken over numbers of clusters, for each combination of number of observations and variables. All probabilities are less than 0.10. The maximum CCC exceeded 3.0 only once in the study, for 160 observations and 1 variable. Therefore, a CCC value exceeding 2 or 3 can be taken as evidence favoring rejection of the null hypothesis of a uniform distribution on a hyperbox, although a precise significance level cannot be specified.

Figure 15.3 *Figure 3.1: Probability of CCC Exceeding 2.0 Plotted against the Number of Clusters for Uniform Hypercubical Distributions, Number of Variables = 1*

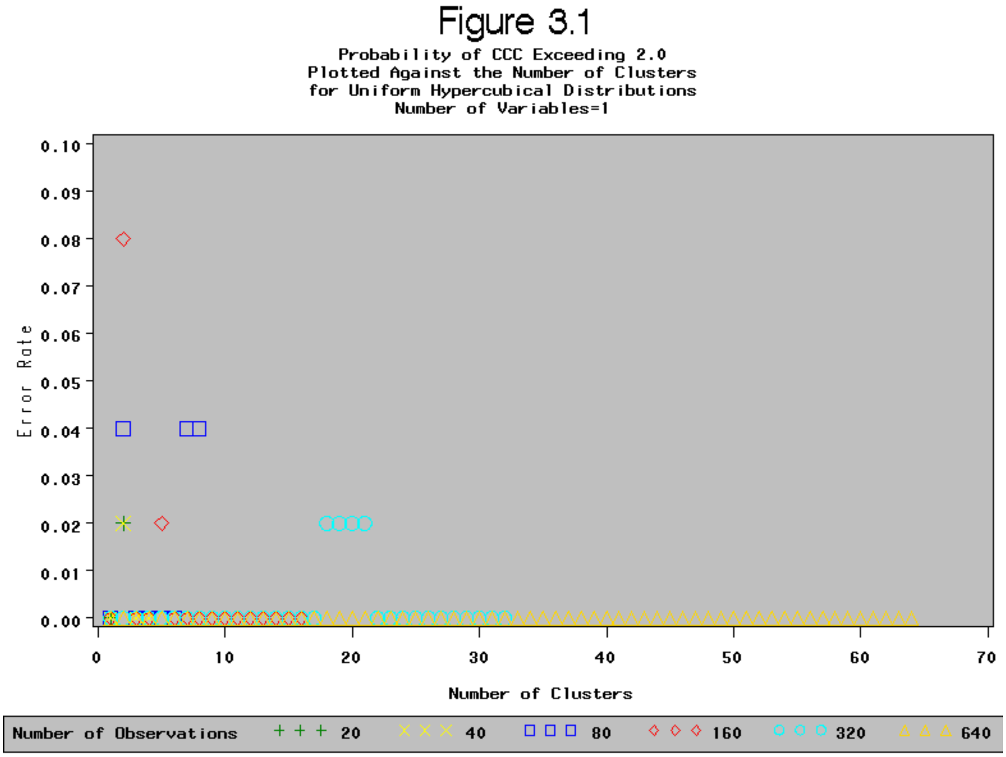


Figure 15.4 *Figure 3.2: Probability of CCC Exceeding 2.0 Plotted Against the Number of Clusters for Uniform Hypercubical Distributions, Number of Variables = 2*

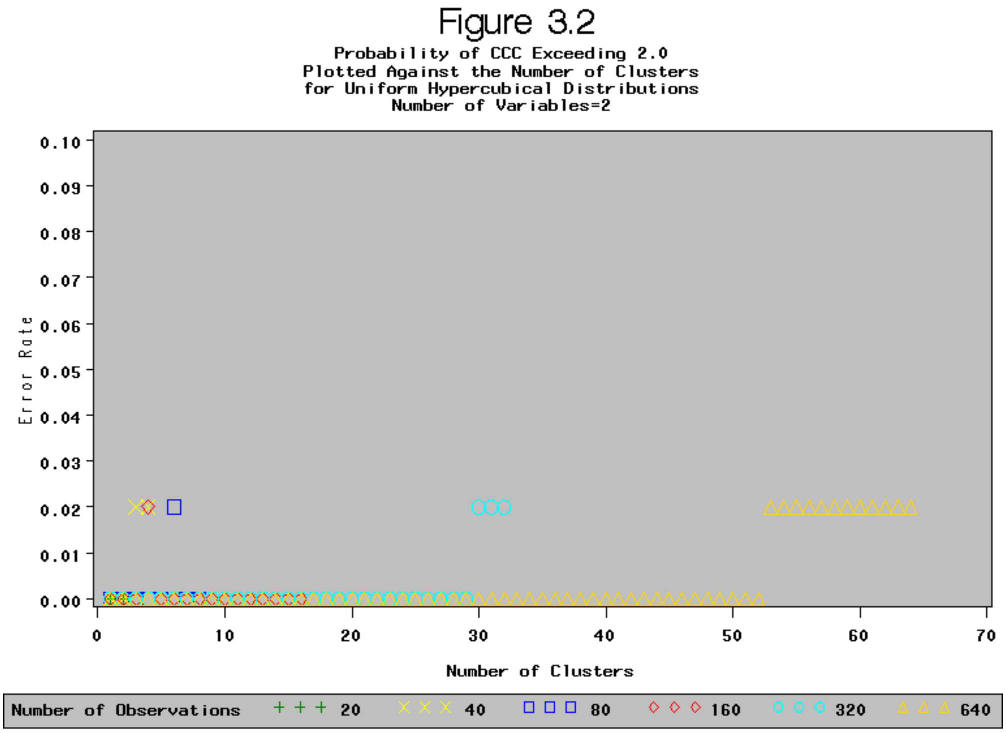


Figure 15.5 Figure 3.3: Probability of CCC Exceeding 2.0 Plotted Against the Number of Clusters for Uniform Hypercubical Distributions, Number of Variables = 4

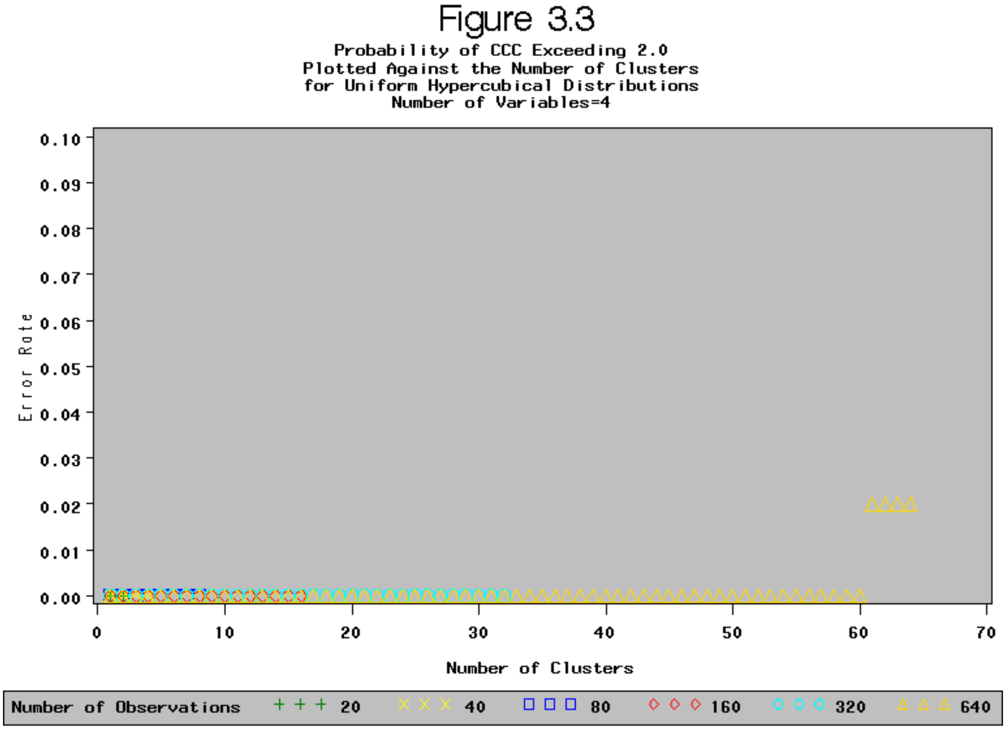


Figure 15.6 Figure 3.4: Probability of CCC Exceeding 2.0 Plotted Against the Number of Clusters for Uniform Hypercubical Distributions, Number of Variables = 8

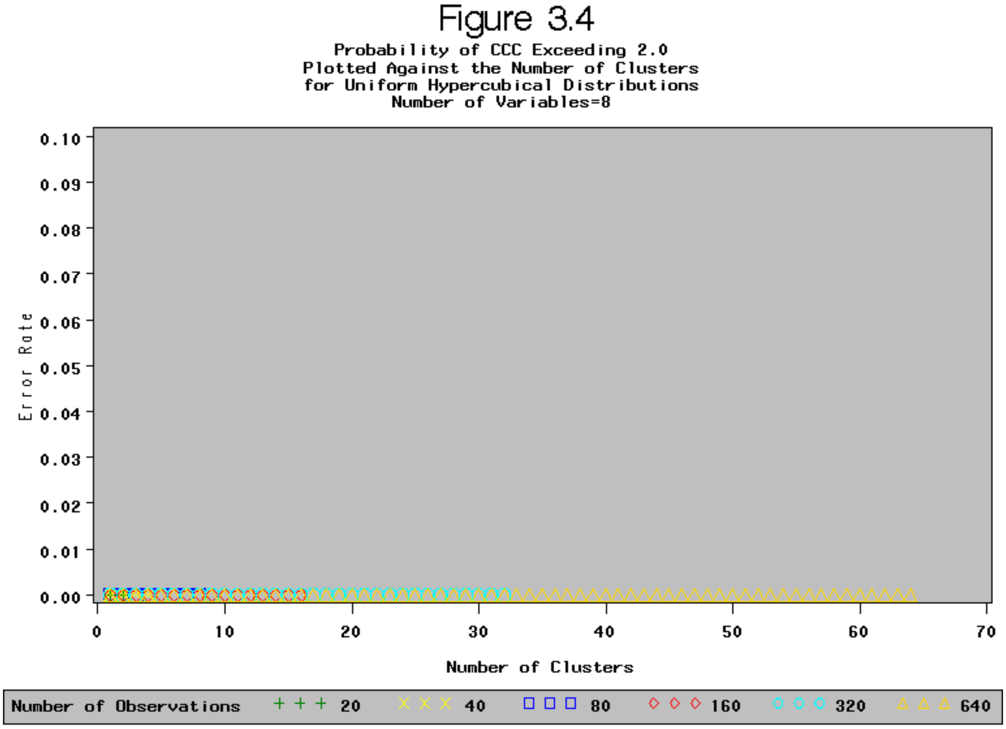
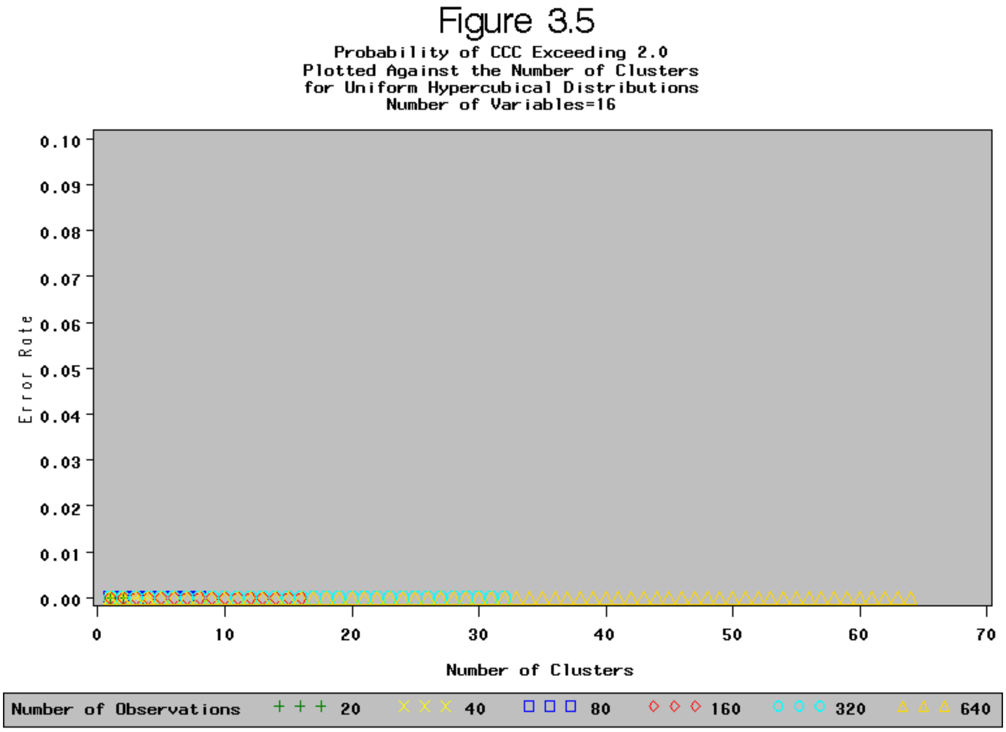


Figure 15.7 *Figure 3.5: Probability of CCC Exceeding 2.0 Plotted Against the Number of Clusters for Uniform Hypercubical Distributions, Number of Variables = 16*



Observations	Variables				
	1	2	4	8	16
20	2	0	0	0	0
40	2	2	0	0	0
80	8	2	0	0	0
160	8	2	0	0	0
320	2	2	0	0	0
640	0	2	2	0	0

Note: Each table entry is based on 50 samples.

The first Monte Carlo study examined hypercubical distributions. A second Monte Carlo was run to evaluate the CCC in uniform distributions on non-cubical hyperboxes. To keep computer time within reasonable limits the dimensionality was limited to four while the ranges were varied in three dimensions. The number of observations was 80, 160, 320, or 640. Fifty samples were generated in each cell. Tables 3.1 and 3.2 give the mean and standard deviation of the CCC for each combination of number of observations, clusters, and shape of hyperbox. The shapes are given as four numbers indicating the ranges in the four dimensions. Again the results are conservative. Error rates analogous to those in Table 2 were computed, and none exceeded the 0.02 level.

Table 15.2 *Table 2: Mean and Standard Deviation of the CCC for Each Combination of Number of Observations, Clusters, and Shape of Hyperbox.*

N	Clusters	Shape							
		1111		**2111**		**2211**		**2221**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**
80	2	-2.4	0.7	-1.1	0.8	-1.5	0.8	-1.9	0.9
80	3	-3.2	0.8	-2.1	0.7	-2.7	0.8	-2.9	0.8
80	4	-4	0.8	-2.7	0.7	-1.9	1.1	-3.7	0.9
80	5	-4.3	0.9	-3.2	0.8	-1.8	1	-2.9	0.9
80	6	-3.6	1	-2.9	0.8	-1.8	0.9	-2.4	0.8
80	7	-3.1	1	-2.7	0.9	-1.8	0.9	-1.9	0.9
80	8	-2.6	0.9	-2.4	0.8	-1.8	0.8	-1.6	1
160	2	-3.5	1	-1.7	1.1	-2	1.1	-2.7	1.2
160	3	-5	0.8	-2.9	1	-4.4	1	-4.5	0.7
160	4	-6.2	0.9	-4.2	0.9	-3.1	1.4	-6.3	0.9
160	5	-7.2	0.9	-5.1	1	-3.5	1.2	-4.8	1
160	6	-6	1	-4.8	0.9	-3.6	1.2	-3.7	1.1
160	7	-5.1	1	-4.5	0.9	-3.5	1.1	-3.1	1.1
160	8	-4.4	1	-4.2	0.8	-3.4	1	-2.8	1.2
160	9	-3.7	1	-3.9	0.8	-3.3	1	-2.6	1.1
160	10	-3.2	1	-3.7	0.8	-3.1	0.9	-2.5	1.1
160	11	-2.7	1	-3.4	0.9	-3	0.9	-2.3	1.2
160	12	-2.3	1	-3.1	0.9	-2.9	1	-2.2	1.2
160	13	-2	1	-2.8	0.9	-2.7	1	-2.1	1.1
160	14	-1.8	1	-2.6	0.9	-2.5	1	-1.9	1.1
160	15	-1.5	1	-2.3	0.9	-2.3	1	-1.8	1
160	16	-1.4	1	-2.1	0.9	-2.2	1	-1.7	1
320	2	-5.2	1.3	-2.6	1.2	-2.9	1.2	-4.4	1.5
320	3	-7.6	1	-4.1	1	-6.8	1.2	-6.8	1.3
320	4	-10	1.1	-6.3	1	-4.7	1.6	-9.8	1.2

N	Clusters	Shape							
		1111		2111		2211		2221	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD
320	5	-12.5	1.1	-8.2	1	-5.4	1.5	-7.9	1.1
320	6	-11	1.2	-8.2	1	-5.5	1.2	-6.4	1.2
320	7	-9.8	1.2	-7.9	1	-5.6	1.1	-5.3	1.2
320	8	-8.7	1.3	-7.6	1.1	-5.6	1	-4.9	1.4
320	9	-7.7	1.3	-7.2	1.1	-5.5	0.9	-4.9	1.2
320	10	-6.8	1.4	-6.8	1.1	-5.4	0.9	-4.7	1.1
320	11	-6.1	1.4	-6.3	1.1	-5.3	0.8	-4.6	1
320	12	-5.5	1.4	-5.9	1.1	-5.2	0.7	-4.4	1
320	13	-5	1.3	-5.5	1.1	-5	0.7	-4.3	0.9
320	14	-4.5	1.3	-5.1	1.1	-4.8	0.7	-4.1	0.9
320	15	-4.2	1.3	-4.8	1.1	-4.6	0.7	-4	0.9
320	16	-3.8	1.2	-4.4	1.1	-4.4	0.7	-3.8	0.9
320	17	-3.5	1.2	-4.1	1	-4.1	0.7	-3.7	0.9
320	18	-3.3	1.2	-3.8	1	-3.9	0.7	-3.6	0.9
320	19	-3.1	1.1	-3.5	1	-3.7	0.7	-3.4	0.9
320	20	-2.8	1.1	-3.3	1	-3.5	0.8	-3.3	0.9
320	21	-2.6	1.1	-3	1	-3.3	0.8	-3.1	0.9
320	22	-2.5	1.1	-2.8	1	-3.1	0.8	-3	0.9
320	23	-2.3	1.1	-2.6	1	-2.9	0.8	-2.9	0.9
320	24	-2.1	1.1	-2.4	1.1	-2.7	0.8	-2.7	0.9
320	25	-2	1.1	-2.2	1.1	-2.5	0.8	-2.6	0.9
320	26	-1.8	1.1	-2.1	1.1	-2.3	0.8	-2.5	0.9
320	27	-1.7	1.1	-1.9	1.1	-2.2	0.8	-2.4	0.9
320	28	-1.5	1.1	-1.8	1	-2	0.8	-2.2	0.9
320	29	-1.4	1.1	-1.7	1	-1.9	0.8	-2.1	0.9
320	30	-1.3	1.1	-1.5	1	-1.7	0.8	-2	0.9

N	Clusters	Shape							
		1111		2111		2211		2221	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD
320	31	-1.2	1.1	-1.4	1	-1.6	0.8	-1.8	0.9
320	32	-1.1	1.1	-1.3	1	-1.4	0.8	-1.7	0.9
640	2	-7.5	1.6	-3.3	1.5	-4.4	1.8	-5.3	2.3
640	3	-11.3	1.1	-6.1	1.1	-9.8	1.5	-10.1	1.5
640	4	-15.2	1.2	-9.4	1.1	-7.1	2.1	-15	1.4
640	5	-19.4	1.3	-12.4	1.3	-8.5	1.6	-12.5	1.4
640	6	-17.3	1.3	-12.6	1.2	-8.5	1.4	-10.2	1.4
640	7	-15.4	1.3	-12.3	1.1	-8.6	1.3	-8.5	1.5
640	8	-13.8	1.4	-12	1.1	-8.8	1.2	-7.9	1.8
640	9	-12.5	1.3	-11.6	1.2	-8.9	1.2	-7.9	1.5
640	10	-11.3	1.3	-11.1	1.2	-9	1.1	-7.8	1.4
640	11	-10.3	1.3	-10.6	1.2	-9	1.1	-7.6	1.3
640	12	-9.5	1.3	-10.1	1.2	-8.9	1.1	-7.5	1.3
640	13	-8.7	1.4	-9.6	1.2	-8.7	1.1	-7.4	1.3
640	14	-8.1	1.5	-9.1	1.2	-8.6	1	-7.2	1.2
640	15	-7.6	1.4	-8.6	1.2	-8.4	1	-7	1.2
640	16	-7.2	1.3	-8.2	1.2	-8.2	1	-6.8	1.2
640	17	-6.9	1.2	-7.8	1.2	-7.9	1	-6.7	1.1
640	18	-6.6	1.2	-7.4	1.2	-7.7	1	-6.6	1.1
640	19	-6.4	1.1	-6.9	1.2	-7.4	1	-6.4	1.1
640	20	-6.1	1.1	-6.5	1.2	-7.1	0.9	-6.2	1.1
640	21	-5.9	1.1	-6.2	1.2	-6.8	0.9	-6.1	1.1
640	22	-5.7	1.1	-5.9	1.2	-6.6	0.9	-5.9	1
640	23	-5.5	1.1	-5.6	1.2	-6.3	0.9	-5.7	1
640	24	-5.3	1.1	-5.4	1.2	-6.1	0.9	-5.5	1
640	25	-5.1	1.1	-5.1	1.2	-5.8	0.9	-5.3	1

N	Clusters	Shape							
		1111		2111		2211		2221	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	26	-4.9	1.1	-4.9	1.2	-5.6	0.9	-5.2	1
640	27	-4.7	1.1	-4.7	1.2	-5.4	0.8	-5	1
640	28	-4.6	1.1	-4.5	1.2	-5.2	0.8	-4.8	1
640	29	-4.4	1	-4.3	1.2	-5	0.8	-4.6	1
640	30	-4.2	1	-4.1	1.2	-4.8	0.8	-4.5	0.9
640	31	-4.1	1	-3.9	1.2	-4.6	0.8	-4.3	0.9
640	32	-3.9	1	-3.8	1.2	-4.4	0.8	-4.1	0.9
640	33	-3.8	0.9	-3.6	1.2	-4.2	0.8	-3.9	0.9
640	34	-3.6	0.9	-3.5	1.1	-4.1	0.8	-3.8	0.9
640	35	-3.5	0.9	-3.4	1.1	-3.9	0.7	-3.6	0.9
640	36	-3.3	0.9	-3.3	1.1	-3.7	0.7	-3.5	0.9
640	37	-3.2	0.9	-3.1	1.1	-3.6	0.7	-3.4	0.9
640	38	-3	0.9	-3	1.1	-3.4	0.7	-3.2	0.9
640	39	-2.9	0.9	-2.9	1.1	-3.3	0.8	-3.1	0.9
640	40	-2.8	0.9	-2.8	1.1	-3.2	0.8	-3	0.9
640	41	-2.6	0.9	-2.7	1.1	-3	0.8	-2.8	0.9
640	42	-2.5	0.9	-2.6	1.1	-2.9	0.8	-2.7	0.9
640	43	-2.4	0.9	-2.5	1.1	-2.8	0.8	-2.6	0.9
640	44	-2.2	0.9	-2.4	1.1	-2.7	0.8	-2.5	0.9
640	45	-2.1	0.9	-2.3	1.1	-2.5	0.8	-2.3	0.9
640	46	-2	0.9	-2.2	1.1	-2.4	0.8	-2.2	0.9
640	47	-1.9	0.9	-2.1	1.1	-2.3	0.8	-2.1	0.9
640	48	-1.8	0.9	-2	1	-2.2	0.8	-2	0.9
640	49	-1.7	0.9	-1.9	1	-2	0.8	-1.9	0.9
640	50	-1.6	0.9	-1.8	1	-1.9	0.8	-1.8	0.9
640	51	-1.4	0.9	-1.7	1	-1.8	0.8	-1.7	0.9

N	Clusters	Shape							
		1111		2111		2211		2221	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	52	-1.3	0.9	-1.6	1	-1.7	0.8	-1.6	0.9
640	53	-1.2	0.9	-1.5	1	-1.6	0.8	-1.5	0.9
640	54	-1.2	0.9	-1.5	1	-1.5	0.8	-1.4	0.9
640	55	-1.1	0.9	-1.4	1	-1.4	0.8	-1.3	0.9
640	56	-1	0.9	-1.3	1	-1.3	0.8	-1.2	0.9
640	57	-0.9	0.9	-1.2	1	-1.2	0.8	-1.1	0.9
640	58	-0.8	0.9	-1.1	1	-1.1	0.8	-1	0.9
640	59	-0.7	0.9	-1.1	1	-1	0.8	-0.9	0.9
640	60	-0.6	0.9	-1	0.9	-0.9	0.8	-0.8	0.9
640	61	-0.5	0.9	-0.9	0.9	-0.8	0.8	-0.9	0.9
640	62	-0.4	0.9	-0.9	0.9	-0.8	0.8	-0.7	0.9
640	63	-0.4	0.9	-0.8	0.9	-0.7	0.9	-0.6	0.9
640	64	-0.3	0.9	-0.7	0.9	-0.6	0.9	-0.5	0.9

Note: Each mean and standard deviation is based on 50 samples.

Table 15.3 *Table 3: Mean and Standard Deviation of the CCC for Each Combination of Number of Observations, Clusters, and Shape of Hyperbox.*

N	Clusters	Shape											
		4111		4211		4221		4411		4421		4441	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
80	2	-0.8	1	-1.1	1	-1.2	1	-1.3	1.1	-1.5	0.8	-2	0.8
80	3	-0.6	0.6	-1.9	0.9	-1.7	0.8	-2.7	1.2	-2.8	1	-2.8	0.9
80	4	-1	0.6	-1.9	0.9	-2.5	0.8	-1.4	1.2	-2	1.1	-3.7	1.1
80	5	-1.3	0.6	-1.6	0.8	-2.4	0.8	-1.4	0.9	-2.1	0.9	-2.7	1.2
80	6	-1.5	0.6	-1.4	0.8	-2.2	0.9	-1.2	0.9	-2.1	0.8	-2	1.2
80	7	-1.7	0.7	-1.3	0.9	-1.9	0.9	-1.1	0.8	-2	0.8	-1.5	1.1
80	8	-1.8	0.7	-1.3	0.8	-1.7	0.9	-1	0.7	-1.9	0.8	-1.3	1

N	Clusters	Shape											
		4111		**4211**		**4221**		**4411**		**4421**		**4441**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**
160	2	-1.7	1.5	-1.3	1.2	-1.1	1	-1.9	1.1	-1.6	1	-2.5	1.1
160	3	-1.3	0.7	-2.5	1	-2.4	0.8	-4.6	1.1	-4.3	1.1	-4.4	0.9
160	4	-1.6	0.8	-2.8	0.8	-3.8	0.8	-2.4	1.4	-2.5	1.1	-5.9	1.2
160	5	-2.1	0.7	-2.4	0.8	-3.7	0.9	-2.4	1.1	-2.7	1.1	-4.7	1.1
160	6	-2.6	0.7	-2	0.7	-3.4	1	-2.1	0.9	-2.7	1	-3.7	1.3
160	7	-2.8	0.7	-1.8	0.8	-3.1	0.9	-1.9	0.8	-2.6	0.9	-3	1.3
160	8	-2.9	0.6	-2	0.8	-2.8	0.9	-1.6	0.8	-2.5	0.9	-2.4	1.2
160	9	-2.9	0.6	-2	0.9	-2.5	0.8	-1.5	0.8	-2.3	0.9	-2.1	1.1
160	10	-2.9	0.6	-2	0.8	-2.2	0.8	-1.4	0.8	-2.2	0.9	-1.8	1
160	11	-2.8	0.6	-2.1	0.8	-2	0.8	-1.3	0.7	-2	0.9	-1.6	1
160	12	-2.7	0.7	-2.1	0.8	-1.8	0.8	-1.2	0.7	-1.9	0.8	-1.5	1
160	13	-2.6	0.7	-2.1	0.8	-1.6	0.8	-1.2	0.7	-1.7	0.8	-1.3	0.9
160	14	-2.5	0.7	-2	0.8	-1.5	0.9	-1.2	0.8	-1.6	0.8	-1.2	0.9
160	15	-2.4	0.7	-2	0.8	-1.3	0.9	-1.2	0.7	-1.4	0.8	-1.1	0.9
160	16	-2.3	0.7	-1.9	0.8	-1.2	0.9	-1.3	0.7	-1.3	0.8	-1	0.9
320	2	-2	1.6	-1.8	1.5	-2.3	1.4	-2.6	1.5	-2.9	1.5	-3.7	1.6
320	3	-1.7	0.9	-3.9	1	-3.7	1.1	-6.8	1.3	-6.8	1.2	-6.6	1.5
320	4	-2.1	1	-4.3	0.8	-6.1	1.1	-3.9	1.8	-4.1	1.6	-9.6	1.3
320	5	-2.9	0.9	-3.8	0.9	-6.1	0.9	-3.6	1.3	-4.4	1.3	-8	1.2
320	6	-3.7	0.8	-3.2	1	-5.9	0.9	-3.3	1	-4.5	1.1	-6.5	1.3
320	7	-4.2	0.8	-2.8	1.1	-5.6	0.9	-3	0.9	-4.4	1	-5.3	1.4
320	8	-4.6	0.8	-2.9	1.2	-5.2	0.9	-2.7	0.8	-4.3	1	-4.4	1.2
320	9	-4.8	0.8	-3.1	1.2	-4.7	0.9	-2.5	0.8	-4.2	1	-4	1.2
320	10	-4.9	0.8	-3.2	1.2	-4.3	0.9	-2.4	0.8	-4	1	-3.8	1.2
320	11	-4.9	0.8	-3.3	1.1	-3.9	0.9	-2.3	0.7	-3.8	1	-3.6	1.1
320	12	-4.9	0.8	-3.3	1	-3.6	0.9	-2.2	0.7	-3.6	1.1	-3.5	1

N	Clusters	Shape											
		4111		4211		4221		4411		4421		4441	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
320	13	-4.9	0.8	-3.3	1	-3.3	0.9	-2.1	0.7	-3.4	1.1	-3.2	0.9
320	14	-4.8	0.8	-3.3	0.9	-3	0.9	-2.1	0.7	-3.2	1	-3	0.9
320	15	-4.6	0.7	-3.3	0.9	-2.9	0.9	-2.2	0.8	-2.9	1	-2.8	0.9
320	16	-4.5	0.7	-3.3	0.9	-2.8	1	-2.2	0.8	-2.7	1	-2.6	0.9
320	17	-4.3	0.7	-3.3	0.8	-2.8	1	-2.3	0.8	-2.5	1	-2.4	0.9
320	18	-4.1	0.7	-3.2	0.8	-2.7	0.9	-2.3	0.8	-2.4	1	-2.2	0.9
320	19	-4	0.7	-3.1	0.8	-2.6	0.9	-2.3	0.8	-2.2	1	-2.1	1
320	20	-3.8	0.7	-3	0.8	-2.5	0.9	-2.3	0.8	-2.1	1	-1.9	0.9
320	21	-3.6	0.7	-2.9	0.8	-2.4	0.9	-2.3	0.8	-1.9	1	-1.8	0.9
320	22	-3.4	0.7	-2.9	0.8	-2.3	0.9	-2.3	0.7	-1.8	1	-1.7	0.9
320	23	-3.2	0.7	-2.8	0.8	-2.2	0.8	-2.2	0.7	-1.7	1	-1.6	0.9
320	24	-3.1	0.7	-2.7	0.8	-2.1	0.8	-2.2	0.7	-1.6	1	-1.5	0.8
320	25	-2.9	0.7	-2.5	0.8	-2	0.8	-2.2	0.7	-1.5	1	-1.4	0.8
320	26	-2.8	0.7	-2.4	0.8	-1.9	0.8	-2.1	0.7	-1.4	1	-1.4	0.8
320	27	-2.6	0.7	-2.3	0.8	-1.8	0.8	-2.1	0.7	-1.4	1	-1.3	0.8
320	28	-2.5	0.7	-2.2	0.8	-1.7	0.8	-2	0.7	-1.3	1	-1.2	0.8
320	29	-2.3	0.7	-2.1	0.8	-1.6	0.8	-2	0.7	-1.2	1	-1.2	0.8
320	30	-2.2	0.7	-2	0.8	-1.5	0.8	-1.9	0.7	-1.2	1	-1.1	0.8
320	31	-2	0.7	-1.9	0.8	-1.4	0.7	-1.9	0.7	-1.1	1	-1	0.8
320	32	-1.9	0.7	-1.8	0.8	-1.3	0.7	-1.8	0.7	-1.1	0.9	-1	0.8
640	2	-2.7	2.7	-3.5	2.2	-3.5	1.9	-3.4	1.6	-3.8	1.4	-4.9	2.2
640	3	-2.9	0.9	-5.9	1.3	-5.7	1.2	-10.3	1.6	-9.9	1.6	-9.5	1.6
640	4	-3.3	1.1	-6.7	1	-9.3	1.1	-5.8	1.9	-6.5	1.8	-14.8	1.7
640	5	-4.7	1.1	-6.1	0.9	-9.7	1.1	-5.7	1.5	-7.1	1.5	-12.4	1.4

N	**Clusters**	**Shape**											
		4111		**4211**		**4221**		**4411**		**4421**		**4441**	
		Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**
640	6	-5.9	0.9	-5.2	1.1	-9.4	1.2	-5.3	1.1	-7	1.1	-10.2	1.4
640	7	-6.8	0.9	-4.5	0.9	-8.8	1.1	-5	1	-7.2	0.9	-8.4	1.4
640	8	-7.4	0.9	-4.8	1.2	-8.2	1	-4.6	1	-7.2	0.8	-7.2	1.2
640	9	-7.8	0.8	-5.3	1.2	-7.7	0.9	-4.3	0.9	-7.2	0.8	-6.7	1.1
640	10	-8.1	0.8	-5.6	1	-7.1	1	-4.1	0.8	-7.1	0.8	-6.4	1.1
640	11	-8.2	0.8	-5.7	0.9	-6.6	1	-4	0.8	-7	0.8	-6.2	1.1
640	12	-8.2	0.8	-5.8	0.8	-6.1	1.1	-3.8	0.8	-6.8	0.8	-6	1.1
640	13	-8.2	0.8	-5.9	0.8	-5.8	1	-3.6	0.8	-6.5	0.8	-5.7	1.1
640	14	-8.1	0.8	-5.9	0.8	-5.4	1	-3.5	0.7	-6.3	0.8	-5.4	1.1
640	15	-7.9	0.8	-6	0.8	-5.2	1.1	-3.5	0.9	-6	0.8	-5.2	1.1
640	16	-7.8	0.8	-6	0.8	-5.2	1.2	-3.9	0.9	-5.7	0.8	-5	1.1
640	17	-7.6	0.8	-6	0.8	-5.3	1.2	-4	0.9	-5.4	0.8	-4.8	1
640	18	-7.5	0.8	-6	0.8	-5.2	1.1	-4.1	0.9	-5.1	0.8	-4.6	1
640	19	-7.3	0.8	-6	0.8	-5.1	1	-4.2	0.9	-4.9	0.9	-4.4	1
640	20	-7.1	0.7	-5.9	0.8	-5	1	-4.1	0.8	-4.7	0.9	-4.2	0.9
640	21	-6.9	0.7	-5.9	0.8	-4.9	1	-4.1	0.8	-4.5	0.9	-4	0.9
640	22	-6.7	0.7	-5.9	0.8	-4.8	1	-4.1	0.8	-4.3	0.9	-3.9	0.9
640	23	-6.5	0.7	-5.8	0.8	-4.7	1	-4.1	0.8	-4.1	0.9	-3.7	0.9
640	24	-6.3	0.7	-5.7	0.8	-4.6	1	-4.1	0.8	-4	0.9	-3.6	0.9
640	25	-6	0.7	-5.6	0.8	-4.5	1	-4.1	0.8	-3.8	0.9	-3.5	0.9
640	26	-5.8	0.7	-5.5	0.8	-4.5	1	-4.1	0.8	-3.7	0.9	-3.4	0.9
640	27	-5.6	0.7	-5.4	0.9	-4.4	0.9	-4.1	0.8	-3.6	0.9	-3.3	0.9
640	28	-5.4	0.7	-5.3	0.9	-4.3	0.9	-4	0.8	-3.5	0.9	-3.1	0.9
640	29	-5.2	0.7	-5.2	0.8	-4.2	0.9	-4	0.8	-3.4	0.9	-3	0.9
640	30	-5	0.7	-5.1	0.8	-4.1	0.9	-4	0.8	-3.3	1	-2.9	0.9
640	31	-4.8	0.7	-5	0.8	-4	0.9	-3.9	0.8	-3.3	1	-2.8	0.9

N	Clusters	Shape											
		4111		4211		4221		4411		4421		4441	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	32	-4.6	0.7	-4.9	0.9	-3.9	0.9	-3.9	0.8	-3.2	1	-2.7	0.9
640	33	-4.4	0.7	-4.8	0.9	-3.8	0.9	-3.9	0.8	-3.2	1	-2.6	0.9
640	34	-4.3	0.7	-4.6	0.9	-3.7	0.9	-3.8	0.8	-3.2	1	-2.5	0.9
640	35	-4.1	0.8	-4.5	0.8	-3.6	0.9	-3.8	0.8	-3.1	0.9	-2.4	0.9
640	36	-4	0.8	-4.4	0.9	-3.5	0.9	-3.8	0.8	-3	0.9	-2.3	0.9
640	37	-3.8	0.8	-4.3	0.9	-3.5	0.9	-3.7	0.9	-2.9	0.9	-2.2	0.9
640	38	-3.7	0.8	-4.1	0.9	-3.4	0.8	-3.7	0.9	-2.8	0.9	-2.2	0.9
640	39	-3.6	0.8	-4	0.9	-3.3	0.8	-3.6	0.9	-2.8	0.9	-2.1	0.9
640	40	-3.4	0.8	-3.9	0.9	-3.2	0.8	-3.5	0.9	-2.7	0.9	-2	0.9
640	41	-3.3	0.8	-3.7	0.9	-3.1	0.8	-3.5	0.9	-2.6	0.9	-2	0.9
640	42	-3.3	0.8	-3.7	0.9	-3.1	0.8	-3.4	0.9	-2.5	1	-1.9	0.9
640	43	-3.2	0.8	-3.6	0.9	-3	0.8	-3.4	0.9	-2.4	0.9	-1.8	0.9
640	44	-2.9	0.8	-3.4	0.9	-2.8	0.8	-3.3	0.9	-2.4	0.9	-1.8	0.8
640	45	-2.8	0.8	-3.2	0.9	-2.8	0.8	-3.2	0.9	-2.3	1	-1.7	0.8
640	46	-2.7	0.8	-3.1	0.9	-2.7	0.8	-3.2	0.9	-2.2	1	-1.6	0.8
640	47	-2.6	0.8	-3	0.9	-2.6	0.8	-3.1	0.9	-2.2	1	-1.6	0.8
640	48	-2.5	0.8	-2.9	0.9	-2.5	0.8	-3	0.9	-2.1	0.9	-1.5	0.8
640	49	-2.4	0.8	-2.8	0.9	-2.4	0.8	-3	0.9	-2	0.9	-1.4	0.8
640	50	-2.3	0.8	-2.6	0.9	-2.3	0.8	-2.9	0.9	-2	0.9	-1.4	0.8
640	51	-2.2	0.8	-2.5	0.9	-2.2	0.8	-2.8	0.9	-1.9	0.9	-1.3	0.8
640	52	-2.1	0.8	-2.4	0.9	-2.2	0.8	-2.8	0.9	-1.8	0.9	-1.3	0.8
640	53	-2	0.8	-2.3	0.9	-2.1	0.8	-2.7	0.9	-1.8	1	-1.2	0.8
640	54	-1.9	0.8	-2.2	0.9	-2	0.8	-2.6	0.9	-1.7	1	-1.2	0.8
640	55	-1.8	0.8	-2.1	0.9	-1.9	0.8	-2.6	0.9	-1.6	1	-1.1	0.8
640	56	-1.7	0.8	-2	0.9	-1.8	0.8	-2.5	0.9	-1.6	1	-1.1	0.8
640	57	-1.6	0.8	-1.9	0.9	-1.7	0.8	-2.4	0.9	-1.5	1	-1	0.8

N	Clusters	Shape											
		4111		4211		4221		4411		4421		4441	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
640	58	-1.5	0.8	-1.8	0.9	-1.6	0.8	-2.3	0.9	-1.5	1	-1	0.8
640	59	-1.5	0.8	-1.8	0.9	-1.6	0.8	-2.3	0.9	-1.4	1	-0.9	0.8
640	60	-1.4	0.8	-1.7	0.9	-1.5	0.8	-2.2	0.9	-1.3	1	-0.9	0.8
640	61	-1.3	0.8	-1.6	0.9	-1.4	0.8	-2.1	0.9	-1.3	1	-0.8	0.8
640	62	-1.2	0.8	-1.5	0.9	-1.3	0.8	-2.1	0.9	-1.2	1	-0.8	0.8
640	63	-1.1	0.8	-1.4	0.9	-1.3	0.8	-2	0.9	-1.2	1	-0.8	0.8
640	64	-1.1	0.8	-1.4	0.9	-1.2	0.8	-1.9	0.9	-1.1	1	-0.7	0.8

Table 4 provides an indication of the power of the CCC to detect a mixture of two spherical normal distributions with unit variance and equal sampling probabilities. Ten samples of 100 observations were generated from each of 15 populations with 1, 2, 4, 8, or 16 variables and a distance between component means of 4, 5, or 6 standard deviations. Table 4 shows the frequency with which the CCC exceeded 2. The power decreases as the dimensionality increases, as expected. With 1 variable, a separation of 4 or 5 standard deviations is required for good power, while 16 variables require a separation of 6 or more standard deviations.

Table 15.4 *Table 4: Power of CCC to Detect MIxture of Two Spherical Normal Distributions with Unit Variance and Equal Sampling Probabilities*

Variables	Distance Between Centroids		
	4	5	6
1	5	10	10
2	3	10	10
4	0	8	10
8	0	4	10
16	0	0	7

Milligan and Cooper (1983) performed a Monte Carlo comparison of 30 criteria for the number of clusters, including the CCC. In the overall evaluation, the CCC ranked sixth best, correctly identifying the number of clusters 321 times in 432 attempts. The CCC tended to overestimate the number of clusters, probably because some of the clusters were elliptical rather than spherical.

Examples

Figures 4 through 6 show CCC plots for samples of 100 observations from various normal distributions clustered by Ward's method. In each case the CCC values are negative and generally decreasing as the number of clusters increases. Figure 4 is based on a univariate normal distribution. Figure 5 comes from a spherical multivariate normal distribution in 16 dimensions. Figure 6 illustrates an elliptical normal distribution in 16 dimensions, for which the standard deviation in the jth dimension is j.

Figure 15.8 *CCC Plot: 100 Observations from a Univariate Normal Distribution*

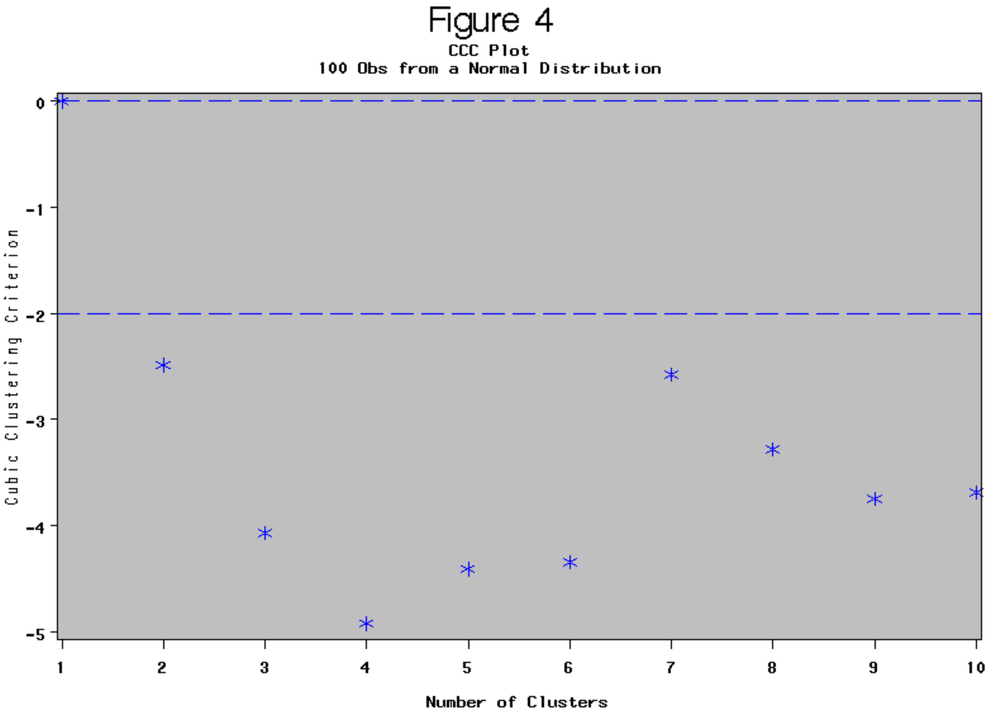


Figure 15.9 CCC Plot: 100 Observations from a Spherical Multivariate Normal Distribution in 16 Dimensions

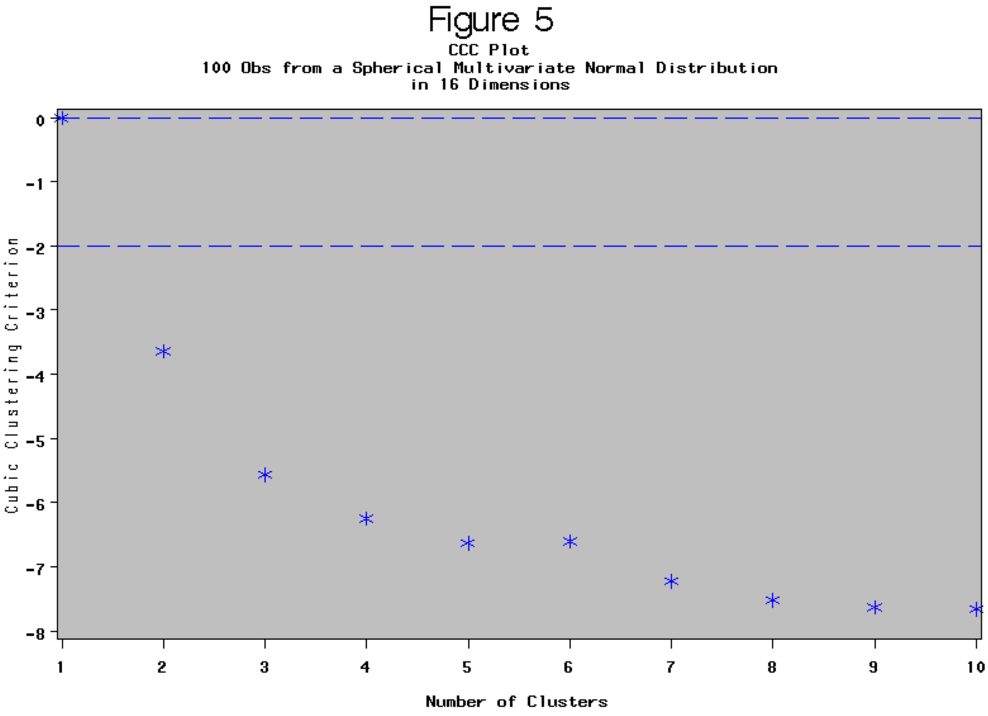


Figure 15.10 CCC Plot: 100 Observations from an Elliptical Multivariate Normal Distribution in 16 Dimensions

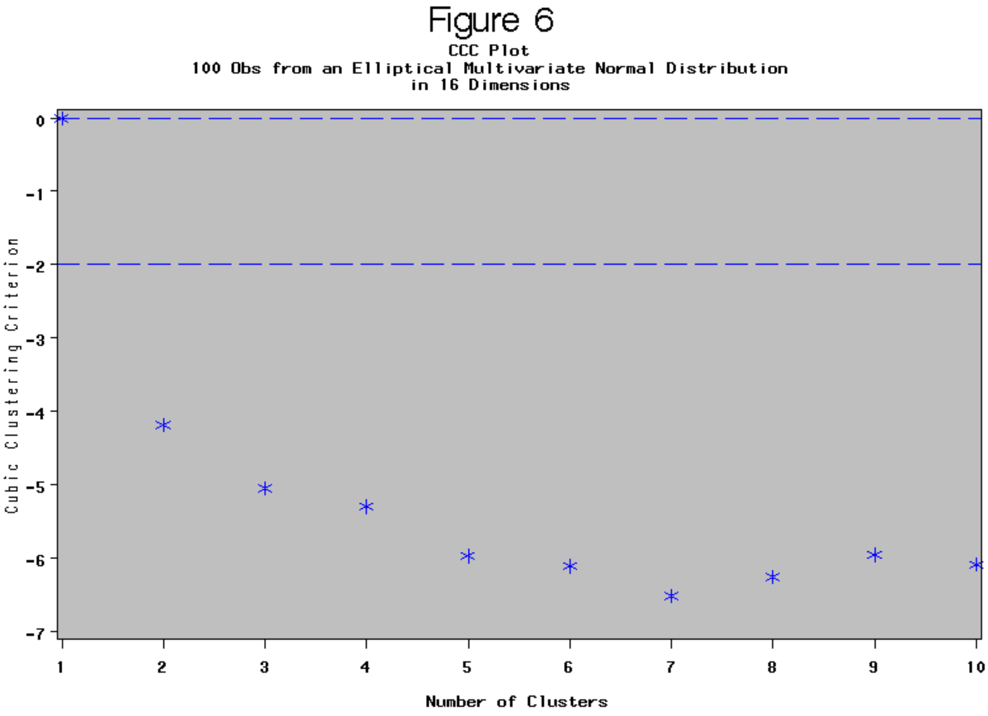


Figure 7.1 is a scatter plot of 100 observations from a mixture of two circular normal distributions (50 observations each) separated by 6 standard deviations. Figure 7.2 shows the corresponding CCC plot with a sharp peak clearly indicating two clusters. Figure 7.3

presents an analysis of the data in Figure 7.1 standardized to unit standard deviations. Standardization causes the clusters to become highly elliptical in violation of the alternative hypothesis on which the CCC is based. The resulting plot suggests the possibility of four or nine clusters. This example illustrates the danger of indiscriminate standardization.

Figure 15.11 *Plot of 50 Observations from each of Two Circular Normal Distributions Separated by Six Standard Deviations*

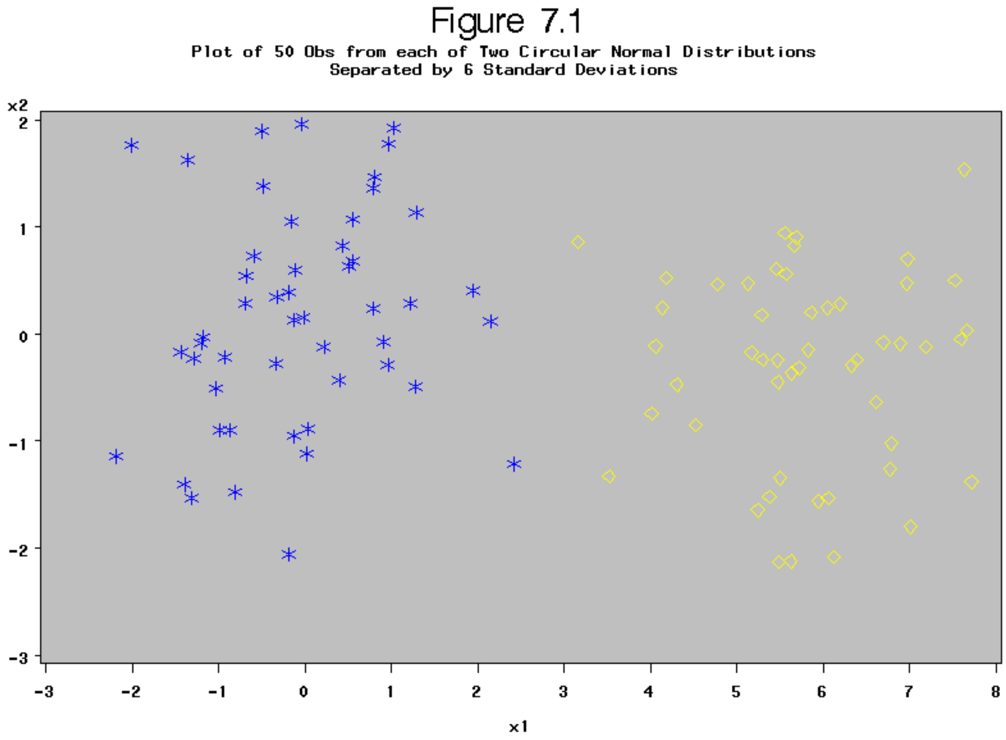


Figure 15.12 CCC Plot of Raw Data in Figure 7.1

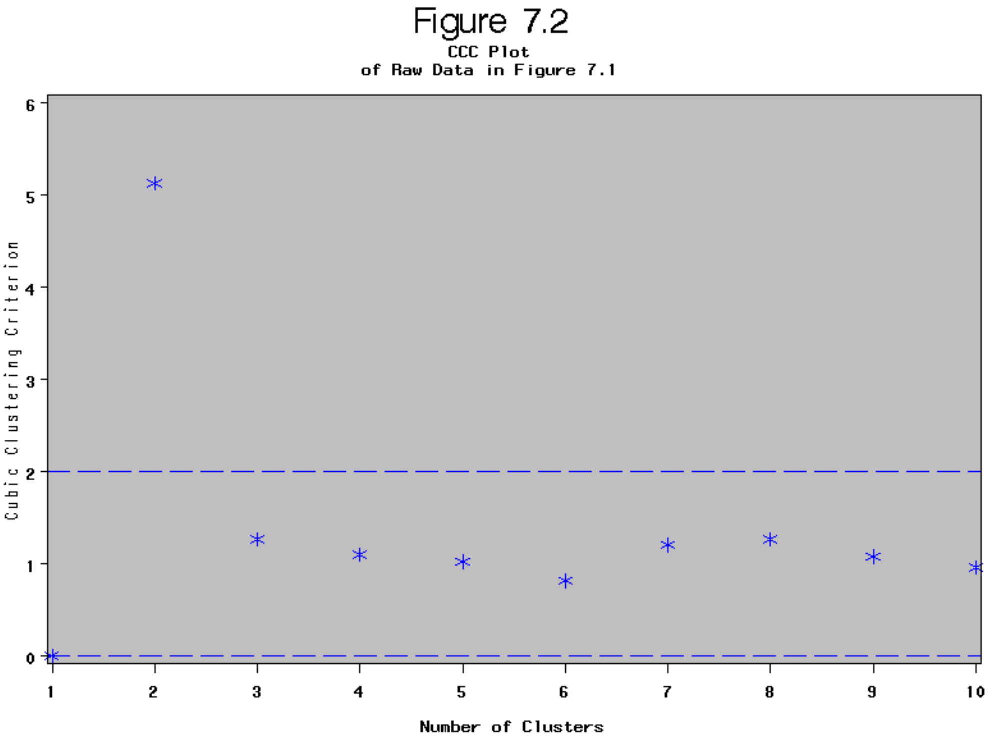


Figure 15.13 CCC Plot of Standardized Data in Figure 7.1

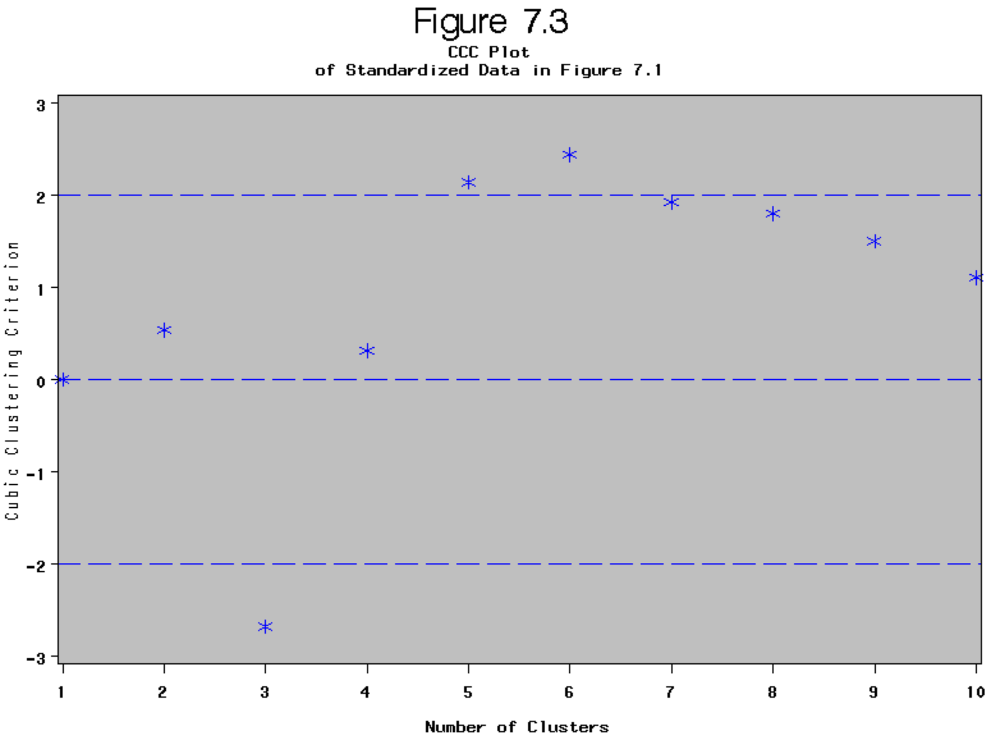
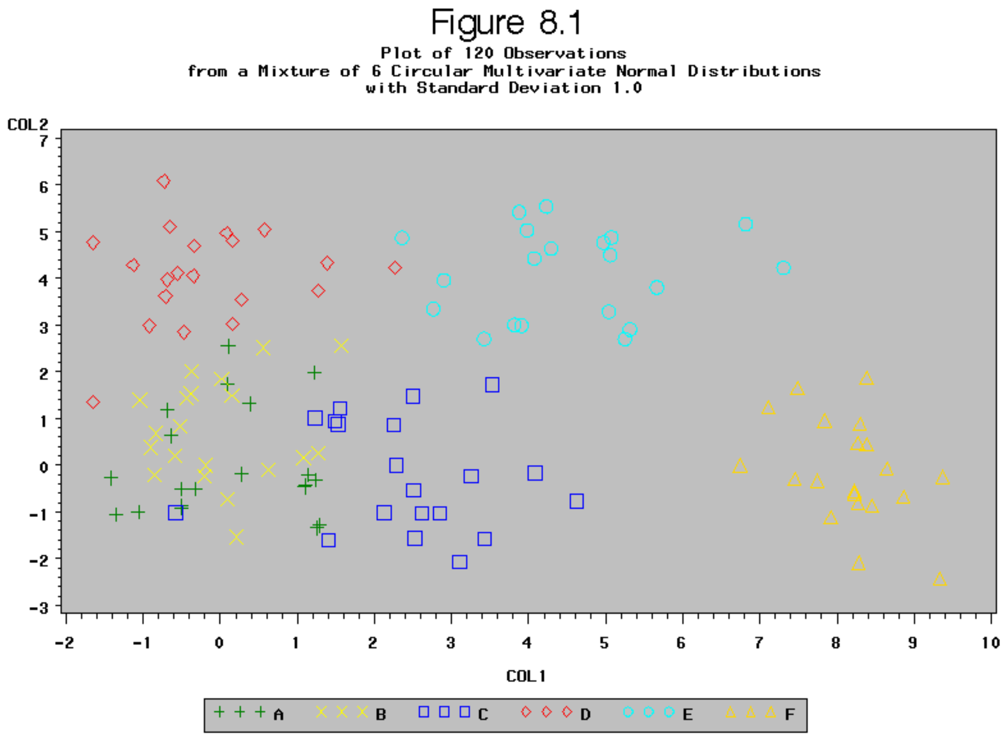


Table 5 gives the means of six clusters in two dimensions that are used to generate data in Figures 8.1 and 9.1. In the scatter plot in Figure 9.1, the standard deviation of each cluster is reduced to 0.25 units. The CCC plot in Figure 9.2 has a blunt peak at six clusters, suggesting either six circular clusters or five clusters of which one might be elliptical.

	Cluster Means	
Cluster	COL1	COL2
A	0	0
B	0	1
C	2	0
D	0	4
E	5	0
F	8	4

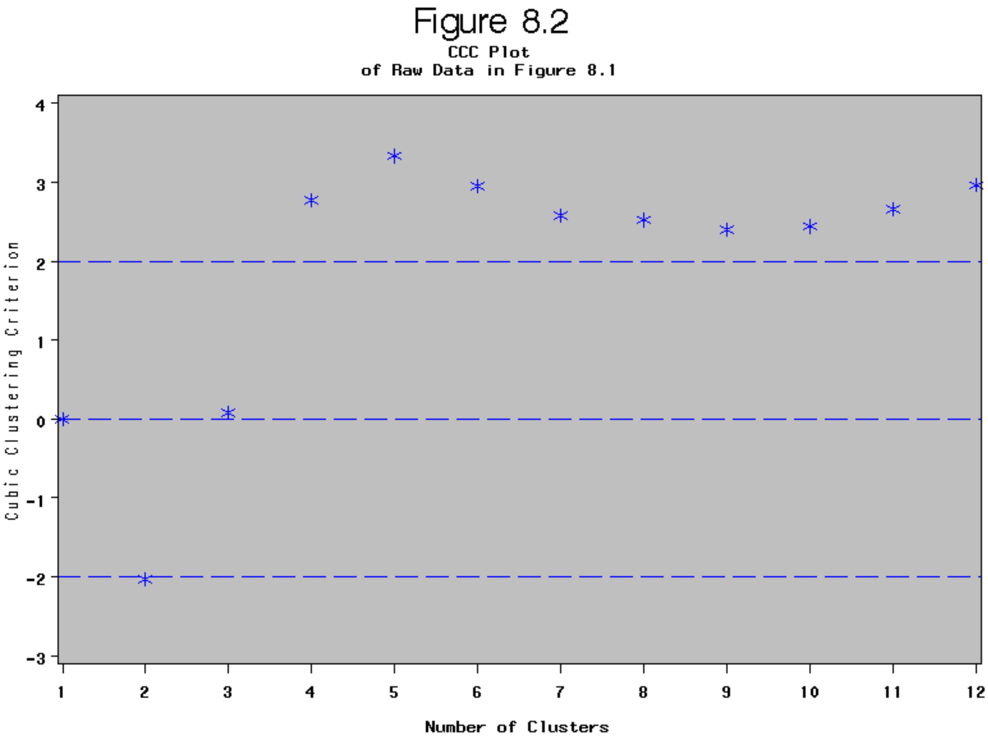
Figure 8.1 shows 120 observations from six circular normal distributions with the given means and standard deviations of 1.0 unit. It is apparent that there are at least four clusters, but the clusters labeled A, B, and C cannot be easily distinguished.

Figure 15.14 *Plot of 120 Observations from a Mixture of Six Circular Multivariate Normal Distributions with Standard Deviation 1.0*

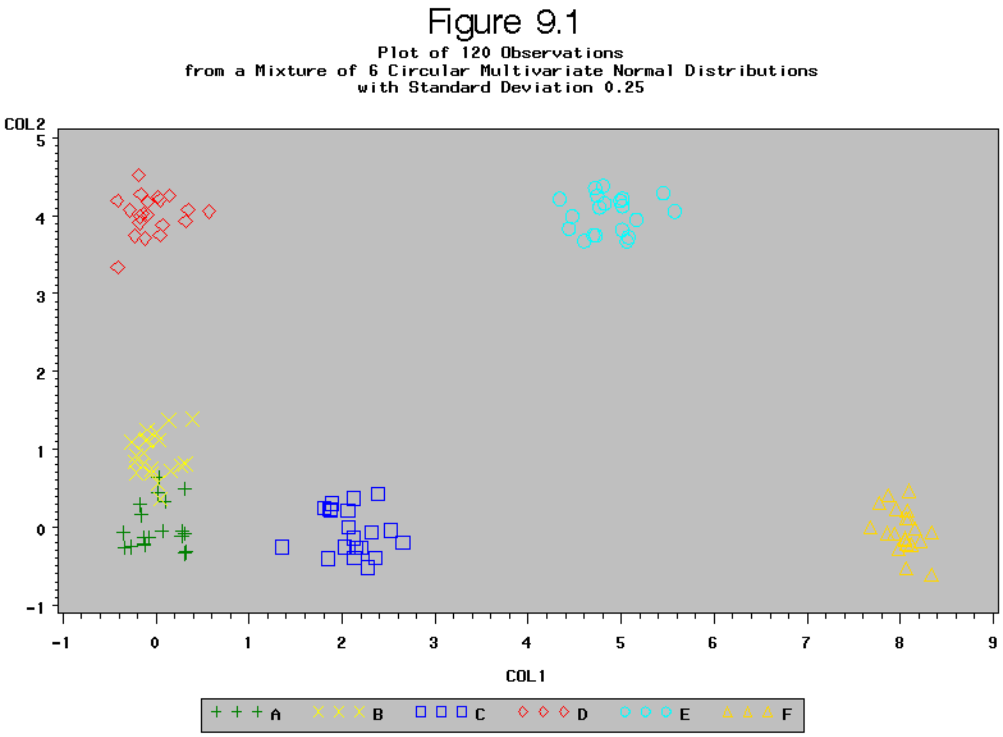


The CCC plot in Figure 8.2 has a peak at five clusters, but the peak is rather blunt, indicating that two of the five clusters are not well separated, or perhaps that there are only four clusters, one of which might be elliptical.

Figure 15.15 *CCC Plot of Raw Data in Figure 8.1*



In the scatter plot in Figure 9.1, the standard deviation of each cluster is reduced to 0.25 units.



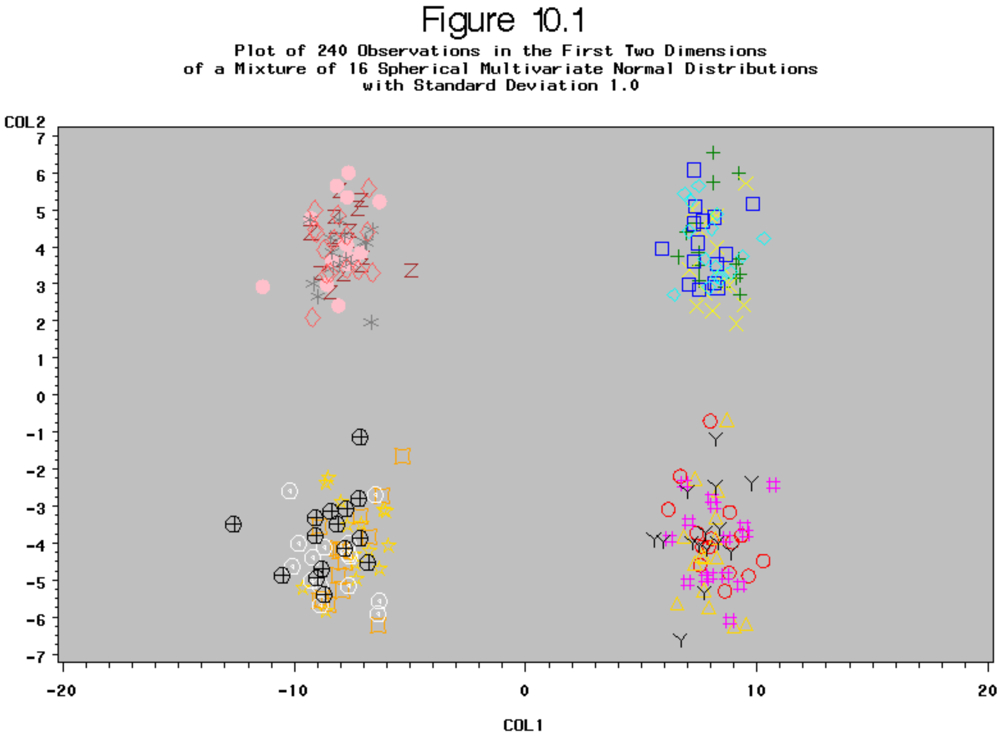
The CCC plot in Figure 9.2 has a blunt peak at six clusters, suggesting either six circular clusters or five clusters of which one might be elliptical.

Table 6 contains the means of 16 clusters in a hierarchical arrangement used to generate data in Figures 10.1 and 11.1.

Cluster	Cluster Means			
	COL1	COL2	COL3	COL4
A	8	4	2	1
B	8	4	2	-1
C	8	4	-2	1
D	8	4	-2	-1
E	8	-4	2	1
F	8	-4	2	-1
G	8	-4	-2	1
H	8	-4	-2	-1
I	-8	4	2	1
J	-8	4	2	-1
K	-8	4	-2	1
L	-8	4	-2	-1
M	-8	-4	2	1
N	-8	-4	2	-1
O	-8	-4	-2	1
P	-8	-4	-2	-1

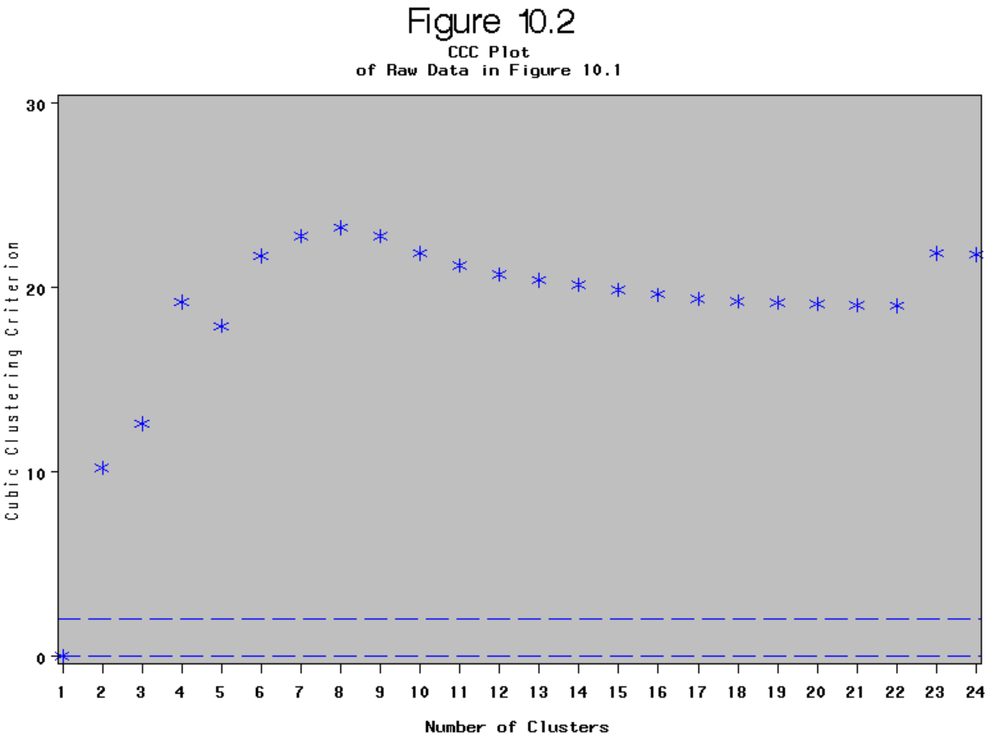
Figure 10.1 plots the first two dimensions, showing clusters with standard deviations of 1.0 unit. There are four apparent clusters, each of which is actually four clusters separated in the two dimensions not shown on the plot. The view through the other two dimensions would be similar but the apparent separation among the clusters would be reduced by a factor of four.

Figure 15.16 *Plot of 240 Observations in the First Two Dimensions of a Mixture of 16 Spherical Multivariate Normal Distributions with Standard Deviation 1.0*



The levels of interest in the hierarchy are 2, 4, 8, or 16 clusters. The first three of these levels can be seen in the CCC plot in Figure 10.2 as large jumps or local peaks in the CCC.

Figure 15.17 *CCC Plot of Raw Data in Figure 10.1*



In Figure 11.1 the standard deviations are reduced to 0.25 units, and the corresponding CCC plot in Figure 11.2 shows all four levels of the hierarchy.

Figure 15.18 *Plot of 240 Observations in the First Two Dimensions of a Mixture of Sixteen Spherical Multivariate Normal Distributions with Standard Deviation 0.25*

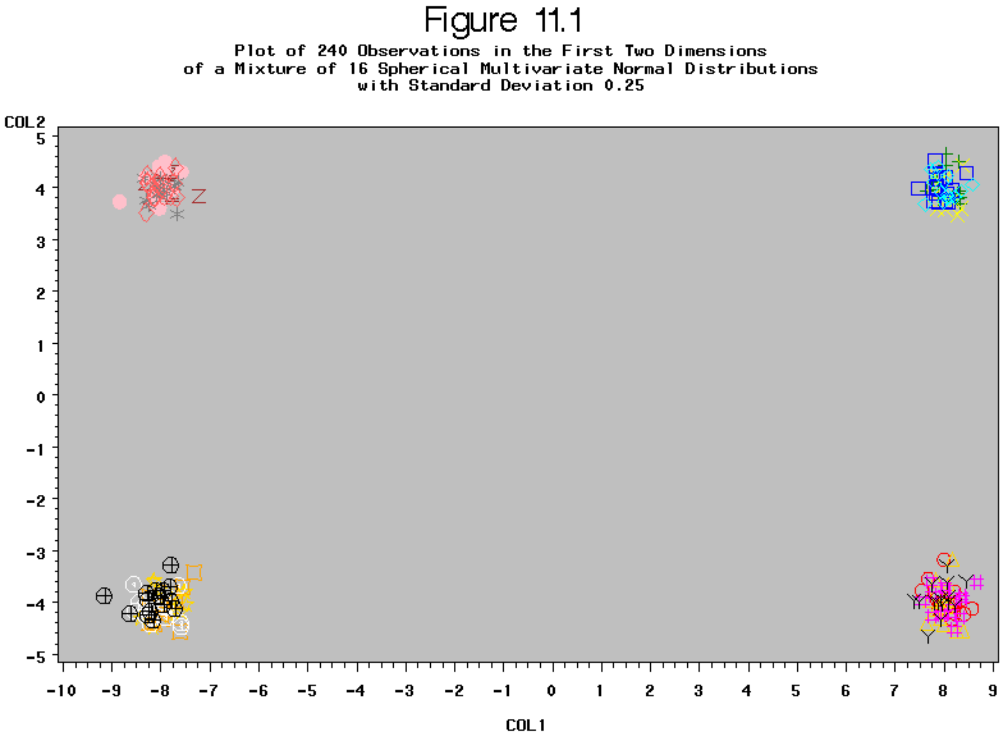


Figure 15.19 *CCC Plot of Raw Data in Figure 11.1*

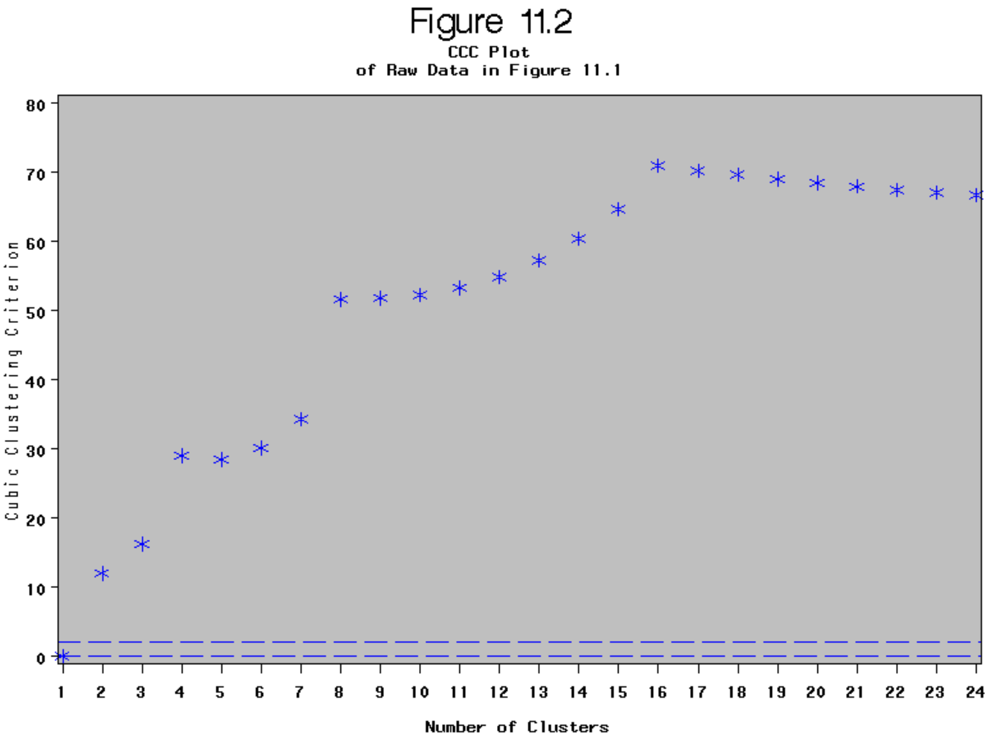
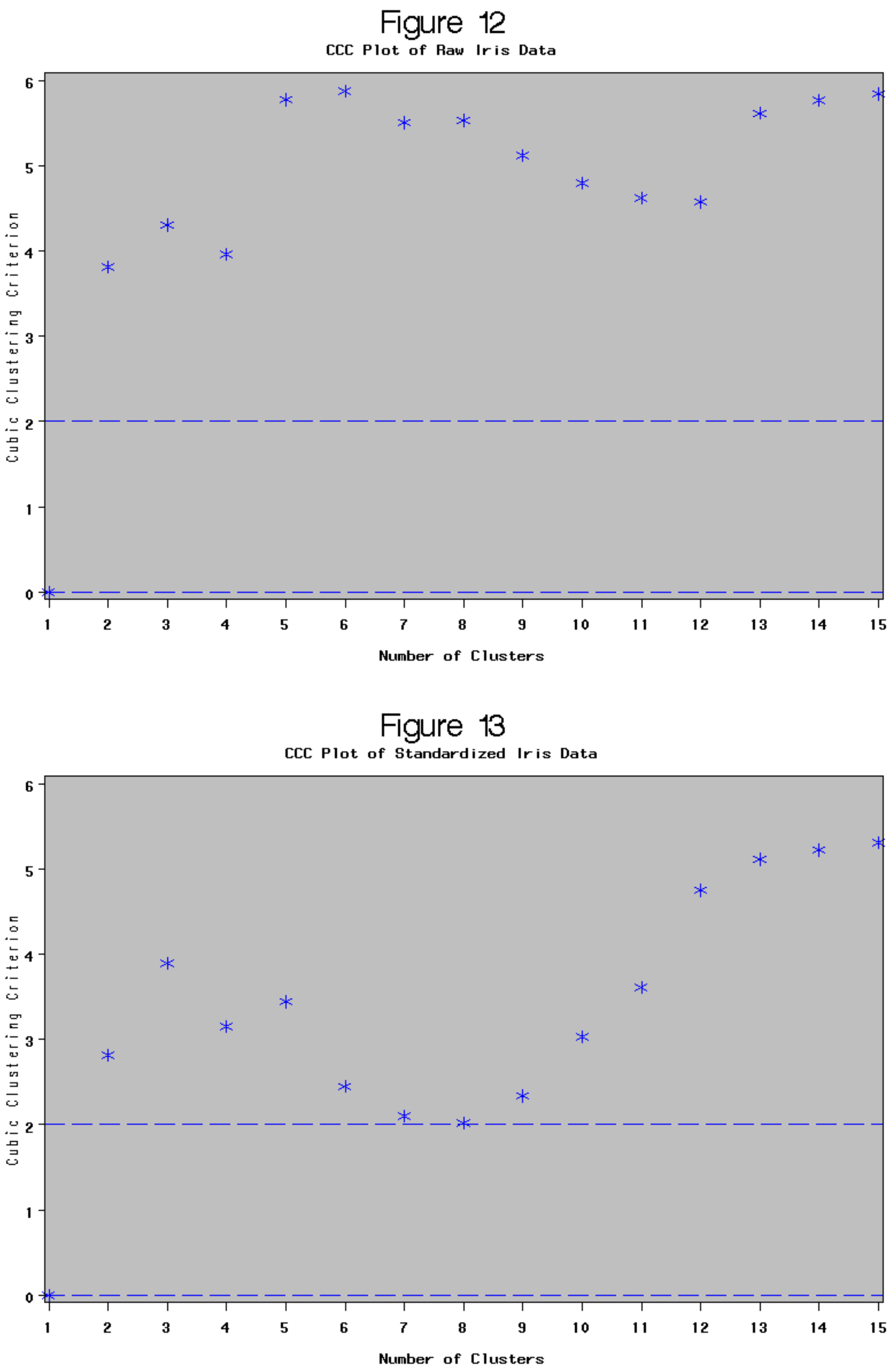


Figure 12 shows a CCC plot for the raw iris data (SAMPSIO.DMAIRIS) from Fisher (1936). There is a local peak at three clusters, the correct value, but also a much higher peak at five or six clusters due to the elliptical nature of the clusters. If the data is standardized, the three-cluster solution becomes apparent as shown in Figure 13.

Figure 15.20 *CCC Plot of Raw Iris Data*



Conclusion

The best way to use the CCC is to plot its value against the number of clusters, ranging from one cluster up to about one-tenth the number of observations. The CCC might not behave well if the average number of observations per cluster is less than ten. The following guidelines should be used for interpreting the CCC:

- Peaks on the plot with the CCC greater than 2 or 3 indicate good clusterings.
- Peaks with the CCC between 0 and 2 indicate possible clusters but should be interpreted cautiously.
- There can be several peaks if the data has a hierarchical structure.
- Very distinct nonhierarchical spherical clusters usually show a sharp rise before the peak followed by a gradual decline.
- Very distinct nonhierarchical elliptical clusters often show a sharp rise to the correct number of clusters followed by a further gradual increase and eventually a gradual decline.
- If all values of the CCC are negative and decreasing for two or more clusters, the distribution is probably unimodal or long-tailed.
- Very negative values of the CCC, say, -30, might be due to outliers. Outliers generally should be removed before clustering.
- If the CCC increases continually as the number of clusters increases, the distribution might be grainy or the data might have been excessively rounded or recorded with just a few digits.

A final and very important warning: neither the CCC nor is an appropriate criterion for clusters that are highly elongated or irregularly shaped. If you do not have prior substantive reasons for expecting compact clusters, use a nonparametric clustering method such as Wong and Lane's (1983) rather than Ward's method or k-means.

References

Anderberg, M. R. 1973. *Cluster Analysis for Applications*. New York: Academic Press.

Duran, B. S. and P. L. Odell. 1974. *Cluster Analysis*, New York: Springer-Verlag.

Edwards, A. W. F. and L. L. Cavalli-Sforza. 1965. "A method for cluster analysis." *Biometrics* 21:362-375.

Englemann, L. and J. A. Hartigan. 1969. "Percentage Points of a Test for Clusters." *Journal of the American Statistical Association* 64:1647-1648.

Everitt, B. S. 1980. *Cluster Analysis. 2nd ed*. London: Heineman Educational Books Ltd.

Fisher, R. A. 1936. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7:179-188.

Gordon, A. D. and J. T. Henderson. 1977. "An Algorithm for Euclidean Sum of Squares Classification." *Biometrics* 33:355-362.

Hartigan, J. A. 1975. *Clustering Algorithms*. New York: John Wiley & Sons.

Hartigan, J. A. 1978. "Asymptotic Distributions for Clustering Criteria." *Annals of Statistics* 6:117-131.

Hubert, L. 1974. "Approximate Evaluation Techniques for the Single-link and Complete-link Hierarchical Clustering Procedures." *Journal of the American Statistical Association* 69:698-704.

Hubert, L. J. and F. B. Baker. 1977. "An Empirical Comparison of Baseline Models for Goodness-of-Fit in r-Diameter Hierarchical Clustering." In *Classification and Clustering*, ed. J. Van Ryzin. New York: Academic Press.

Ling, R. F. 1973. "A Probability Theory of Cluster Analysis." *Journal of the American Statistical Association* 68:159-169.

MacQueen, J. B. 1967. "Some Methods for Classification and Analysis of Multivariate Observations." *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1:281-297.

Mezzich, J. E. and H. Solomon. 1980. *Taxonomy and Behavioral Science*. New York: Academic Press.

Milligan, G. W. and M. C. Cooper. 1983. "An Examination of Procedures for Determining the Number of Clusters in a Data Set" *College of Administrative Science Working Paper Series* 83-51. Columbus: The Ohio State University.

Scott, A. J. and M. J. Symons. 1971. "Clustering Methods Based on Likelihood Ratio Criteria." *Biometrics* 27:387-397.

Ward, J. H. 1963. "Hierarchical grouping to optimize an objective function." *Journal of the American Statistical Association* 58:236-244.

Wolfe, J. H. 1970. "Pattern Clustering by Multivariate Mixture Analysis." *Multivariate Behavioral Research* 5:329-350.

Wolfe, J. H. 1978. "Comparative Cluster Analysis of Patterns of Vocational Interest." *Multivariate Behavioral Research* 13:33-44.

Wong, M. A. 1982. "Asymptotic Properties of Bivariate k-means Clusters." *Communications in Statistics, Theory and Methods* 11:1155-1171.

Wong, M. A. and T. Lane. 1983. "A kth Nearest Neighbor Clustering Procedure." *Journal of the Royal Statistical Society, Series B* 45:362-368.

Chapter 16
Predictive Modeling

Predictive Modeling

Terminology

Predictive modeling tries to find good rules (models) for guessing (predicting) the values of one or more variables in a data set from the values of other variables in the data set. After a good rule has been found, it can be applied to new data sets (scoring) that might or might not contain the variable or variables that are being predicted. The various methods that find prediction rules go by different names in different areas of research, such as regression, function mapping, classification, discriminant analysis, pattern recognition, concept learning, supervised learning, and so on.

In the present context, prediction does not mean forecasting time series. In time series analysis, an entity is observed repeatedly over time, and past values are used to forecast future values. For the predictive modeling methods in SAS Enterprise Miner, each case in a data set represents a different entity, independent of the other cases in the data set. If the entities in question are, for example, customers, then all of the information pertaining to any one customer must be contained in a single case in the data set. If you have a data set in which each customer is described by multiple cases, you must first rearrange the data to place all of the information about any one customer into the same case. It is possible to fit some simple autoregressive models by preprocessing the data using the LAG and DIF functions in the SAS Code node, but SAS Enterprise Miner has no convenient interface for making forecasts.

SAS Enterprise Miner provides a number of tools for predictive modeling. Three of these tools are the Regression node, the Decision Tree node, and the Neural Network node. The methods used in these nodes come from several areas of research, including statistics, pattern recognition, and machine learning. These different areas use different terminology, so before discussing predictive modeling methods, it will be helpful to clarify the terms used in SAS Enterprise Miner. The following list of terms is in logical, not alphabetical order. A more extensive alphabetical glossary can be found in the Glossary.

Synonym
: A word having a meaning similar to but not necessarily identical to that of another word in at least one sense.

Case
: A collection of information about one of numerous entities represented in a data set. Synonyms: observation, record, example, pattern, sample, instance, row, vector, pair, tuple, fact.

Variable
: One of the items of information represented in numeric or character form for each case in a data set. Synonyms: column, feature, attribute, coordinate, measurement.

Target
: A variable whose value is known in some currently available data, but will be unknown in some future/fresh/operational data set. You want to be able to predict the values of the target variable or variables from other known variables. Synonyms: dependent variable, response, observed values, training values, desired output, correct output, outcome.

Input
: A variable used to predict the value of the target variable or variables. Synonyms: independent variable, predictor, regressor, explanatory variable, carrier, factor, covariate.

Output
: A variable computed from the inputs as a prediction of the value of the target variable or variables Synonyms: predicted value, estimate, y-hat.

Model
: A class of formulas or algorithms used to compute outputs from inputs. A statistical model also includes information about the conditional distribution of the targets given the inputs. See also trained model below. Synonyms: architecture (for neural nets), classifier, expert, equation, function.

Weights
: Numeric values used in a model that are usually unknown or unspecified prior to the analysis. Synonyms: estimated parameters, estimates, regression coefficients, standardized regression coefficients, betas.

Case Weight
: A nonnegative numeric variable that indicates the importance of each case. There are three types of case weights: frequencies, sampling weights, and variance weights. SAS Enterprise Miner supports only frequencies.

Parameters
: The true or optimal values of the weights or other quantities (such as standard deviations) in a model.

Training

The process of computing good values for the weights in a model, or, for tree-based models, choosing good split variables and split values. Synonyms: estimation, fitting, learning, adaptation, induction, growing (trees, that is).

Trained Model

A specific formula or algorithm for computing outputs from inputs, with all weights or parameter estimates in the model chosen via a training algorithm from a class of such formulas or algorithms designated by the model. Synonyms: fitted model.

Generalization

The ability of a model to compute good outputs from input data not used during training. Synonyms: interpolation and extrapolation, prediction.

Population

The set of all cases that you want to be able to generalize to. The data to be analyzed in data mining are usually a subset of the population.

Sample

A subset of the population that is available for analysis.

Noise

Unpredictable variation, usually in a target variable. For example, if two cases have identical input values but different target values, the variation in those different target values is not predictable from any model using only those inputs, hence that variation is noise. Noise is often assumed to be random. In that case, it is inherently unpredictable. Since noise prevents target values from being accurately predicted, the distribution of the noise can be estimated statistically given enough data. Synonym: error.

Signal

Predictable variation in a target variable. It is often assumed that target values are the sum of signal and noise, where the signal is a function of the input variables. Synonyms: Function, systematic component.

Training Data

Data containing input and target values, used for training to estimate weights or other parameters. Synonyms: Training set, design set.

Test Data

Data containing input and target values, not used during training in any way, but instead used to estimate generalization error. Synonyms: Test data set (often confused with validation data).

Validation Data

Data containing input and target values, used indirectly during training for model selection or early stopping. Synonyms: Validation set (often confused with test data).

Scoring

Applying a trained model to data to compute outputs. Synonyms: running (for neural nets), simulating (for neural nets), filtering (for trees), interpolating or extrapolating.

Interpolation

Scoring or generalization for cases on or within the convex hull of the training set in the space of the input variables.

Extrapolation

Scoring or generalization for cases outside the convex hull of the training set in the space of the input variables.

Operational Data

Data to be scored in a practical application, containing inputs but not target values. Scoring operational data is the main purpose of training models in data mining. Synonyms: scoring data.

Categorical Variable

A variable which for all practical purposes has only a limited number of possible values. Synonyms: class variable, label.

Category

One of the possible values of a categorical variable. Synonyms: class, level, label.

Class Variable

In data mining, pattern recognition, knowledge discovery, neural networks, and so on, a class variable means a categorical target variable, and classification means assigning cases to categories of a target variable. In traditional SAS procedures, class variable means simply categorical variable, either an input or a target.

Measurement

The process of assigning numbers to things such that the properties of the numbers reflect some attribute of the things.

Measurement Level

One of several ways in which properties of numbers can reflect attributes of things. The most common measurement levels are nominal, ordinal, interval, log-interval, ratio, and absolute. For details, see the Measurement Theory FAQ at **ftp://ftp.sas.com/pub/neural/measurement.html**.

Nominal Variable

A numeric or character categorical variable in which the categories are unordered, and the category values convey no additional information beyond category membership.

Ordinal Variable

A numeric or character categorical variable in which the categories are ordered, but the category values convey no additional information beyond membership and order. In particular, the number of levels between two categories is not informative, and for numeric variables, the difference between category values is not informative. The results of an analysis that includes ordinal variables will typically be unchanged if you replace all the values of an ordinal variable by different numeric or character values as long as the order is maintained, although some algorithms might use the numeric values for initialization. SAS Enterprise Miner provides no explicit support for continuous ordinal variables, although some procedures in other SAS products do so, such as TRANSREG and PRINQUAL.

Interval Variable

A numeric variable for which differences of values are informative.

Ratio Variable

A numeric variable for which ratios of values are informative. In SAS Enterprise Miner, ratio and higher-level variables are not generally distinguished from interval variables, since the analytical methods are the same. However, ratio measurements are required for some computations in model assessment, such as profit and ROI measures.

Binary Variable

A variable that takes only two distinct values. A binary variable can be legitimately treated as nominal, ordinal, interval, or sometimes ratio.

Common Features of Predictive Modeling Nodes

Table of Common Features

The predictive modeling nodes are designed to share many common features. The following table lists some features that are broadly applicable to predictive modeling and indicates which nodes have the features. Decision options, output data sets, and score variables are described in subsequent sections of this chapter.

Table 16.1 Features of Predictive Modeling Nodes

	Neural Network	Regression	Decision Tree
Input Data Sets:			
Training	Yes	Yes	Yes
Validation	Yes	Yes	Yes
Test	Yes	Yes	Yes
Scoring			
Input Variables			
Nominal	Yes	Yes	Yes
Ordinal	Yes	No#	Yes
Interval	Yes	Yes	Yes
Other Variable Roles:			
Frequency	Yes	Yes	Yes
Sampling Weight	No*	No*	No*
Variance Weight	No	No	No
Cost	Yes	Yes	Yes
Decision Options:			
Prior Probabilities	Yes	Yes	Yes
Profit or Loss Matrix	Yes	Yes	Yes
Output Data Sets:			
Scores	Yes	Yes	Yes

	Neural Network	Regression	Decision Tree
Model (weights, trees)	Yes	Yes	Yes
Fit Statistics	Yes	Yes	Yes
Profit or Loss Summaries	Yes	Yes	Yes
Score Variables:			
Output (predicted value, posterior probability)	Yes	Yes	Yes
Residual	Yes	Yes	Yes
Classify (from, into)	Yes	Yes	Yes
Expected Profit or Loss	Yes	Yes	Yes
Profit or Loss Computed from Target	Yes	Yes	Yes
Decision	Yes	Yes	Yes
Other Features:			
Interactive Training	Yes	No	Yes
Save and reuse models	Yes	Yes	Yes
Apply model with missing inputs	No	No	Yes
DATA step code for scoring	Yes	Yes	Yes

Note: # – The Regression node treats ordinal inputs as nominal; it does not preserve the ordering of the levels.

Note: * – Planned for a future release.

Setting the “Use” Column in SAS Enterprise Miner Node Variable Tables

The Properties Panel for every SAS Enterprise Miner node contains a **Variables** property. You use this property to view and specify the status and role of variables that you use in each node of your data mining analysis. To the right of the **Variables** property for each node is an ellipsis button [...] that opens the SAS Enterprise Miner

Variables Editor for the selected tool. You use the tool's variable table editor to specify which variables to include or exclude, and for included variables, the analytical role of the variable (such as input, target, frequency, and so on) in the mining analysis..

In the case of modeling nodes, the **Use** column in the Variables table means "use as input", and the possible values for the **Use** column are Yes, No, and Default. The meanings are as follows:

- **Yes** is the default **Use** value for all variables with a role of "frequency". **Yes** is also the default **Use** value for the *first* variable in the variables table that has a role of "Target".
- **No** is the default **Use** value for all target variables in the table that follow the first target variable.
- **Default** is available only as a **Use** column value if the variables table has only one target variable. For variable tables that have only one target variable, the meaning of the **Default** setting for the **Use** column depends on the variable's role. If a variable has a role of **Input**, SAS Enterprise Miner treats the **Use** setting of **Default** as Yes. If a variable has a role of **Rejected**, then SAS Enterprise Miner treats the **Use** setting of **Default** as No.

Categorical Variables

Categories for nominal and ordinal variables are defined by the normalized, formatted values of the variable. If you have not explicitly assigned a format to a variable, the default format for a numeric variable is BEST12., and the default format for a character variable is $w., where w is the length of the variable. The formatted value is normalized by:

1. Removing leading blanks
2. Truncating to 32 characters
3. Changing lowercase letters to uppercase.

Hence, if two values of a variable differ only in the number of leading blanks and in the case of their letters, they will be assigned to the same category. Also, if two values differ only past the first 32 characters (after left-justification), they will be assigned to the same category.

Dummy variables are generated for categorical variables in the Regression and Neural Network nodes. If a categorical variable has c categories, the number of dummy variables will be either c or c-1, depending on the role of the variable and what options are specified. The computer time and memory requirements for analyzing a categorical variable with c categories are the same as the requirements for analyzing c or c-1 interval-level variables for the Regression and Neural Network nodes.

When a categorical variable appears in two or more data sets used in the same modeling node, such as the training set (prior to DMDB processing), validation set, and decision data set, the variable is not required to have the same type and length in each data set. For example, a variable named TEMPERAT could be numeric in the training set with values such as 98.6, while a variable by the same name in the validation set could be character with values such as "98.6". As long as the normalized, formatted values from the two data sets agree, the values of the two variables will be matched correctly. In the Neural Network node only, a categorical variable that appears in two or more data sets must have the same formatted length in each data set.

Predicted Values and Posterior Probabilities

For an interval target variable, by default the modeling nodes try to predict the conditional mean of the target given the values of the input variables. The Neural Network node also provides robust error functions that can be used to predict approximately the conditional median or mode of the target.

For an interval target variable, by default the modeling nodes try to predict the conditional mean of the target given the values of the input variables. The Neural Network node also provides robust error functions that can be used to predict approximately the conditional median or mode of the target.

You can also specify a profit or loss matrix to classify cases according to the business consequences of the decision. (See the section below on Decisions.) The robust error functions in the Neural Network node can be used to output the approximately most probable class.

When comparing predictive models, it is essential to compare all models using the same cases. If a case is omitted from scoring for one model but not from another (for example, because of missing input variables) you get invalid, "apples-and-oranges" model comparisons. Therefore, SAS Enterprise Miner modeling nodes compute predictions for all cases, even for cases where the model is inapplicable because of missing inputs or other reasons (except, of course, when there are no valid target values).

For cases where the model cannot be applied, the modeling nodes output the unconditional mean (the mean for all cases used for training) for interval targets, or the prior probabilities for categorical targets (see the section below on Prior Probabilities). If you do not specify prior probabilities, implicit priors are used, which are the proportions of the classes among all cases used for training. A variable named _WARN_ in the scored data set indicates why the model could not be applied. If you have lots of cases with missing inputs, you should either use the Decision Tree node for modeling, or use the Impute node to impute missing values before using the Regression or Neural Network nodes.

The Frequency Variable and Weighted Estimation

All of the SAS Enterprise Miner modeling nodes enable you to specify a frequency variable. Typically, the values of the frequency variable are nonnegative integers. The data are treated as if each case were replicated as many times as the value of the frequency variable.

Unlike most SAS procedures, the modeling nodes in SAS Enterprise Miner accept values for a frequency variable that are not integers without truncating the fractional part. Thus, you can use a frequency variable to perform weighted analyses.

However, SAS Enterprise Miner does not provide explicit support for sampling weights, noise-variance weights, or other analyses where the weight variable does not represent the frequency of occurrence of each case. If the frequency variable represents sampling weights or noise-variance weights, the point estimates of regression coefficients and neural network weights will be valid. But if the frequency variable does not represent actual frequencies, then standard errors, significance tests, and statistics such as MSE, AIC, and SBC might be invalid.

If you want to do weighted estimation under the usual assumption for weighted least squares that the weights are inversely proportional to the noise variance (error variance) of the target variable, then you can obtain statistically correct results by specifying frequency values that add up to the sample size.

If you want to use sampling weights that are inversely proportional to the sampling probability of each case, you can get approximate estimates for MSE and related statistics in the Regression and Neural Network nodes by specifying frequencies that add up to the effective sample size. A pessimistic approximation to the effective sample size is provided by

$$\frac{[\sum W(i)]^2}{\sum W(i)^2}$$

where W(i) is a sampling weight for case i. This approximation will not work properly with the Decision Tree node.

Differences among Predictive Modeling Nodes

The Regression node, the Tree node, and the Neural Network node can all learn complex models from data, but they have different ways of representing complexity in their models. Choosing a model of appropriate complexity is important for making accurate predictions, as discussed in the section below on Generalization. Simple models are best for learning simple functions of the data (as long as the model is correct, of course), but complex models are required for learning complex functions. With all data mining models, one way to increase the complexity of a model is to add input variables. Other ways to increase complexity depend on the type of model:

- In regression models, you can add interactions and polynomial terms.
- In neural networks, you can add hidden units.
- In tree-based models, you can grow a larger tree.

One fundamental difference between tree-based models and both regression and neural net models is that tree-based models learn step functions, whereas the other models learn continuous functions. If you expect the function to be discontinuous, a tree-based model is a good way to start. However, given enough data and training time, neural networks can approximate discontinuities arbitrarily well. Polynomial regression models are not good at learning discontinuities. To model discontinuities using regression, you need to know where the discontinuities occur and construct dummy variables to indicate the discontinuities before fitting the regression model.

For both regression and neural networks, the simplest models are linear functions of the inputs. Hence regression and neural nets are both good for learning linear functions. Tree-based models require many branches to approximate linear functions accurately.

When there are many inputs, learning is inherently difficult because of the curse of dimensionality (see the Neural Network FAQ at the URL **`ftp://ftp.sas.com/pub/neural/FAQ2.html#A_curse`**.

To learn general nonlinear functions, all modeling methods require a degree of complexity that grows exponentially with the number of inputs. That is, as the number of inputs increases, the number of interactions and polynomial terms required in a regression model grows exponentially, the number of hidden units required in a neural network grows exponentially, and the number of branches required in a tree grows exponentially. The amount of data and the amount of training time required to learn such models also grow exponentially.

Fortunately, in most practical applications with a large number of inputs, most of the inputs are irrelevant or redundant, and the curse of dimensionality can be circumvented. Tree-based models are especially good at ignoring irrelevant inputs, since trees often use a relatively small number of inputs even when the total number of inputs is large.

If the function to be learned is linear, stepwise regression is good for choosing a small number out of a large set of inputs. For nonlinear models with many inputs, regression is not a good choice unless you have prior knowledge of which interactions and polynomial terms to include in the model. Among various neural net architectures, multilayer perceptrons and normalized radial basis function (RBF) networks are good at ignoring irrelevant inputs and finding relevant subspaces of the input space, but ordinary radial basis function networks should be used only when all or most of the inputs are relevant.

All of the modeling nodes can process redundant inputs effectively. Adding redundant inputs has little effect on the effective dimensionality of the data; hence the curse of dimensionality does not apply. When there are redundant inputs, the training cases lie close to some (possibly nonlinear) subspace. If this subspace is linear, redundancy is called multicollinearity.

In statistical theory, it is well-known that redundancy causes parameter estimates (weights) to be unstable. That is, different parameter estimates can produce similar predictions. But if the purpose of the analysis is prediction, unstable parameter estimates are not necessarily a problem. If the same redundancy applies to the test cases as to the training cases, the model needs to produce accurate outputs only near the subspace occupied by the data, and stable parameter estimates are not needed for accurate prediction. However, if the test cases do not follow the same pattern of redundancy as the training cases, generalization will require extrapolation and will rarely work well.

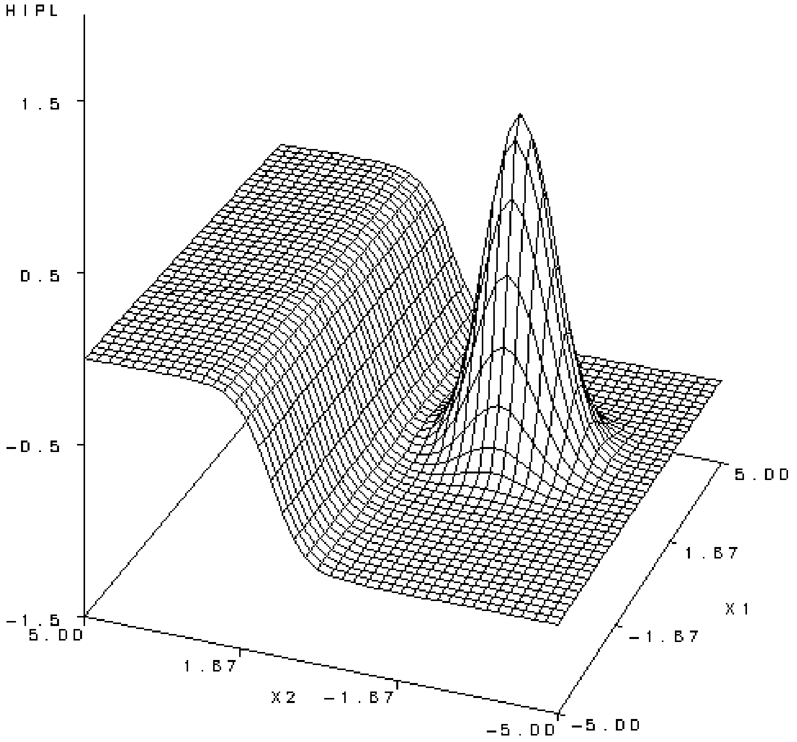
If extrapolation is required, decision tree-based models are safest, because trees choose just one of several redundant inputs and produce constant predictions outside the range of the training data. Stepwise linear regression or linear-logistic regression are the next safest methods for extrapolation if a large singularity criterion is used to make sure that the parameter estimates do not become excessively unstable. Polynomial regression is usually a bad choice for extrapolation, because the predictions will often increase or decrease rapidly outside the range of the training data. Neural networks are also dangerous for extrapolation if the weights are large. Weight decay and early stopping can be used to discourage large weights. Normalized radial basis function (RBF) networks are the safest type of neural net architecture for extrapolation, since the range of predictions will never exceed the range of the hidden-to-output weights.

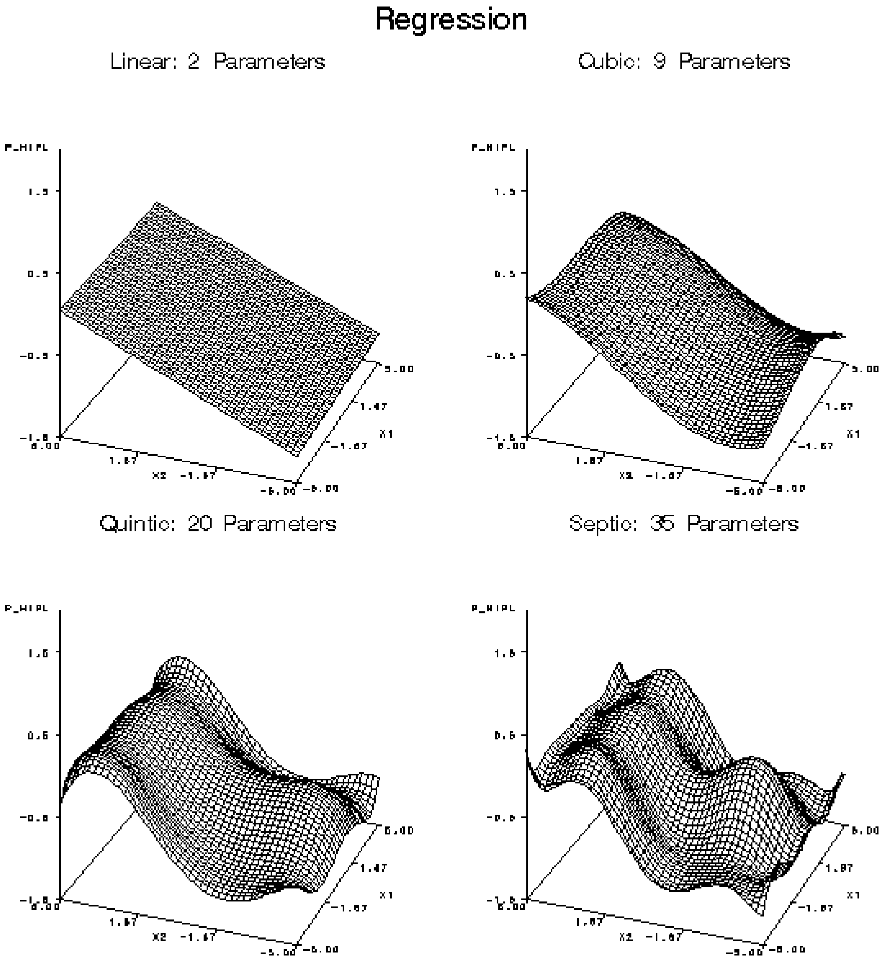
The Decision Tree node can use cases with missing inputs for training and provides several ways of making predictions from cases with missing inputs. The Regression and Neural Network nodes cannot use cases with missing inputs for training; predictions are based on the unconditional mean or prior probabilities. (See Predicted Values and Posterior Probabilities.)

The Neural Network node can model two or more target variables in the same network. Having multiple targets in the network can be an advantage when there are features common to all the targets. Otherwise, it is more efficient to train separate networks. The Regression node and the Decision Tree node process only one target at a time, but the Start Group node can be used to handle multiple targets.

The following figures illustrate the types of approximation error that commonly occur with each of the modeling nodes. The noise-free data come from the hill-and-plateau function, which was chosen because it is difficult for typical neural networks to learn. Given sufficient model complexity, all of the modeling nodes can, of course, learn the data accurately. These examples show what happens with insufficient model complexity. The cases in the training set lie on a 21 by 21 grid. Those in the test data set are on a 41 by 41 grid.

Hill and Plateau Function: Test Data





Regression

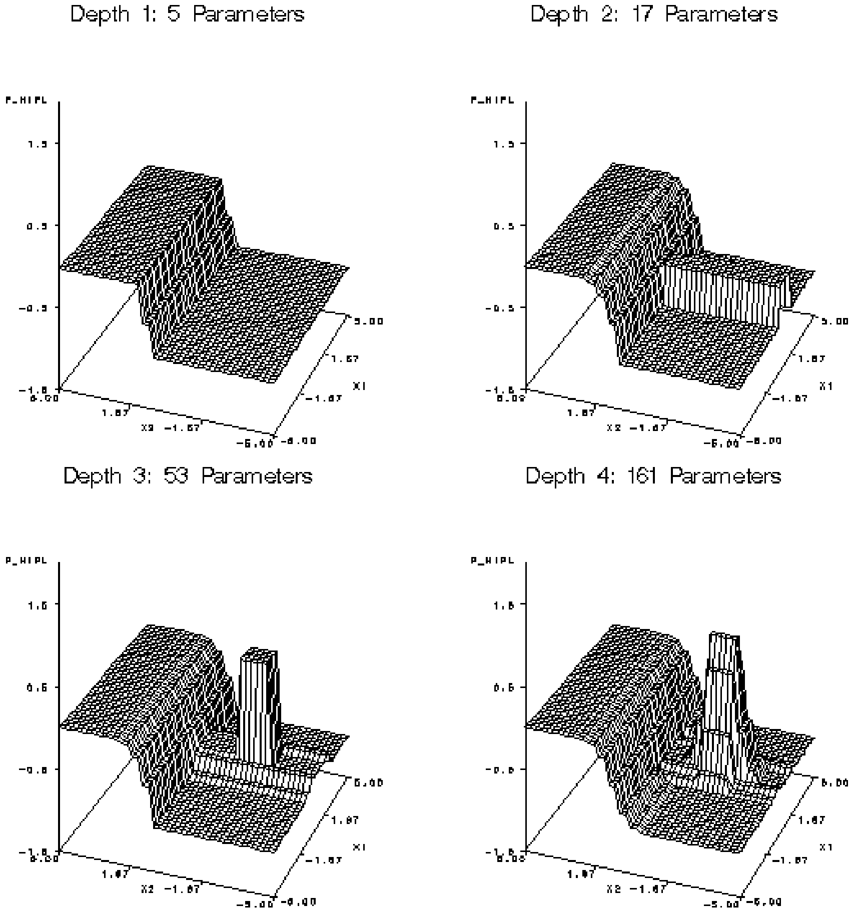
Linear: 2 Parameters

Cubic: 9 Parameters

Quintic: 20 Parameters

Septic: 35 Parameters

Empirical Decision Tree with 3–Way Branches

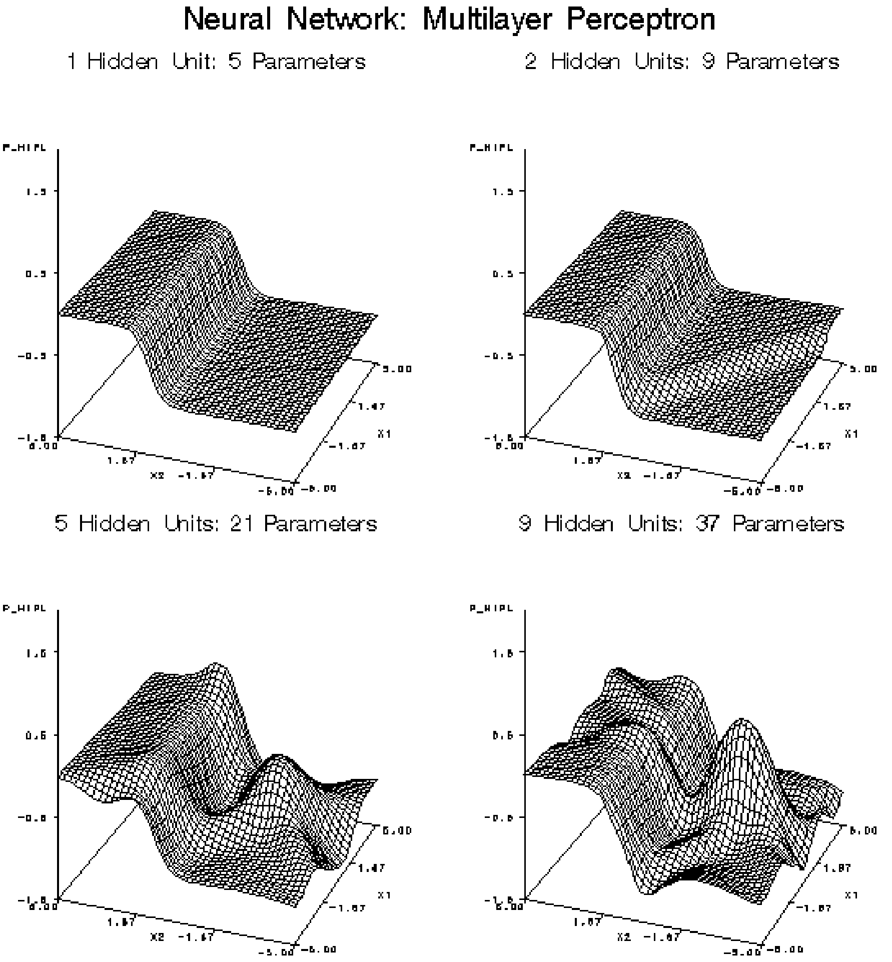


Depth 1: 5 Parameters

Depth 2: 17 Parameters

Depth 3: 53 Parameters

Depth 4: 161 Parameters



Neural Network: Multilayer Perceptron

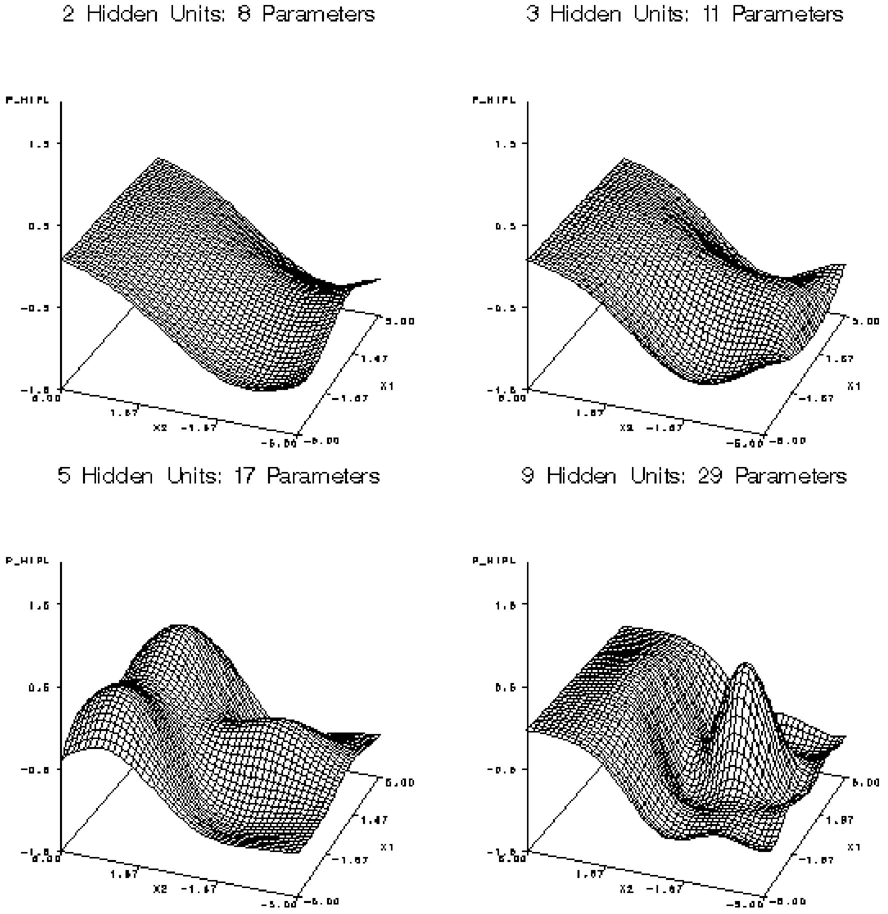
1 Hidden Unit: 5 Parameters

2 Hidden Units: 9 Parameters

5 Hidden Units: 21 Parameters

9 Hidden Units: 37 Parameters

Neural Network: Ordinary RBF Network with Equal Widths and Heights

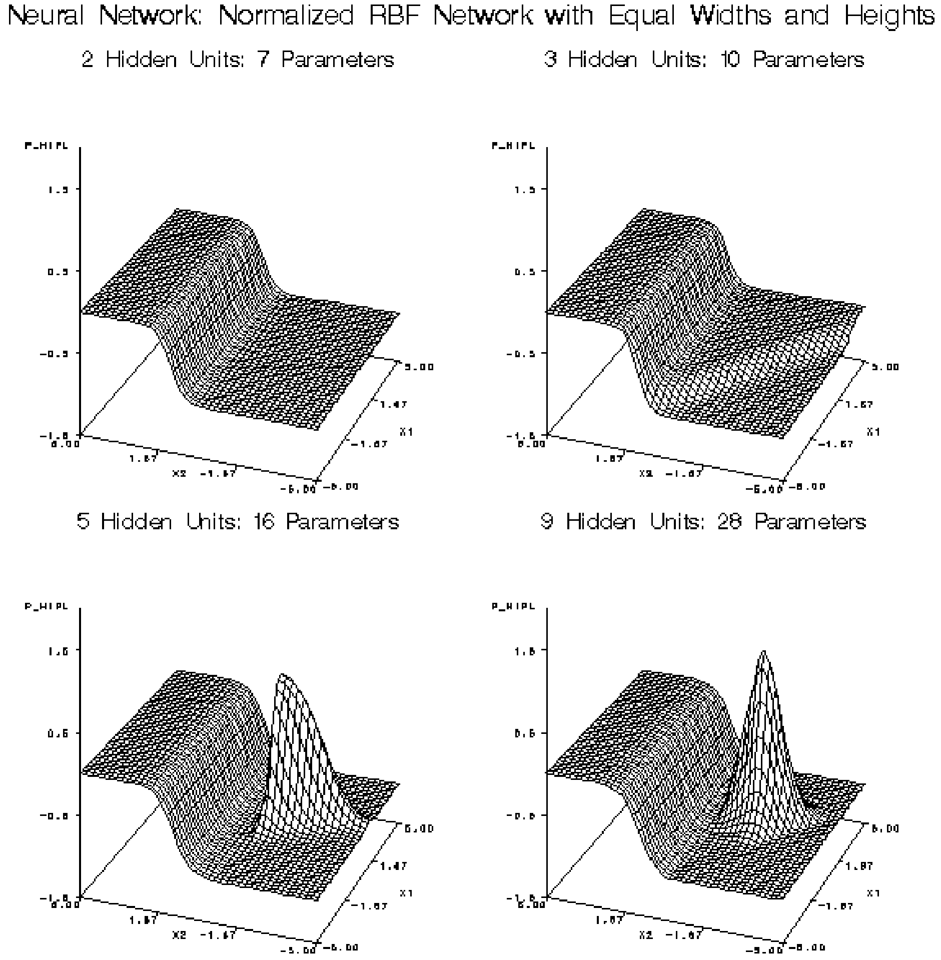


2 Hidden Units: 8 Parameters

3 Hidden Units: 11 Parameters

5 Hidden Units: 17 Parameters

9 Hidden Units: 29 Parameters



Computer Resources

The computer time and memory required for an analysis depend on the number of cases, the number of variables, the complexity of the model, and the training algorithm. For many modeling methods, there is a trade-off between time and memory.

For all modeling nodes, memory is required for the operating system, SAS supervisor, and the SAS Enterprise Miner diagram and programs, resulting in an overhead of about 20 to 30 megabytes.

Let:

N

be the number of cases.

V

be the number of input variables.

I

be the number of input terms or units, including dummy variables, intercepts, interactions, and polynomials.

W

be the number of weights in a neural network.

O

be the number of output units.

D

be the average depth of a tree.

R

be the number of times the training data are read in logistic regression or neural nets, which depends on the training technique, the termination criteria, the model, and the data. R is typically much larger for neural nets than for logistic regression. In regard to training techniques, R is usually smallest for Newton-Raphson or Levenberg-Marquardt, larger for quasi-Newton, and still larger for conjugate gradients.

S

be the number of steps in stepwise regression, or 1 if stepwise regression is not used.

For the Decision Tree node, the minimum additional memory required for an analysis is about 8N bytes. Training will be considerably faster if there is enough RAM to hold the entire data set, which is about 8N(V+1) bytes. If the data will not fit in memory, they must be stored in a utility file. Memory is also required to hold summary statistics for a node, such as means or a contingency table, but this amount is usually much smaller than the amount required for the data.

For the Regression node, the memory required depends on the type of model and on the training technique. For linear regression, memory usage is dominated by the SSCP matrix, which requires 8I2 bytes. For logistic regression, memory usage depends on the training technique as documented in the SAS/OR Technical Report: The NLP Procedure, ranging from about 40I bytes for the conjugate gradient technique to about 8I2 bytes for the Newton-Raphson technique.

For the Neural Network node, memory usage depends on the training technique as documented in the SAS/OR Technical Report: The NLP Procedure. About 40W bytes are needed for the conjugate gradient technique, and $4W^2$ bytes are needed for the quasi-Newton and Levenberg-Marquardt techniques. For a network with biases and H hidden units in one layer, W = (I+1)H + (H+1)O. For both logistic regression and neural networks, the conjugate gradient technique, which requires the least memory, must usually read the training data many more times than the Newton-Raphson and Levenberg-Marquardt techniques.

Assuming that the number of training cases is greater than the number of inputs or weights, the time required for training is approximately proportional to:

NI^2

for linear regression.

SRNI

for logistic regression using conjugate gradients.

$SRNI^2$

for logistic regression using quasi-Newton or Newton-Raphson. Note that R is usually considerably less for these techniques than for conjugate gradients.

DNI

for decision tree-based models.

RNW

for neural nets using conjugate gradients.

RNW^2

for neural nets using quasi-Newton or Levenberg-Marquardt. Note that R is usually considerably less for these techniques than for conjugate gradients.

Prior Probabilities

For a categorical target variable, each modeling node can estimate posterior probabilities for each class, which are defined as the conditional probabilities of the classes given the input variables. By default, the posterior probabilities are based on implicit prior probabilities that are proportional to the frequencies of the classes in the training set. You can specify different prior probabilities via the Target Profile using the Prior Probabilities tab. (See the Target Profile chapter.) Also, given a previously scored data set containing posterior probabilities, you can compute new posterior probabilities for different priors by using the DECIDE procedure, which reads the prior probabilities from a decision data set.

Prior probabilities should be specified when the sample proportions of the classes in the training set differ substantially from the proportions in the operational data to be scored, either through sampling variation or deliberate bias. For example, when the purpose of the analysis is to detect a rare class, it is a common practice to use a training set in which the rare class is over represented. If no prior probabilities are used, the estimated posterior probabilities for the rare class will be too high. If you specify correct priors, the posterior probabilities will be correctly adjusted no matter what the proportions in the training set are. For more information, see Detecting Rare Classes.

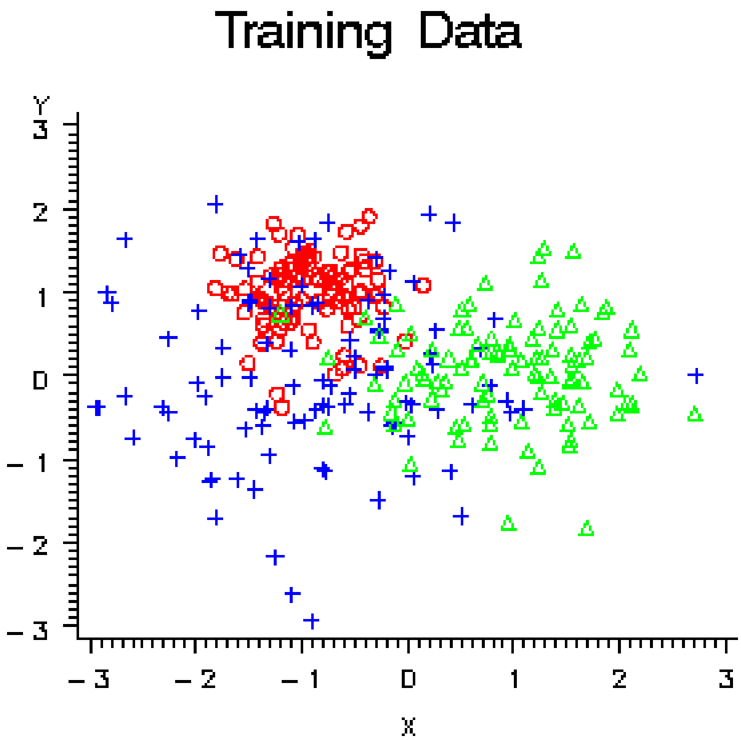
Increasing the prior probability of a class increases the posterior probability of the class, moving the classification boundary for that class so that more cases are classified into the class. Changing the prior will have a more noticeable effect if the original posterior is near 0.5 than if it is near zero or one.

For linear logistic regression and linear normal-theory discriminant analysis, classification boundaries are hyperplanes; increasing the prior for a class moves the hyperplanes for that class farther from the class mean, and decreasing the prior moves the hyperplanes closer to the class mean, but changing the priors does not change the angles of the hyperplanes.

For quadratic logistic regression and quadratic normal-theory discriminant analysis, classification boundaries are quadratic hypersurfaces; increasing the prior for a class moves the boundaries for that class farther from the class mean, and decreasing the prior moves the boundaries closer to the class mean, but changing the priors does not change the shapes of the quadratic surfaces.

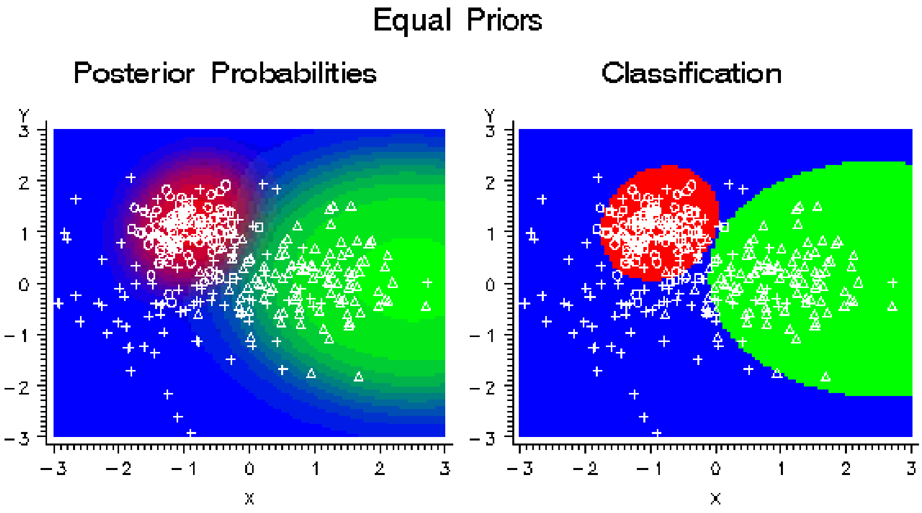
To show the effect of changing prior probabilities, the data in the following figure were generated to have three classes, shown as red circles, blue crosses, and green triangles. Each class has 100 training cases with a bivariate normal distribution.

Figure 16.1 *Training Data Plot with Three Classes*



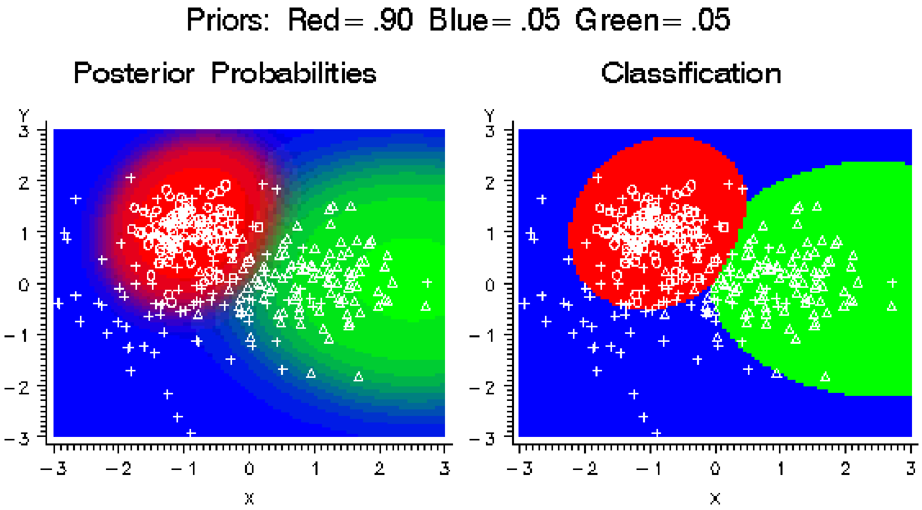
These training data were used to fit a quadratic logistic regression model using the Neural Network engine. Since each class has the same number of training cases, the implicit prior probabilities are equal. In the following figure, the plot on the left shows color-coded posterior probabilities for each class. Bright red areas have a posterior probability near 1.0 for the red circle class, bright blue areas have a posterior probability near 1.0 for the blue cross class, and bright green areas have a posterior probability near 1.0 for the green triangle class. The plot on the right shows the classification results as red, blue, and green regions.

Figure 16.2 *Classification Plots with Equal Priors*



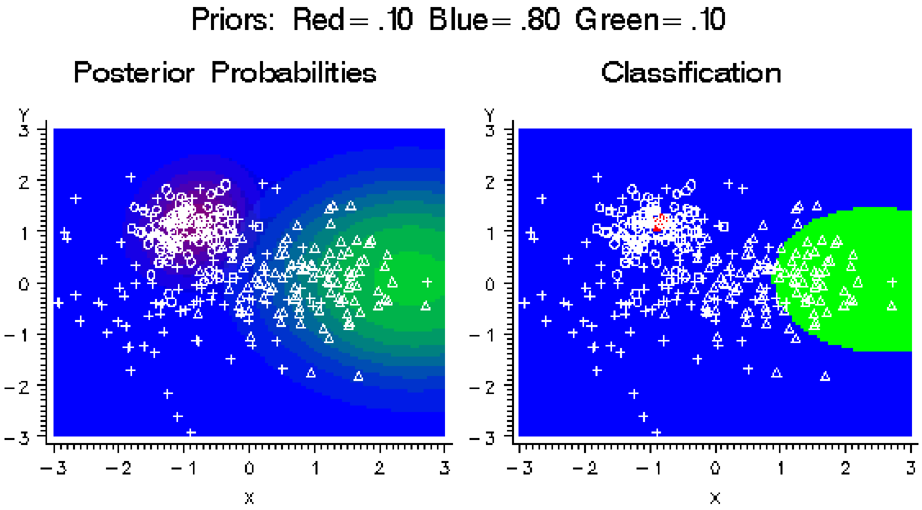
If the prior probability for the red class is increased, the red areas in the plots expand in size as shown in the following figure. The red class has a small variance, so the effect is not widespread. Since the priors for the blue and green classes are still equal, the boundary between blue and green has not changed.

Figure 16.3 *Classification Plots with Priors: Red = .90 Blue = .05 Green = .05*



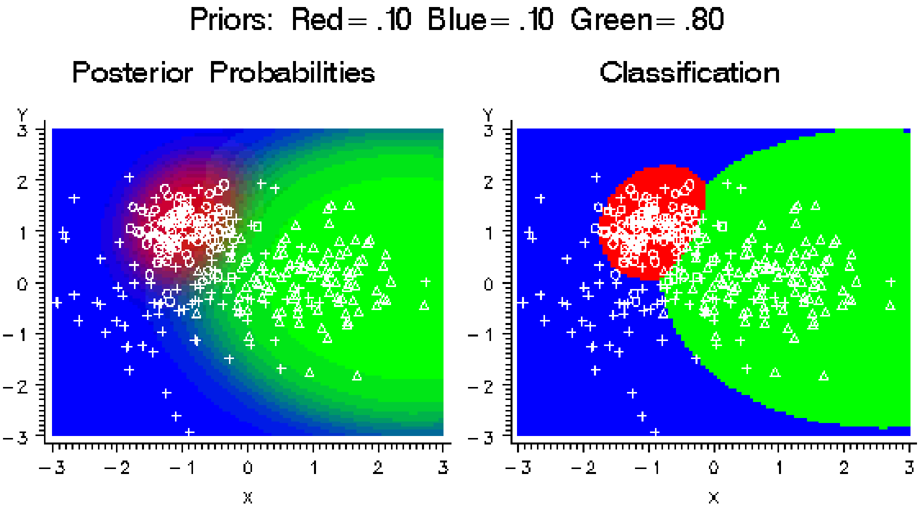
If the prior probability for the blue class is increased, the blue areas in the plots expand in size as shown in the following figure. The blue class has a large variance and has a substantial density extending beyond the high-density red region, so increasing the blue prior causes the red areas to contract dramatically.

Figure 16.4 *Classification Plots with Priors: Red = .10 Blue = .80 Green = .10*



If the prior probability for the green class is increased, the green areas in the plots expand as shown in the following figure.

Figure 16.5 *Classification Plots with Priors: Red = .10 Blue = .10 Green = .80*



In the literature on data mining, statistics, pattern recognition, and so on, prior probabilities are used for a variety of purposes that are sometimes confusing. In SAS Enterprise Miner, however, the nodes are designed to use prior probabilities in a simple, unambiguous way:

- Prior probabilities are assumed to be estimates of the true proportions of the classes in the operational data to be scored.
- Prior probabilities are not used by default for parameter estimation. This enables you to manipulate the class proportions in the training set by using nonproportional sampling, or by using a frequency variable in the method of your choice.
- If you specify prior probabilities, the posterior probabilities computed by the modeling nodes are always adjusted for the priors.
- If you specify prior probabilities, the profit and loss summary statistics are always adjusted for priors and therefore provide valid model comparisons, assuming that you specify valid decision consequences. (See the following section on Decisions.)

If you do not explicitly specify prior probabilities (or if you specify None for prior probabilities in the target profile), no adjustments for priors are performed by any nodes.

Posterior probabilities are adjusted for priors as follows. Let:

t

be an index for target values (classes)

i>

be an index for cases

OldPrior(t)

be the old prior probability or implicit prior probability for target t

OldPost(i,t)

be the posterior probability based on OldPrior(t)

Prior(t)

be the new prior probability desired for target t

Post(i,t)

be the posterior probability based on Prior(t)

Then:

$$Post(i,t) = \frac{OldPost(i,t)Prior(t) / OldPrior(t)}{\sum_{j} OldPost(i,j)Prior(j) / OldPrior(j)}$$

For classification, each case i is assigned to the class with the greatest posterior probability, that is, the class t for which Post(i,t) is maximized.

Prior probabilities have no effect on estimating parameters in the Regression node, on learning weights in the Neural Network node, or, by default, on growing trees in the Tree node. Prior probabilities do affect classification and decision processing for each case. Hence, if you specify the appropriate options for each node, prior probabilities can affect the choice of models in the Regression node, early stopping in the Neural Network node, and pruning in the Tree node.

Prior probabilities are also used to adjust the relative contribution of each class when computing the total and average profit and loss as described in the section below on Decisions. The adjustment of total and average profit and loss is distinct from the adjustment of posterior probabilities. The latter is used to obtain correct posteriors for individual cases, whereas the former is used to obtain correct summary statistics for the sample. The adjustment of total and average profit and loss is done only if you explicitly specify prior probabilities; the adjustment is not done when the implicit priors based on the training set proportions are used.

Note that the fit statistics such as misclassification rate and mean squared error are not adjusted for prior probabilities. These fit statistics are intended to provide information about the training process under the assumption that you have provided an appropriate training set with appropriate frequencies, hence adjustment for prior probabilities could present a misleading picture of the training results. The profit and loss summary statistics are intended to be used for model selection, and to assess decisions that are made using the model under the assumption that you have provided the appropriate prior probabilities and decision values. Therefore, adjustment for prior probabilities is required for data sets that lack representative class proportions. For more details, see Decisions .

If you specify priors explicitly, SAS Enterprise Miner assumes that the priors that you specify represent the true operational prior probabilities and adjusts the profit and loss summary statistics accordingly. Therefore:

- If you are using profit and loss summary statistics, the class proportions in the validation and test sets need not be the same as in the operational data as long as your priors are correct for the operational data.
- You can use training sets based on different sampling methods or with differently weighted classes (using a frequency variable), and as long as you use the same explicitly specified prior probabilities, the profit and loss summary statistics for the training, validation, and test sets will be comparable across all of those different training conditions.
- If you fit two or more models with different specified priors, the profit and loss summary statistics will not be comparable and should not be used for model selection, since the different summary statistics apply to different operational data sets.

If you do not specify priors, SAS Enterprise Miner assumes that the validation and test sets are representative of the operational data, hence the profit and loss summary statistics are not adjusted for the implicit priors based on the training set proportions. Therefore:

- If the validation and test sets are indeed representative of the operational data, then regardless of whether you specify priors, you can use training sets based on different sampling methods or with differently weighted classes (using a frequency variable), and the profit and loss summary statistics for the validation and test sets will be comparable across all of those different training conditions.
- If the validation and test sets are not representative of the operational data, then the validation statistics might not provide valid model comparisons, and the test-set statistics might not provide valid estimates of generalization accuracy.

If a class has both an old prior and a new prior of zero, then it is omitted from the computations. If a class has a zero old prior, you might not assign it a positive new prior, since that would cause a division by zero. Prior probabilities might not be missing or negative. They must sum to a positive value. If the priors do not sum to one, they are automatically adjusted to do so by dividing each prior by the sum of the priors. A class might have a zero prior probability, but if you use PROC DECIDE to update posterior probabilities, any case having a nonzero posterior corresponding to a zero prior will cause the results for that case to be set to missing values.

To summarize, prior probabilities do not affect:

- Estimating parameters in the Regression node.
- Learning weights in the Neural Network node.
- Growing (as opposed to pruning) trees in the Decision Tree node unless you configure the property Use Prior Probability in Split Search.
- Residuals, which are based on posteriors before adjustment for priors, except in the Decision Tree node if you choose to use prior probabilities in the split search.
- Error functions such as deviance or likelihood, except in the Decision Tree node if you choose to use prior probabilities in the split search.
- Fit statistics such as MSE based on residuals or error functions, except in the Decision Tree node if you choose to use prior probabilities in the split search.

Prior probabilities do affect:

- Posterior probabilities
- Classification
- Decisions
- Misclassification rate
- Expected profit or loss
- Profit and loss summary statistics, including the relative contribution of each class.

Prior probabilities will by default affect the following processes if and only if there are two or more decisions in the decision matrix:

- Choice of models in the Regression node
- Early stopping in the Neural Network node
- Pruning trees in the Tree node.

Decisions

Each modeling node can make a decision for each case in a scoring data set, based on numerical consequences specified via a decision matrix and cost variables or cost constants. The decision matrix can specify profit, loss, or revenue. In the GUI, the decision matrix is provided via the Target Profile. With a previously scored data set containing posterior probabilities, decisions can also be made using PROC DECIDE, which reads the decision matrix from a decision data set.

When you use decision processing, the modeling nodes compute summary statistics giving the total and average profit or loss for each model. These profit and loss summary statistics are useful for selecting models. To use these summary statistics for model selection, you must specify numeric consequences for making each decision for each value of the target variable. It is your responsibility to provide reasonable numbers for the decision consequences based on your particular application.

In some applications, the numeric consequences of each decision might not all be known at the time you are training the model. Hence you might want to perform what-if analyses to explore the effects of different decision consequences using the Model Comparison node. In particular, when one of the decisions is to "do nothing," the profit charts in the Model Comparison node provide a convenient way to see the effect of applying different thresholds for the do-nothing decision.

To use profit charts, the do-nothing decision should not be included in the decision matrix; the Model Comparison node will implicitly supply a do-nothing decision when computing the profit charts. When you omit the do-nothing decision from the profit matrix so you can obtain profit charts, you should not use the profit and loss summary statistics for comparing models, since these summary statistics will not incorporate the implicit do-nothing decision. This topic is discussed further in Decision Thresholds and Profit Charts.

The decision matrix contains columns (decision variables) corresponding to each decision, and rows (observations) corresponding to target values. The values of the decision variables represent target-specific consequences, which might be profit, loss, or revenue. These consequences are the same for all cases being scored. A decision data set might contain prior probabilities in addition to the decision matrix.

For a categorical target variable, there should be one row for each class. The value in the decision matrix located at a given row and column specifies the consequence of making the decision corresponding to the column when the target value corresponds to the row. The decision matrix is allowed to contain rows for classes that do not appear in the data being analyzed. For a profit or revenue matrix, the decision is chosen to maximize the expected profit. For a loss matrix, the decision is chosen to minimize the expected loss.

For an interval target variable, each row defines a knot in a piecewise linear spline function. The consequence of making a decision is computed by linear interpolation in the corresponding column of the decision matrix. If the predicted target value is outside the range of knots in the decision matrix, the consequence of a decision is computed by linear extrapolation. Decisions are made to maximize the predicted profit or minimize the predicted loss.

For each decision, there might also be either a cost variable or a numeric cost constant. The values of cost variables represent case-specific consequences, which are always treated as costs. These consequences do not depend on the target values of the cases being scored. Costs are used for computing return on investment as (revenue-cost)/cost.

Cost variables might be specified only if the decision matrix contains revenue, not profit or loss. Hence if revenues and costs are specified, profits are computed as revenue minus cost. If revenues are specified without costs, the costs are assumed to be zero. The interpretation of consequences as profits, losses, revenues, and costs is needed only to compute return on investment. You can specify values in the decision matrix that are target-specific consequences but that might have some practical interpretation other than profit, loss, or revenue. Likewise, you can specify values for the cost variables that are case-specific consequences but that might have some practical interpretation other than costs. If the revenue/cost interpretation is not applicable, the values computed for return on investment might not be meaningful. There are some restrictions on the use of cost variables in the Decision Tree node; see the documentation on the Decision Tree node for more information.

In principle, consequences need not be the sum of target-specific and case-specific terms, but SAS Enterprise Miner does not support such nonadditive consequences.

For a categorical target variable, you can use a decision matrix to classify cases by specifying the same number of decisions as classes and having each decision correspond to one class. However, there is no requirement for the number of decisions to equal the number of classes except for ordinal target variables in the Decision Tree node.

For example, suppose there are three classes designated red, blue, and green. For an identity decision matrix, the average profit is equal to the correct-classification rate:

Table 16.2 *Profit Matrix to Compute the Correct-Classification Rate*

Target Value:	Decision:		
	Red	**Blue**	**Green**
Red	1	0	0
Blue	0	1	0
Green	0	0	1

To obtain the misclassification rate, you can specify a loss matrix with zeros on the diagonal and ones everywhere else:

Table 16.3 *Loss Matrix to Compute the Misclassification Rate*

Target Value:	Decision:		
	Red	**Blue**	**Green**
Red	0	1	1
Blue	1	0	1
Green	1	1	0

If it is 20 times more important to classify red cases correctly than blue or green cases, you can specify a diagonal profit matrix with a profit of 20 for classifying red cases correctly and a profit of one for classifying blue or green cases correctly:

Table 16.4 *Profit Matrix for Detecting a Rare (Red) Class*

Target Value:	Decision:		
	Red	**Blue**	**Green**

Target Value:	Decision:		
Red	20	0	0
Blue	0	1	0
Green	0	0	1

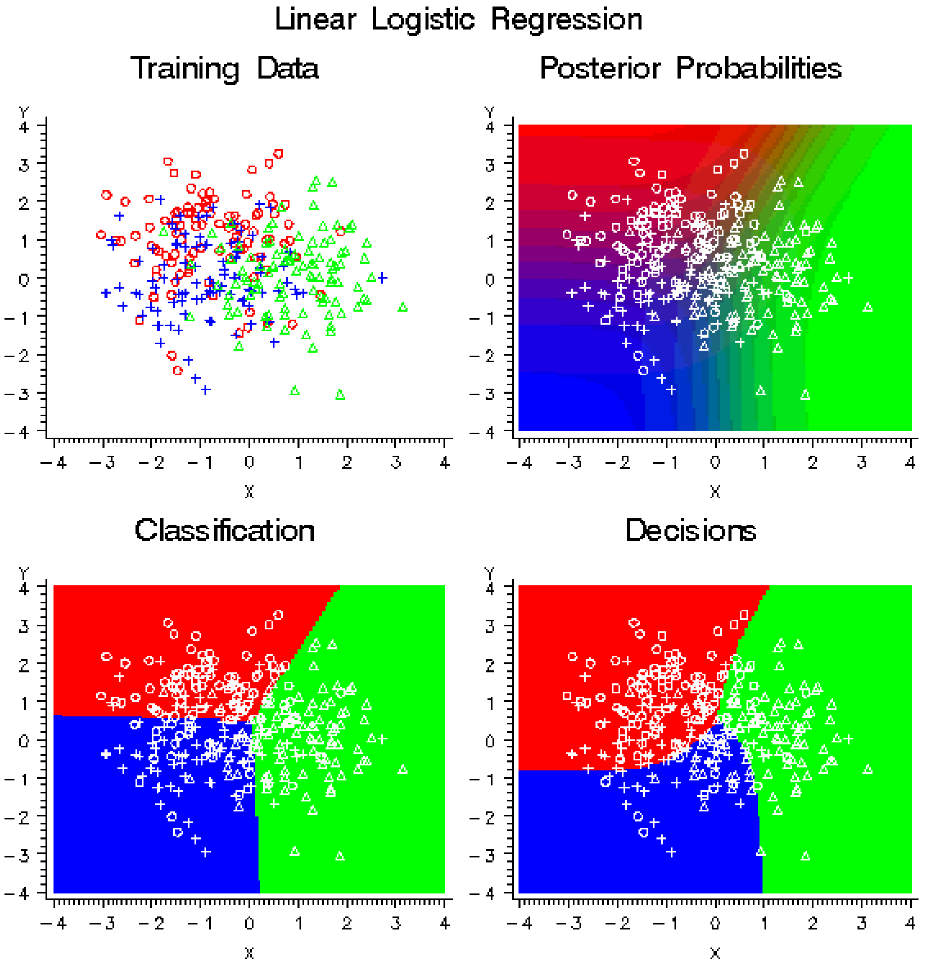
When you use a diagonal profit matrix, the decisions depend only on the products of the prior probabilities and the corresponding profits, not on their separate values. Hence, for any given combination of priors and diagonal profit matrix, you can make any change to the priors (other than replacing a zero with a nonzero value) and find a corresponding change to the diagonal profit matrix that leaves the decisions unchanged, even though the expected profit for each case might change.

Similarly, for any given combination of priors and diagonal profit matrix, you can find a set of priors that will yield the same decisions when used with an identity profit matrix. Therefore, using a diagonal profit matrix does not provide you with any power in decision making that could not be achieved with no profit matrix by choosing appropriate priors (although the profit matrix might provide an advantage in interpretability). Furthermore, any two by two decision matrix can be transformed into a diagonal profit matrix as discussed in the following section on Decision Thresholds and Profit Charts.

When the decision matrix is three by three or larger, it might not be possible to diagonalize the profit matrix, and some nondiagonal profit matrices will produce effects that could not be achieved by manipulating the priors. To show the effect of a nondiagonalizable decision matrix, the data in the upper left plot of the following figure were generated to have three classes, shown as red circles, blue crosses, and green triangles.

Each class has 100 training cases with a bivariate normal distribution. The training data were used to fit a linear logistic regression model using the Neural Network engine. The posterior probabilities are shown in the upper right plot. Classification according to the posterior probabilities yields linear classification boundaries as shown in the lower left plot. Use of a nondiagonalizable decision matrix causes the decision boundaries in the lower right plot to be rotated in comparison with the classification boundaries, and the decision boundaries are curved rather than linear.

Figure 16.6 *Linear Logistic Regression with a Nondiagonal Profit Matrix*



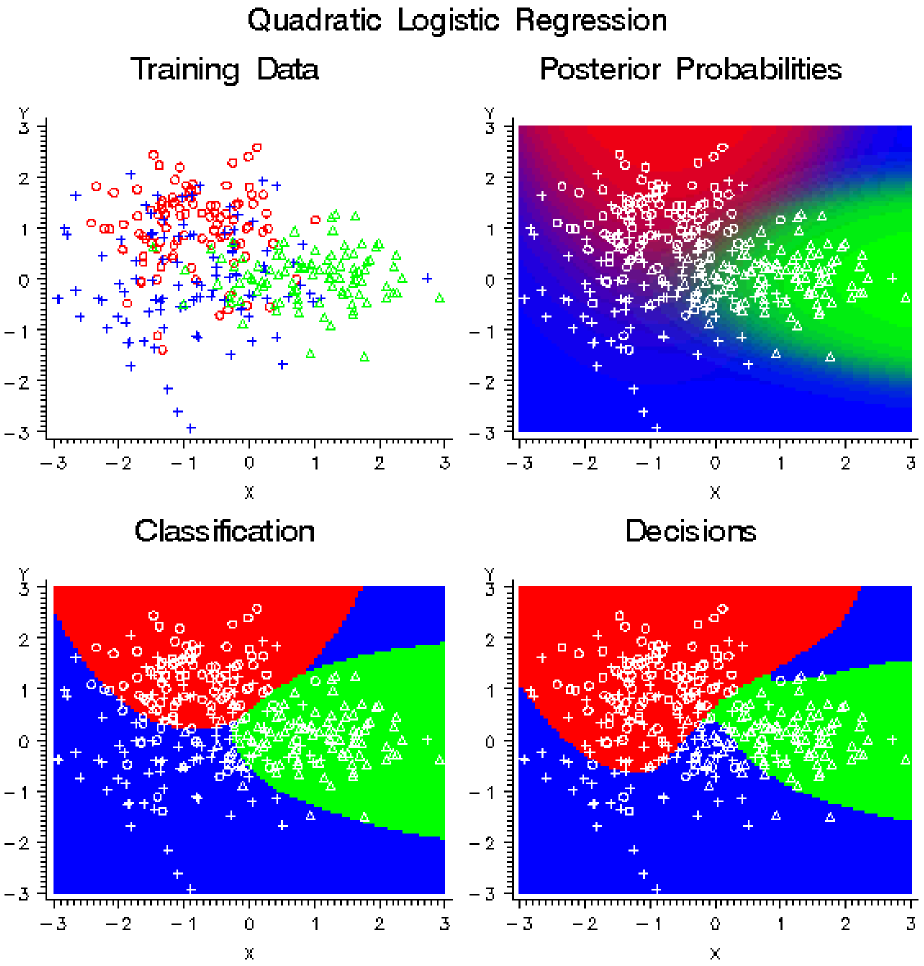
The decision matrix that produced the curved decision boundaries is shown in the following table:

Table 16.5 *Nondiagonal Profit Matrix*

Target Value:	**Decision:**		
	Red	**Blue**	**Green**
Red	4	0	3
Blue	3	4	0
Green	0	3	4

In each row, the two profit values for misclassification are different, hence it is impossible to diagonalize the matrix by adding a constant to each row. Consider the blue row. The greatest profit is for a correct assignment into blue, but there is also a smaller but still substantial profit for assignment into red. There is no profit for assigning red into blue, so the red/blue decision boundary is moved toward the blue mean in comparison with the classification boundary based on posterior probabilities. The following figure shows the effect of the same nondiagonal profit matrix on a quadratic logistic regression model.

Figure 16.7 *Quadratic Logistic Regression with a Nondiagonal Profit Matrix*



For the Neural Network and Regression nodes, a separate decision is made for each case. For the Decision Tree node, a common decision is made for all cases in the same leaf of the tree, so when different cases have different costs, the average cost in the leaf is used in place of the individual costs for each case. That is, the profit equals the revenue minus the average cost among all training cases in the same leaf, hence a single decision is assigned to all cases in the same leaf of a tree.

The decision alternative assigned to a validation, test or scoring case ignores any cost associated with the case. The new data are assumed similar to the training data in cost as well as predictive relations. However, the actual cost values for each case are used for the investment cost, ROI, and quantities that depend on the actual target value.

Decision and cost matrices do not affect:

- Estimating parameters in the Regression node
- Learning weights in the Neural Network node
- Growing (as opposed to pruning) trees in the Decision Tree node unless the target is ordinal

- Residuals, which are based on posteriors before adjustment for priors
- Error functions such as deviance or likelihood
- Fit statistics such as MSE based on residuals or error functions
- Posterior probabilities
- Classification
- Misclassification rate.

- Growing trees in the Decision Tree node when the target is ordinal
- Decisions
- Expected profit or loss
- Profit and loss summary statistics, including the relative contribution of each class.

Decision and cost matrices will by default affect the following processes if and only if there are two or more decisions:

- Choice of models in the Regression node
- Early stopping in the Neural Network node
- Pruning trees in the Decision Tree node.

Formulas will be presented first for the Neural Network and Regression nodes. Let:

t
: be an index for target values (classes)

d
: be an index for decisions

i
: be an index for cases

n_d
: be the number of decisions

Class(t)
: be the set of indices of cases belonging to target t

Profit(t,d)
: be the profit for making decision d when the target is t

Loss(t,d)
: be the loss for making decision d when the target is t

Revenue(t,d)
: be the revenue for making decision d when the target is t

Cost(i,d)
: be the cost for making decision d for case i

Q(i,t,d)
: be the combined consequences for making decision d when the target is t for case i

Prior(t)
: be the prior probability for target t

Paw(t)
: be the prior-adjustment weight for target t

Post(i,t)
: be the posterior probability of target t for case i

F(i)
be the frequency for case i

T(i)
be the index of the actual target value for case i

A(i,d)
be the expected profit of decision d for case i

B(i)
be the best possible profit for case i based on the actual target value

C(i)
be the computed profit for case i based on the actual target value

D(i)
be the index of the decision chosen by the model for case i

E(i)
be the expected profit for case i of the decision chosen by the model

IC(i)
be the investment cost for case i for the decision chosen by the model

ROI(i)
be the return on investment for case i for the decision chosen by the model.

These quantities are related by the following formulas:

$$Profit(t,d) = -Loss(t,d)$$

$$Q(i,t,d) = \begin{cases} \mathrm{Re}\,venue(t,d) - Cost(i,d) & \text{if revenue and costs are specified} \\ \mathrm{Pr}\,ofit(t,d) & \text{if profit is specified} \\ -Loss(t,d) & \text{if loss is specified} \end{cases}$$

When the target variable is categorical, the expected profit for decision d in case i is:

$$A(i,d) = \sum_t Q(i,t,d)Post(i,t)$$

For each case i, the decision is made by choosing D(i) to be the value of d that maximizes the expected profit:

$$D(i) = \arg\max_d A(i,d) = \arg\max_d \sum_t Q(i,t,d)Post(i,t)$$

If two or more decisions are tied for maximum expected profit, the first decision in the user-specified list of decisions is chosen.

The expected profit E(i) is the expected combined consequence for the chosen decision D(i), computed as a weighted average over the target values of the combined consequences, using the posterior probabilities as weights:

$$E(i) = A(i,D(i)) = \sum_t Q(i,t,D(i))Post(i,t)$$

The expected loss is the negative of expected profit.

Note that E(i) and D(i) can be computed without knowing the target index T(i). When T(i) is known, two more quantities useful for evaluating the model can also be computed. C(i) is the profit computed from the target value using the decision chosen by the model:

$$C(i) = Q(i, T(i), D(i))$$

The loss computed from the target value is the negative of C(i). C(i) is the most important variable for assessing and comparing models. The best possible profit for any of the decisions, which is an upper bound for C(i), is:

$$B(i) = \max_{d} Q(i, T(i), d)$$

The best possible loss is the negative of B(i).

When revenue and cost are specified, investment cost is:

$$IC(i) = Cost(i, D(i))$$

And return on investment is:

$$ROI = \begin{cases} \dfrac{C(i)}{IC(i)} & IC(i) > 0 \\ .I(\infty) & IC(i) \leq 0, C(i) > 0 \\ .(\text{missing }) & IC(i) \leq 0, C(i) = 0 \\ ,M(-\infty) & IC(i) \leq 0, C(i) < 0 \end{cases}$$

For an interval target variable, let:

Y(i)
be the actual target value for case i

P(i)
be the predicted target value for case i

K(t)
be the knot value for the row of the decision matrix.

For interval targets, the predicted value is assumed to be accurate enough that no integration over the predictive distribution is required. Define the functions:

$$K_{-}(y) = \max\{t \mid K(t) \leq y\}$$

$$K_{+}(y) = \min\{t \mid K(t) \geq y\}$$

$$L(i, y, d) = Q(i, K_{-}(y), d) + \frac{y - K_{-}(y)}{K_{+}(y) - K_{-}(y)}[Q(i, K_{+}(y), d) - Q(i, K_{-}(y), d)]$$

Then the decision is made by maximizing the expected profit:

$$D(i) = \arg\max_{d} L(i, P(i), d)$$

The expected profit for the chosen decision is:

$$E(i) = L(i, P(i), D(i))$$

When Y(i) is known, the profit computed from the target value using the decision chosen by the model is:

$$C(i) = L(i, Y(i), D(i))$$

And the best possible profit for any of the decisions is:

$$B(i) = \max_{d} L(i, Y(i), d)$$

For both categorical and interval targets, the summary statistics for decision processing with profit and revenue matrices are computed by summation over cases with nonmissing cost values. If no adjustment for prior probabilities is used, the sums are weighted only by the case frequencies, hence total profit and average profit are given by the following formulas:

$$TotalProfit = \sum_{i} F(i)C(i)$$

$$AverageProfit = \frac{TotalProfit}{\sum_{i} F(i)}$$

For loss matrices, total loss and average loss are the negatives of total profit and average profit, respectively.

If total and average profit are adjusted for prior probabilities, an additional weight Paw(t)is used:

$$Paw(t) = \frac{Prior(t)}{\sum_{i \in Class(t)} F(i)} \sum_{i} F(i)$$

Total and average profit are then given by:

$$TotalProfit = \sum_{i} F(i)C(i)Paw[T(i)] = \sum_{i} Paw(t) \sum_{i \in Class(t)} F(i)C(i)$$

$$AverageProfit = \frac{TotalProfit}{\sum_{i} F(i)}$$

If any class with a positive prior probability has a total frequency of zero, total and average profit and loss cannot be computed and are assigned missing values. Note that the adjustment of total and average profit and loss is done only if you explicitly specify prior probabilities; the adjustment is not done when the implicit priors based on the training set proportions are used.

The adjustment for prior probabilities is not done for fit statistics such as SSE, deviance, likelihood, or misclassification rate. For example, consider the situation shown in the following table:

	Proportion In				**Unconditional Misclassification Rate**	
Class	**Operational Data**	**Training Data**	**Prior Probability**	**Conditional Misclassification Rate**	**Unadjusted**	**Adjusted**
Rare	0.1	0.5	0.1	0.8	0.5 * 0.8 + 0.5 * 0.2 = 0.50	0.1 * 0.8 + 0.9 * 0.2 = 0.26
Common	0.9	0.5	0.9	0.2		

There is a rare class comprising 10% of the operational data, and a common class comprising 90%. For reasons discussed in the section below on Detecting Rare Classes, you might want to train using a balanced sample with 50% from each class. To obtain correct posterior probabilities and decisions, you specify prior probabilities of .1 and .9 that are equal to the operational proportions of the two classes.

Suppose the conditional misclassification rate for the common class is low, just 20%, but the conditional misclassification rate for the rare class is high, 80%. If it is important to detect the rare class accurately, these misclassification rates are poor.

The unconditional misclassification rate computed using the training proportions without adjustment for priors is a mediocre 50%. But adjusting for priors, the unconditional misclassification rate is apparently much better at only 26%. Hence the adjusted misclassification rate is misleading.

For the Decision Tree node, the following modifications to the formulas are required. Let Leaf(i) be the set of indices of cases in the same leaf as case i. Then:

$$\widetilde{Cost}(i,d) = \frac{\sum_{j \in Leaf(i)} F(j) Cost(j,d)}{\sum_{j \in Leaf(i)} F(j)}$$

The combined consequences are:

$$\tilde{Q}(i,t,d) = \begin{cases} Revenue(t,d) - \widetilde{Cost}(i,d) & \text{if revenue and costs are specified} \\ Profit(t,d) & \text{if profit is specified} \\ -Loss(t,d) & \text{if loss is specified} \end{cases}$$

For a categorical target, the decision is:

$$D(i) = \arg\max_{d} \sum_{t} \tilde{Q}(i,t,d) Post(i,t)$$

And the expected profit is:

$$E(i) = \sum_{t} \tilde{Q}(i,t,D(i)) Post(i,t)$$

For an interval target:

$$\tilde{L}(i,t,d) = \tilde{Q}(i,K_{-}(y),d) + \frac{y - K_{-}(y)}{K_{+}(y) - K_{-}(y)} \left[\tilde{Q}(i,K_{+}(y),d) - \tilde{Q}(i,K_{-}(y),d) \right]$$

The decision is:

$$D(i) = \frac{\arg\max_{d} \sum_{j \in Leaf(i)} F(j) \tilde{L}(j,P(j),d)}{\sum_{j \in Leaf(i)} F(j)}$$

And the expected profit is:

$$E(i) = \frac{\arg\max_{d} \sum_{j \in Leaf(i)} F(j) \tilde{L}(j,P(j),D(i))}{\sum_{j \in Leaf(i)} F(j)}$$

The other formulas are unchanged.

Decision Thresholds and Profit Charts

There are two distinct ways of using decision processing in SAS Enterprise Miner:

- Making firm decisions in the modeling nodes and comparing models on profit and loss summary statistics. For this approach, you include all possible decisions in the decision matrix. This is the traditional approach in statistical decision theory.
- Using a profit chart to set a decision threshold. For this approach, there is an implicit decision (usually a decision to "do nothing") that is not included in the decision matrix. The decisions made in the modeling nodes are tentative. The profit and loss

summary statistics from the modeling nodes are not used. Instead, you look at profit charts (similar to lift or gains charts) in the Model Comparison node to decide on a threshold for the do-nothing decision. Then you use a Transform Variables or SAS Code node that sets the decision variable to "do nothing" when the expected profit or loss is not better than the threshold chosen from the profit chart. This approach is popular for business applications such as direct marketing.

To understand the difference between these two approaches to decision making, you first need to understand the effects of various types of transformations of decisions on the resulting decisions and summary statistics.

Consider the formula for the expected profit of decision d in case i using (without loss of generality) revenue and cost:

$$\begin{aligned} A(i,d) &= \sum_t Q(i,t,d)\,Post(i,t) \\ &= \sum_t [Revenue(t,d) - Cost(i,d)]Post(i,t) \\ &= \sum_t Revenue(t,d)Post(i,t) - Cost(i,d)\sum_t Post(i,t) \end{aligned}$$

Now transform the decision problem by adding a constant to the tth row of the revenue matrix and a constant c_i to the ith row of the cost matrix, yielding a new expected profit A'(i,d):

$$\begin{aligned} A'(i,d) &= \sum_t [Revenue(t,d) + r_t]Post(i,t) - [Cost(i,d) + c_i]\sum_t Post(i,t) \\ &= A(i,d) + \sum_t r_t Post(i,t) + c_i \end{aligned}$$

In the last expression above, the second and third terms do not depend on the decision. Hence this transformation of the decision problem will not affect the choice of decision.

Consider the total profit before transformation and without adjustment for priors:

$$\begin{aligned} Total\,\mathrm{Pr}\,ofit &= \sum_i F(i)C(i) \\ &= \sum_i F(i)Q(i,T(i),D(i)) \\ &= \sum_i F(i)[Revenue(T(i),D(i)) - Cost(i,D(i))] \end{aligned}$$

After transformation, the new total profit, TotalProfit', is:

$$\begin{aligned} Total\,\mathrm{Pr}\,ofit' &= \sum_i F(i)[Revenue)T(i),D(i)) + r_i - Cost(i,D(i)) - c_i] \\ &= \sum_i F(i)[r_t - c_i] \end{aligned}$$

In the last expression above, the second term does not depend on the posterior probabilities and therefore does not depend on the model. Hence this transformation of the decision problem adds the same constant to the total profit regardless of the model, and the transformation does not affect the choice of models based on total profit. The same conclusion applies to average profit and to total and average loss, and also applies when the adjustment for prior probabilities is used.

For example, in the German credit benchmark data set (SAMPSIO.DMAGECR), the target variable indicates whether the credit risk of each loan applicant is good or bad, and a decision must be made to accept or reject each application. It is customary to use the loss matrix:

Table 16.6 *Customary Loss Matrix for the German Credit Data*

Target Value	Decision	
	Accept	**Reject**
Good	0	1
Bad	5	0

This loss matrix says that accepting a bad credit risk is five times worse than rejecting a good credit risk. But this matrix also says that you cannot make any money no matter what you do, so the results might be difficult to interpret (or perhaps you should just get out of business). In fact, if you accept a good credit risk, you will make money, that is, you will have a negative loss. And if you reject an application (good or bad), there will be no profit or loss aside from the cost of processing the application, which will be ignored. Hence it would be more realistic to subtract one from the first row of the matrix to give a more realistic loss matrix:

Table 16.7 *Realistic Loss Matrix for the German Credit Data*

Target Value	Decision	
	Accept	**Reject**
Good	– 1	0
Bad	5	0

This loss matrix will yield the same decisions and the same model selections as the first matrix, but the summary statistics for the second matrix will be easier to interpret.

Sometimes a decision threshold K is used to modify the decision-making process, so that no decision is made unless the maximum expected profit exceeds K. However, making no decision is really a decision to make no decision or to "do nothing." Thus the use of a threshold implicitly creates a new decision numbered N_d+1. Let $D_k(i)$ be the decision based on threshold K. Thus:

$$D_k(i) = \begin{cases} \arg\max\limits_{d=1}^{N_d} A(i,d) & \text{if } A(i,d) > K \\ N_d + 1 & \text{otherwise} \end{cases}$$

If the decision and cost matrices are correctly specified, then using a threshold is suboptimal, since D(i) is the optimal decision, not $D_k(i)$. But a threshold-based decision can be reformulated as an optimal decision using modified decision and cost matrices in several ways.

A threshold-based decision is optimal if "doing nothing" actually yields an additional revenue K. For example, K might be the interest earned on money saved by doing nothing. Using the profit matrix formulation, you can define an augmented profit matrix Profit* with N_d+1 columns, where:

$$Profit^*(t,d) = \begin{cases} Profit(t,d) & d \le N_d \\ K & d = N_d + 1 \end{cases}$$

Let D*(i) be the decision based on Profit*, where:

$$D^*(i) = \arg\max_{d=1}^{N_d} \sum_t Profit^*(t,d) Post(i,t)$$

Then D*(i) = $D_K(i)$. Equivalently, you can define augmented revenue and cost matrices, Revenue*> and Cost*, each with N_d+1 columns, where:

$$Revenue^*(t,d) = \begin{cases} Revenue(t,d) & d \le N_d \\ K & d = N_d + 1 \end{cases}$$

$$Cost^*(i,d) = \begin{cases} Cost(i,d) & d \le N_d \\ -K & d = N_d + 1 \end{cases}$$

Then the decision D*(i) based on Revenue* and Cost* is:

$$D^*(i) = \arg\max_{d=1}^{N_d} \sum_t Profit^*(t,d) Post(i,t)$$

Again, D*(i) = $D_K(i)$.

A threshold-based decision is also optimal if doing anything other than nothing actually incurs an additional cost K. In this situation, you can define an augmented profit matrix Profit* with N_d+1 columns, where:

$$Profit^{*}(t,d) = \begin{cases} Profit(t,d) - K & d \le N_d \\ 0 & d = N_d + 1 \end{cases}$$

This version of Profit* produces the same decisions as the previous version, but the total profit is reduced by $K\sum F(i)$ regardless of the model used. Similarly, you can define Revenue* and Cost* as:

$$Revenue^{*}(t,d) = \begin{cases} Revenue(t,d) & d \le N_d \\ K & d = N_d + 1 \end{cases}$$

$$Cost^{*}(i,d) = \begin{cases} Cost(i,d) - K & d \le N_d \\ 0 & d = N_d + 1 \end{cases}$$

Again, this version of the Revenue* and Cost* matrices produces the same decisions as the previous version, but the total profit is reduced by $K\sum F(i)$ regardless of the model used.

If you want to apply a known decision threshold in any of the modeling nodes in SAS Enterprise Miner, use an augmented decision matrix as described above. If you want to explore the consequences of using different threshold values to make suboptimal decisions, you can use profit charts in the Model Comparison node with a non-augmented decision matrix. In a profit chart, the horizontal axis shows percentile points of the expected profit E(i). By the default, the deciles of E(i) are used to define 10 bins with equal frequencies of cases. The vertical axis can display either cumulative or noncumulative profit computed from C(i).

To see the effect on total profit of varying the decision threshold K, use a cumulative profit chart. Each percentile point p on the horizontal axis corresponds to a threshold K equal to the corresponding percentile of E(i). That is:

$$\frac{p}{100} = \frac{\sum_{i \mid E(i) < K} F(i)}{\sum_{i} F(i)}$$

However, the chart shows only p, not K. Since the chart shows cumulative profit, each case with E(i) < K contributes a profit of C(i), while all other cases contribute a profit of zero. Hence the ordinate (vertical coordinate) of the curve is the total profit for the decision rule $D_k(i)$, assuming that the profit for the decision to do nothing is zero:

$$\sum_{i \mid E(i) < K} F(i)C(i)$$

Transformations that add a constant r_t to the t^th row of the revenue matrix or a constant c_i to the i^th row of the cost matrix can change the expected profit for different cases by different amounts and therefore can alter the order of the cases along the horizontal axis of a profit chart, producing large changes in the cumulative profit curve.

To obtain a profit chart for the German credit data, you need to:

1. Transform the decision matrix to have a column of zeros, as in the "Realistic Loss Matrix" above.
2. Omit the zero column.

Hence the decision matrix presented to the Model Comparison node should be:

Target Value	Decision
	Accept
Good	–1
Bad	5

Detecting Rare Classes

In data mining, predictive models are often used to detect rare classes. For example, an application to detect credit card fraud might involve a data set containing 100,000 credit card transactions, of which only 100 are fraudulent. Or an analysis of a direct marketing campaign might use a data set representing mailings to 100,000 customers, of whom only 5,000 made a purchase. Since such data are noisy, it is quite possible that no credit card transaction will have a posterior probability over 0.5 of being fraudulent, and that no customer will have a posterior probability over 0.5 of responding. Hence, simply classifying cases according to posterior probability will yield no transactions classified as fraudulent and no customers classified as likely to respond.

When you are collecting the original data, it is always good to over-sample rare classes if possible. If the sample size is fixed, a balanced sample (that is, a nonproportional stratified sample with equal sizes for each class) will usually produce more accurate predictions than an unbalanced 5% / 95% split. For example, if you can sample any 100,000 customers,, it would be much better to have 50,000 responders and 50,000 nonresponders than to have 5,000 responders and 95,000 nonresponders.

Sampling designs like this that are stratified on the classes are called case-control studies or choice-based sampling and have been extensively studied in the statistics and econometrics literature. If a logistic regression model is well-specified for the population ignoring stratification, estimates of the slope parameters from a sample stratified on the classes are unbiased. Estimates of the intercepts are biased but can be easily adjusted to be unbiased, and this adjustment is mathematically equivalent to adjusting the posterior probabilities for prior probabilities.

If you are familiar with survey-sampling methods, you might be tempted to apply sampling weights to analyze a balanced stratified sample. Resist the temptation! In sample surveys, sampling weights (inversely proportional to sampling probability) are used to obtain unbiased estimates of population totals. In predictive modeling, you are not primarily interested in estimating the total number of customers who responded to a mailing, but in identifying which individuals are more likely to respond. Use of sampling weights in a predictive model reduces the effective sample size and makes predictions less accurate. Instead of using sampling weights, specify the appropriate prior probabilities and decision consequences, which will provide all the necessary adjustments for nonproportional stratification on classes.

Unfortunately, balanced sampling is often impractical. The remainder of this section will be concerned with samples where the class sizes are severely unbalanced.

Methods for dealing with the problem of rare classes include:

- Specifying correct decision consequences. This is the method of choice with SAS Enterprise Miner, although in some circumstances discussed below, additional methods might also be needed.
- Using false prior probabilities. This method is commonly used with software that does not support decision matrices. When there are only two classes, the same decision results can be obtained either by using false priors or by using correct decision matrices, but with three or more classes, false priors do not provide the full power of decision matrices. You should not use false priors with SAS Enterprise Miner, because Enterprise Mines adjusts profit and loss summary statistics for priors, hence using false priors might give you false profit and loss summary statistics.
- Over-weighting, or weighting rare classes more heavily than common classes during training. This method can be useful when there are three or more classes, but it reduces the effective sample size and can degrade predictive accuracy. Over-weighting can be done in SAS Enterprise Miner by using a frequency variable. However, the current version of SAS Enterprise Miner does not provide full support for sampling weights or other types of weighted analyses, so this method should be approached with care in any analysis where standard errors or significance tests are used, such as stepwise regression. When using a frequency variable for weighting in SAS Enterprise Miner, it is recommended that you also specify appropriate prior probabilities and decision consequences.
- Under-sampling, or omitting cases from common classes in the training set. This method throws away information but can be useful for very large data sets in which the amount of information lost is small compared to the noise level in the data. As with over-weighting, the main benefits occur when there are three or more classes. When using under-sampling, it is recommended that you also specify appropriate prior probabilities and decision consequences. Unless you are using this method simply to reduce computational demands, you should not weight cases (using a frequency variable) in inverse proportion to the sampling probabilities, since the use of sampling weights would cancel out the effect of using nonproportional sampling, accomplishing nothing.
- Duplicating cases from rare classes in the training set. This method is equivalent to using a frequency variable, except that duplicating cases requires more computer time and disk space. Hence, this method is not recommended except for incremental backprop training in the Neural Network node.

A typical scenario for analyzing data with a rare class would proceed as follows:

1. In the Input Data node, open a data set containing a random sample of the population. Specify the prior probabilities in the target profile: For a simple random sample, the priors are proportional to the data. For a stratified random sample, you have to type in numbers for the priors. Also specify the decision matrix in the target profile, including a do-nothing decision if applicable. The profit for choosing the best decision for a case from a rare class should be larger than the profit for choosing the best decision for a case from a common class.
2. You have the option to do the following: For over-weighting, assign a role of Frequency to the weighting variable in the Data Source wizard or Metadata node, or

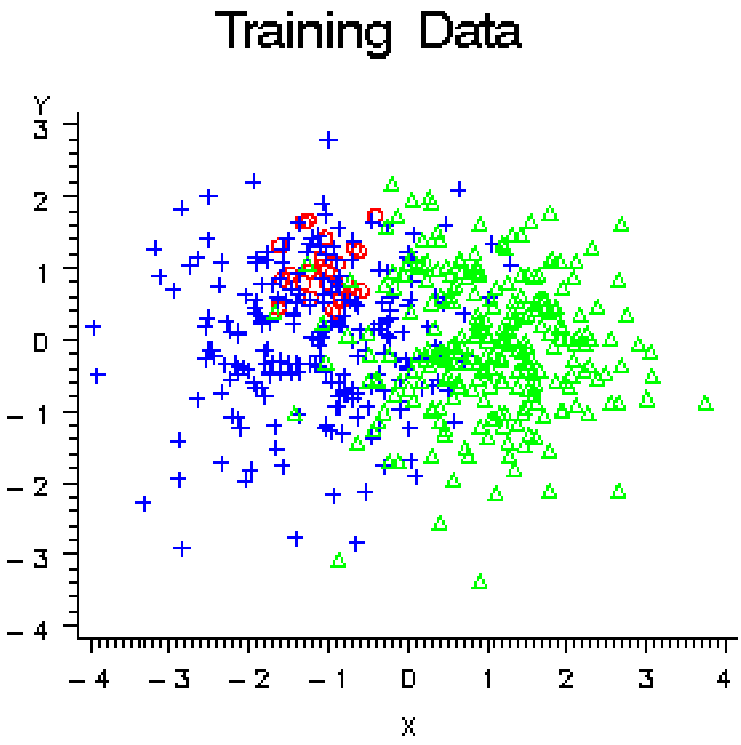
compute a weighting variable in the Transform Variables node. For under-sampling, use the Sampling node to do stratified sampling on the class variable with the Equal Size option.

3. Use the Data Partition node to create training, validation, and test sets.
4. Use one or more modeling nodes.
5. In the Model Comparison node, compare models based on the total or average profit or loss in the validation set.
6. To produce a profit chart in the Model Comparison node, open the target profile for the model of interest and delete the do-nothing decision.

Specifying correct prior probabilities and decision consequences is generally sufficient to obtain correct decision results if the model that you use is well-specified. A model is well-specified if there exist values for the weights and/or other parameters in the model that provide a true description of the population, including the distribution of the target noise. However, it is the nature of data mining that you often do not know the true form of the mechanism underlying the data, so in practice it is often necessary to use misspecified models. It is often assumed that trees and neural nets are only asymptotically well-specified.

Over-weighting or under-sampling can improve predictive accuracy when there are three or more classes, including at least one rare class and two or more common classes. If the model is misspecified and lacks sufficient complexity to discriminate all of the classes, the estimation process will emphasize the common classes and neglect the rare classes unless either over-weighting or under-sampling is used. For example, consider the data with three classes in the following plot:

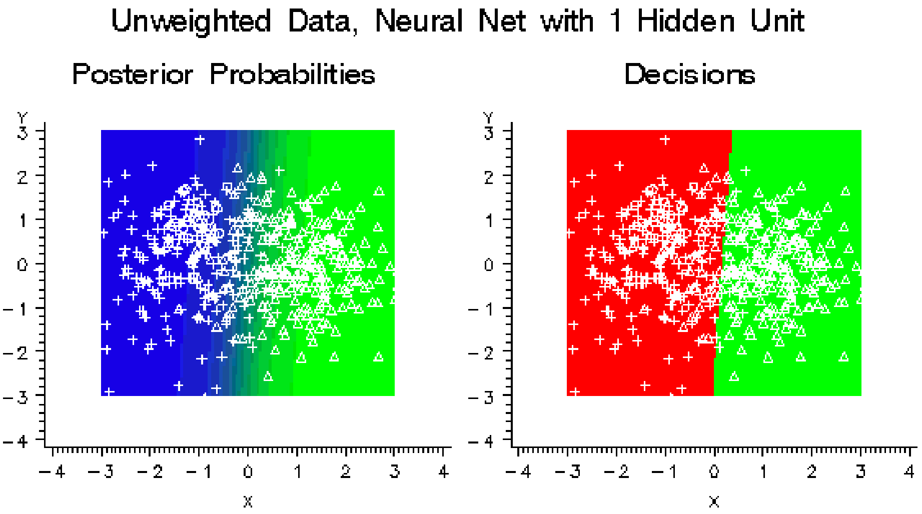
Figure 16.8 *Data with One Rare Class and Two Common Classes*



The two common classes, blue and green, are separated along the X variable. The rare class, red, is separated from the blue class only along the Y variable. A variable selection method based on significance tests, such as stepwise discriminant analysis, would choose X first, since both the R2 and F statistics would be larger for X. But if you were more interested in detecting the rare class, red, than in distinguishing between the common classes, blue and green, you would prefer to choose Y first.

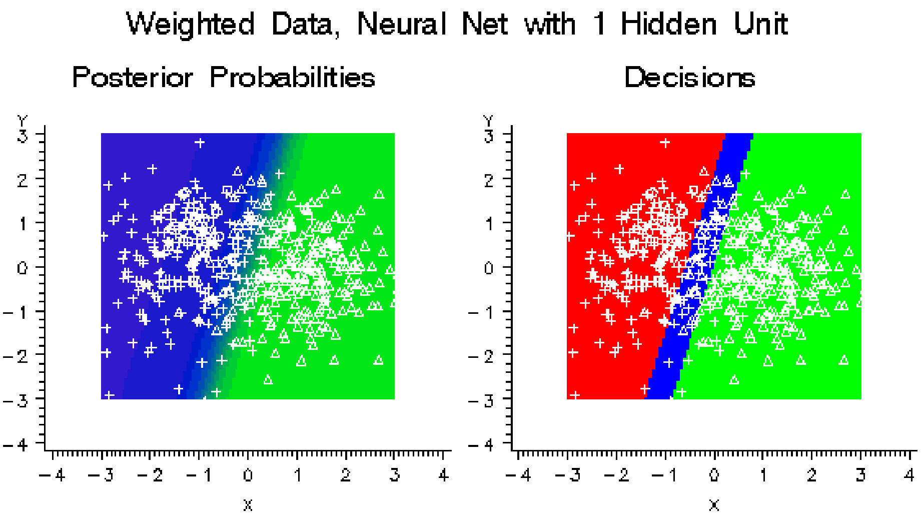
Similarly, if these data were used to train a neural network with one hidden unit, the hidden unit would have a large weight along the X variable, but it would essentially ignore the Y variable, as shown by the posterior probability plot in the following figure. Note that no cases would be classified into the red class using the posterior probabilities for classification. But when a diagonal decision matrix is used, specifying 20 times as much profit for correctly assigning a red case as for correctly assigning a blue or green case, about half the cases are assigned to red, while no cases at all are assigned to blue.

Figure 16.9 *Unweighted Data, Neural Net with 1 Hidden Unit*



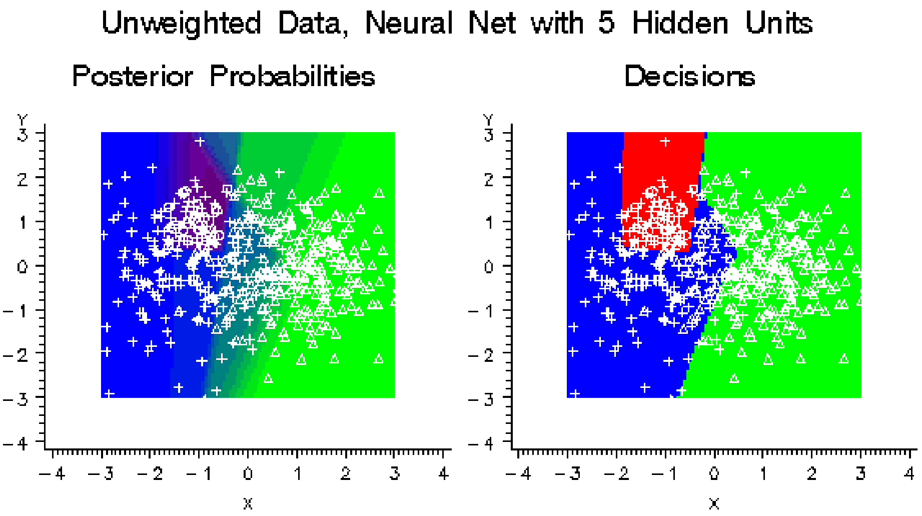
If you weighted the classes in a balanced manner by creating a frequency variable with values inversely proportional to the number of training cases in each class, the hidden unit would learn a linear combination of the X and Y variables that provides moderate discrimination among all three classes instead of high discrimination between the two common classes. But since the model is misspecified, the posterior probabilities are still not accurate. As the following figure shows, there is enough improvement that each class is assigned some cases.

Figure 16.10 *Plots of Weighted Data, Neural Net with 1 Hidden Unit*



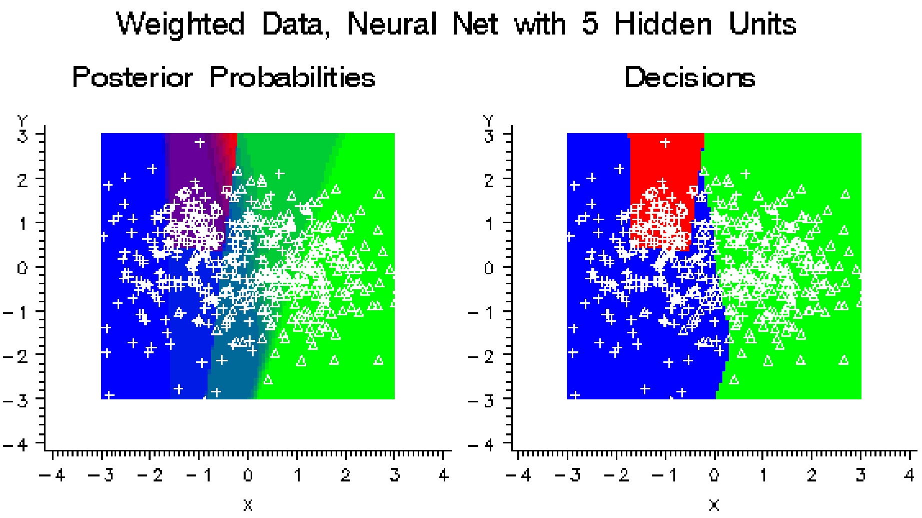
If the neural network had five hidden units instead of just one, it could learn the distributions of all three classes more accurately without the need for weighting, as shown in the following figure:

Figure 16.11 *Unweighted Data, Neural Net with 5 Hidden Units*



Using balanced weights for the classes would have only a small effect on the decisions, as shown in the following figure:

Figure 16.12 *Weighted Data, Neural Net with 5 Hidden Units*



While using balanced weights for a well-specified neural network will not usually improve predictive accuracy, it might make neural network training faster by improving numerical condition and reducing the risk of bad local optima.

While balanced weighting can be important when there are three or more classes, there is little evidence that balance is important when there are only two classes. Scott and Wild (1989) have shown that for a well-specified logistic regression model, balanced weighting increases the standard error of every linear combination of the regression coefficients and therefore reduces the accuracy of the posterior probability estimates. Simulation studies, which will be described in a separate report, have found that even for misspecified models, balanced weighting provides little improvement and often degrades the total profit or loss in logistic regression, normal-theory discriminant analysis, and neural networks.

Generalization

The critical issue in predictive modeling is generalization: how well will the model make predictions for cases that are not in the training set? Data mining models, like other flexible nonlinear estimation methods such as kernel regression, can suffer from either underfitting or overfitting (or as statisticians usually say, oversmoothing or undersmoothing). A model that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A model that is too complex might fit the noise, not just the signal, leading to overfitting. Overfitting can happen even with noise-free data and, especially in neural nets, can yield predictions that are far beyond the range of the target values in the training data.

By making a model sufficiently complex, you can always fit the training data perfectly. For example, if you have N training cases and you fit a linear regression with N-1 inputs, you can always get zero error (assuming that the inputs are not singular). Even if the N-1 inputs are random numbers that are totally unrelated to the target variable, you will still get zero error on the training set. However, the predictions are worthless for such a regression model for new cases that are not in the training set.

Even if you use only one continuous input variable, by including enough polynomial terms in a regression, you can get zero training error. Similarly, you can always get a perfect fit with only one input variable by growing a tree large enough or by adding enough hidden units to a neural net.

On the other hand, if you omit an important input variable from a model, both the training error and the generalization error will be poor. If you use too few terms in a regression, or too few hidden units in a neural net, or too small a tree, then again the training error and the generalization error might be poor.

Hence, with all types of data mining models, you must strike a balance between a model that is too simple and one that is too complex. It is usually necessary to try a variety of models and then choose a model that is likely to generalize well.

There are many ways to choose a model. Some popular methods are heuristic, such as stepwise regression or CHAID tree modeling, where the model is modified in a sequence of steps that terminates when no further steps satisfy a statistical significance criterion. Such heuristic methods might be of use for developing explanatory models, but they do not directly address the question of which model will generalize best. The obvious way to approach this question directly is to estimate the generalization error of each model, then choose the model with the smallest estimated generalization error.

There are many ways to estimate generalization error, but it is especially important not to use the training error as an estimate of generalization error. As previously mentioned, the training error can be very low even when the generalization error is very high. Choosing a model based on training error will cause the most complex model to be chosen even if it generalizes poorly.

A better way to estimate generalization error is to adjust the training error for the complexity of the model. In linear least squares regression, this adjustment is fairly simple if the input variables are assumed fixed or multivariate normal. Let

SSEE
be the sum of squared errors for the training set

N
be the number of training cases

P
be the number of estimated weights including the intercept.

Then the average squared error for the training set is SSE/N, which is designated as ASE by SAS Enterprise Miner modeling nodes. Statistical software often reports the mean squared error, MSE=SSE/(N-P).

MSE adjusts the training error for the complexity of the model by subtracting P in the denominator, which makes MSE larger than ASE. But MSE is not a good estimate of the generalization error of the trained model. Under the usual statistical assumptions, MSE is an unbiased estimate of the generalization error of the model with the best possible ("true") weights, not the weights that were obtained by training.

Hence, a stronger adjustment is required to estimate generalization error of the trained model. One way to provide a stronger adjustment is to use Akaike's Final Prediction Error (FPE):

$$FPE = \frac{SSE(N+P)}{N(N-P)}$$

The formula for FPE multiplies MSE by (N+P)/N, so FPE is larger than MSE. If the input variables are fixed rather than random, FPE is an unbiased estimate of the generalization error of the trained model. If inputs and target are multivariate normal, a further adjustment is required:

$$\frac{SSE(N+1)(N-2)}{N(N-P)(N-P-1)}$$

which is slightly larger than FPE but has no conventional acronym.

The formulas for MSE and FPE were derived for linear least squares regression. For nonlinear models and for other training criteria, MSE and FPE are not unbiased. MSE and FPE might provide adequate approximations if the model is not too nonlinear and the number of training cases is much larger than the number of estimated weights. But simulation studies have shown, especially for neural networks, that FPE is not a good criterion for model choice, since it does not provide a sufficiently severe penalty for overfitting.

There are other methods for adjusting the training error for the complexity of the model. Two of the most popular criteria for model choice are Schwarz's Bayesian criterion, (SBC), also called the Bayesian information criterion, (BIC), and Rissanen's minimum description length principle (MDL). Although these two criteria were derived from different theoretical frameworks — SBC from Bayesian statistics and MDL from information theory — they are essentially the same, and in SAS Enterprise Miner only the acronym SBC is used. For least squares training,

$$SBC = N\ln(\frac{SSE}{N}) + P\ln(N)$$

For maximum likelihood training,

$$SBC = 2NLL + P\ln(N)$$

where NLL is the negative log likelihood. Smaller values of SBC are better, since smaller values of SSE or NLL are better. SBC often takes negative values. SBC is valid for nonlinear models under the usual statistical regularity conditions. Simulation studies have found that SBC works much better than FPE for model choice in neural networks.

However, the usual statistical regularity conditions might not hold for neural networks, so SBC might not be entirely satisfactory. In tree-based models, there is no well-defined number of weights, P, in the model, so SBC is not directly applicable. And other types of models and training methods exist for which no single-sample statistics such as SBC are known to be good criteria for model choice. Furthermore, none of these adjustments for model complexity can be applied to decision processing to maximize total profit. Fortunately, there are other methods for estimating generalization error and total profit that are very broadly applicable; these methods include split-sample or holdout validation, cross validation, and bootstrapping.

Split-sample validation is applicable with any type of model and any training method. You split the available data into a training set and a validation set, usually by simple random sampling or stratified random sampling. You train the model using only the training set. You estimate the generalization error for each model that you train by scoring the validation set. Then you select the model with the smallest validation error. Split-sample validation is fast and is often the method of choice for large data sets. For small data sets, split-sample validation is not so useful because it does not make efficient use of the data.

For small data sets, cross validation is generally preferred to split-sample validation. Cross validation works by splitting the data several ways, training a different model for each split, and then combining the validation results across all the splits. In k-fold cross validation, you divide the data into k subsets of (approximately) equal size. You train the model k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute the error criterion. If k equals the sample size, this is called "leave-one-out" cross validation.

"Leave-v-out" is a more elaborate and expensive version of cross validation that involves leaving out all possible subsets of v cases. Cross validation makes efficient use of the data because every case is used for both training and validation. But, of course, cross validation requires more computer time than split-sample validation. SAS Enterprise Miner provides leave-one-out cross validation in the Regression node; k-fold cross validation can be done easily with SAS macros.

In the literature of artificial intelligence and machine learning, the term "cross validation" is often applied incorrectly to split-sample validation, causing much confusion. The distinction between cross validation and split-sample validation is extremely important because cross validation can be markedly superior for small data sets. On the other hand, leave-one-out cross validation might perform poorly for discontinuous error functions such as the number of misclassified cases, or for discontinuous modeling methods such as stepwise regression or tree-based models. In such discontinuous situations, split-sample validation or k-fold cross validation (usually with k equal to five or ten) are preferred, depending on the size of the data set.

Bootstrapping seems to work better than cross validation in many situations, such as stepwise regression, at the cost of even more computation. In the simplest form of bootstrapping, instead of repeatedly analyzing subsets of the data, you repeatedly analyze subsamples of the data. Each subsample is a random sample with replacement from the full sample. Depending on what you want to do, anywhere from 200 to 2000 subsamples might be used. There are many more sophisticated bootstrap methods that can be used not only for estimating generalization error but also for estimating bias, standard errors, and confidence bounds.

Not all bootstrapping methods use resampling from the data — you can also resample from a nonparametric density estimate, or resample from a parametric density estimate, or, in some situations, you can use analytical results. However, bootstrapping does not work well for some methods such as tree-based models, where bootstrap generalization estimates can be excessively optimistic.

There has been relatively little research on bootstrapping neural networks. SAS macros for bootstrap inference can be obtained from Technical Support.

When numerous models are compared according to their estimated generalization error (for example, the error on a validation set), and the model with the lowest estimated generalization error is chosen for operational use, the estimate of the generalization error of the selected model will be optimistic. This optimism is a consequence of the statistical principle of regression to the mean. Each estimate of generalization error is subject to random fluctuations. Some models by chance will have an excessively high estimate of generalization error, while others will have an excessively low estimate of generalization error.

The model that wins the competition for lowest generalization error is more likely to be among the models that by chance have an excessively low estimate of generalization error. Even if the method for estimating the generalization error of each model individually provides an unbiased estimate, the estimate for the winning model will be biased downward. Hence, if you want an unbiased estimate of the generalization error of the winning model, further computations are required to obtain such an estimate.

For large data sets, the most practical way to obtain an unbiased estimate of the generalization error of the winning model is to divide the data set into three parts, not just two: the training set, the validation set, and the test set. The training set is used to train each model. The validation set is used to choose one of the models. The test data set is used to obtain an unbiased estimate of the generalization error of the chosen model.

The training/validation/test data set approach is explained by Bishop (1995, p. 372) as follows:

"Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the hold out method. Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set."

Input and Output Data Sets

Train, Validate, and Test Data Sets

Because SAS Enterprise Miner is intended especially for the analysis of large data sets, all of the predictive modeling nodes are designed to work with separate training, validation, and test sets. The Data Partition node provides a convenient way to split a single data set into the three subsets, using simple random sampling, stratified random sampling, or user defined sampling. Each predictive modeling node also enables you to specify a fourth scoring data set that is not required to contain the target variable. These four different uses for data sets are called the roles of the data sets. For the training, validation and test sets, the predictive modeling nodes can produce two output data sets: one containing the original data plus scores (predicted values, residuals, classification results, and so on), the other containing various statistics pertaining to the fit of the model (the error function, misclassification rate, and so on). For scoring sets, only the output data set containing scores can be produced.

Scored Data Sets

Output data sets that contain scores have new variables with names that are usually formed by adding prefixes to the name of the target variable or variables (and, in some situations, the input variables or the decision data set). It is best to avoid using the following prefixes when naming your variables. Because variables with the following prefixes are created automatically during the modeling process, it is possible for your data to be overwritten during the modeling process.

Table 16.8 Prefixes Commonly Used in Scored Data Sets

Prefix	Root	Description	Target Needed?
BL_	Decision data set	Best possible loss of any of the decisions, –B(i)	Yes
BP_	Decision data set	Best possible profit of any of the decisions, B(i)	Yes

Prefix	Root	Description	Target Needed?
CL_	Decision data set	Loss computed from the target value, –C(i)	Yes
CP_	Decision data set	Profit computed from the target value, C(i)	Yes
D_	Decision data set	Label of the decision chosen by the model	No
E__	Target	Error function	Yes
EL__	Decision data set	Expected loss for the decision chosen by the model, –E(i)	No
EP__	Decision data set	Expected profit for the decision chosen by the model, E(i)	No
F_	Target	Normalized category that the case comes from	Yes
I__	Target	Normalized category that the case is classified into	No
IC_	Decision data set	Investment cost IC(i)	No
M__	Variable	Missing indicator dummy variable	—
P__	Target or dummy	Outputs (predicted values and posterior probabilities)	No
Q_	Target or dummy	Old posteriors, prior to adjustment for priors	No
R__	Target or dummy	Plain residuals: target minus output	Yes
RA__	Target	Anscombe residuals	Yes
RAS_	Target	Standardized Anscombe residuals	Yes
RAT_	Target	Studentized Anscombe residuals	Yes
RD_	Target	Deviance residuals	Yes

Prefix	Root	Description	Target Needed?
RDS_	Target	Standardized deviance residuals	Yes
RDT_	Target	Studentized deviance residuals	Yes
ROI_	Decision data set	Return on investment, ROI(i)	Yes
RS_	Target	Standardized residuals	Yes
RT_	Target	Studentized residuals	Yes
S_	Variable	Standardized variable	—
T__	Variable	Studentized variable	—
U__	Target	Unformatted category that the case is classified into	No
V_	Validation data set	Predicted values from the validation data	No

Usually, for categorical targets, the actual target values are dummy 0/1 variables. Hence the outputs (P_) are estimates of posterior probabilities. Some modeling nodes might allow other ways of fitting categorical targets. For example, when the Regression node fits an ordinal target by linear least squares, it uses the index of the category as the actual target value, and hence does not produce posterior probabilities.

Outputs (P_) are always predictions of the actual target variable, even if the target variable is standardized or otherwise rescaled during modeling computations. Similarly, plain residuals (R_) are always the actual target value minus the output. Plain residuals are not multiplied by error weights or by frequencies.

For least squares estimation, the error function variable (E_) contains the squared error for each case. For generalized linear models or other methods based on minimizing deviance, the E_ variable is the deviance. For other types of maximum likelihood estimations, the E_ variable is the negative log likelihood. In other words, the E_ variable is whatever the training method is trying to minimize the sum of.

The deviance residual is the signed square root of the value of the error function for a given case. In other words, if you square the deviance residuals, multiply them by the frequency values, and add them, you will get the value of the error function for the entire data set. Hence if the target variable is rescaled, the deviance residuals are based on the rescaled target values, not on the actual target values. However, deviance residuals cannot be computed for categorical target variables.

For categorical target variables, names for dummy target variables are created by concatenating the target name with the formatted target values, with invalid characters replaced by underscores. Output and residual names are created by adding the appropriate prefix (P_, R_, and so on) to the dummy target variable names. The F_ variable is the formatted value of the target variable. The I_ variable is the category that the case is classified into—also a formatted value. The I_ value is the category with the highest posterior probability. If a decision matrix is used, the D_ value is the decision with the largest estimated profit or smallest estimated loss. The D_ value might differ from the I_ value for two reasons:

- The decision alternatives do not necessarily correspond to the target categories.
- The I_ depends directly on the posterior probabilities, not on estimated profit or loss.

However, the I_ value can depend indirectly on the decision matrix when the decision matrix is used in model estimation or selection.

Predicted values are computed for all cases. The model is used to compute predicted values whenever possible, regardless of whether the target variable is missing, inputs excluded from the model (for example, by stepwise selection) are missing, the frequency variable is missing, and so on. When predicted values cannot be computed using the model—for example, when required inputs are missing—the P_ variables are set according to an intercept-only model:

- For an interval target, the P_ variable is the unconditional mean of the target variable.
- For categorical targets, the P_ variables are set to the prior probabilities.

Scored output data sets also contain a variable named _WARN_ that indicates problems computing predicted values or making decisions. _WARN_ is a character variable that either is blank, indicating there were no problems, or that contains one or more of the following character codes:

***Table 16.9** _WARN_ Codes*

Code	Meaning
C	Missing cost variable
M	Missing inputs
P	Invalid posterior probability (for example, <0 or >1)
U	Unrecognized input category

Regardless of how the P_ variables are computed, the I_ variables as well as the residuals and errors are computed exactly the same way given the values of the P_ variables. All cases with nonmissing targets and positive frequencies contribute to the fit statistics. It is important that all such cases be included in the computation of fit statistics because model comparisons must be based on exactly the same sets of cases for every model under consideration, regardless of which modeling nodes are used.

Fixed Variable Names

When you score a data set, SAS Enterprise Miner maps the output variable names to fixed variable names. This mapping is appropriate in the most common case—that is, when there is only one prediction target or one classification target. The table below will help make sense of other cases.

Using the fixed variable names enables scoring users to build processes that can be reused for different models without changing the code that processes the outputs. These fixed names are listed in the **emoutput.xml** file and are described in the table below. Most scoring processes return one or more of these variables.

Fixed Name	Role	Type	Description
EM_ CLASSIFICATION	Classification	$	The predicted target class value.
EM_DECISION	Decision	$	Optimal decision based on a function of probability, cost, and profit or loss weights.
EM_ EVENTPROBABILITY	Probability	N	The probability of the target event. By default, this is the first value in descending order, but you can alter the ordering scheme. This is often the event of interest.
EM_LOSS	Expected Loss	N	Based on the selected decision.
EM_PREDICTION	Prediction	N	The prediction value for an interval target variable.
EM_PROBABILITY	Probability	N	The probability of the predicted classification, which can be any one of the target variable values.
EM_PROFIT	Expected Profit	N	Based on the selected decision.
EM_ROI	Return on Investment	N	Based on the selected decision.
EM_SEGMENT	Decision Tree Leaf, Cluster number, or SOM cell ID	N	Analytical customer segmentation.

Fit Statistics

The output data sets containing fit statistics produced by the Regression node and the Decision Tree node have only one record. Since the Neural Network node can analyze multiple target variables, it produces one record for each target variable and one record for the overall fit; the variable called _NAME_ indicates which target variable the statistics are for.

The fit statistics for training data generally include the following variables, computed from the sum of frequencies and ordinary residuals:

Table 16.10 *Variables Included in Fit Statistics for Training Data*

Name	Label
NOBS	Sum of Frequencies
DFT	Total Degrees of Freedom
DIV	Divisor for ASE
ASE	Train: Average Squared Error
MAX	Train: Maximum Absolute Error
RASE	Train: Root Average Squared Error
SSE	Train: Sum of Squared Error

Note that _DFT_, _DIV_, and _NOBS_ can all be different when the target variable is categorical.

The following fit statistics are computed according to the error function (such as squared error, deviance, or negative log likelihood) that was minimized:

Table 16.11 *Fit Statistics Computed According to the Error Function*

Name	Label
AIC	Akaike's Information Criterion
AVERR	Total Degrees of Freedom
ERR	Divisor for ASE
SBC	Schwarz's Bayesian Criterion

For a categorical target variable, the following statistics are also computed:

Table 16.12 *Additional Statistics Computed for a Categorical Target Variable*

Name	Label
MISC	Train: Misclassification Rate
WRONG	Train: Number of Wrong Classifications

When decision processing is done, the statistics in the following table are also computed for the training set. The profit variables are computed for a profit or revenue matrix, and the loss variables are computed for a loss matrix:

Table 16.13 Additional Statistics Computed for Decision Processing

Name	Label
PROF	Train: Total Profit
APROF	Train: Average Profit
LOSS	Train: Total Loss
ALOSS	Train: Average Loss

For a validation data set, the variable names contain a V following the first underscore. For a test data set, the variable names contain a T following the first underscore. Not all the fit statistics are appropriate for validation and test sets, and adjustments for model degrees of freedom are not applicable. Hence, ASE and MSE become the same. For a validation set, the following fit statistics are computed:

Table 16.14 Fit Statistics Computed for a Validation Data Set

Name	Label
VASE	Valid: Average Squared Error
VAVERR	Valid: Average Error Function
VDIV	Valid: Divisor for ASE
VERR	Valid: Error Function
VMAX	Valid: Maximum Absolute Error
VMSE	Valid: Mean Squared Error
VNOBS	Valid: Sum of Frequencies
VRASE	Valid: Root Average Squared Error
VRMSE	Valid: Root Mean Square Error
VSSE	Valid: Sum of Squared Errors

For a validation set and a categorical target variable, the following fit statistics are computed:

Table 16.15 Fit Statistics Computed for a Validation Data Set with a Categorical Target Variable

Name	Label
VMISC	Valid: Misclassification Rate

Name	Label
VWRONG	Valid: Number of Wrong Classifications

When decision processing is done, the following statistics are also computed for the validation set:

Table 16.16 *Additional Statistics Computed for Decision Processing*

Name	Label
VPROF	Valid: Total Profit
VAPROF	Valid: Average Profit
VLOSS	Valid: Total Loss
VALOSS	Valid: Average Loss

Cross validation statistics are similar to the above except that the letter X appears instead of V. These statistics appear in the same data set or data sets as fit statistics for the training data. For a test set, the following fit statistics are computed:

Table 16.17 *Fit Statistics Computed for a Test Data Set*

Name	Label
TASE	Test: Average Squared Error
TAVERR	Test: Average Error Function
TDIV	Test: Divisor for ASE
TERR	Test: Error Function
TMAX	Test: Maximum Absolute Error
TMSE	Test: Mean Squared Error
TNOBS	Test: Sum of Frequencies
TRASE	Test: Root Average Squared Error
TRMSE	Test: Root Mean Square Error
TSSE	Test: Sum of Squared Errors

For a test data set and a categorical target variable, the following fit statistics are computed:

Table 16.18 Fit Statistics Computed for a Test Data Set with a Categorical Target Variable

Name	Label
TMISC	Test: Misclassification Rate
TMISL	Test: Lower 95% Confidence Limit for TMISC
TMISU	Test: Upper 95% Confidence Limit for TMISC
TWRONG	Test: Number of Wrong Classifications

When decision processing is done, the following statistics are also computed for the test set:

Table 16.19 Fit Statistics Computed for Test Data Sets Using Decision Processing

Name	Label
TPROF	Test: Total Profit
TAPROF	Test: Average Profit
TLOSS	Test: Total Loss
TALOSS	Test: Average Loss

Multiple Models

An average of several measurements is often more accurate than a single measurement. This happens when the errors of individual measurements more often cancel each other than reinforce each other. An average is also more stable than an individual measurement: if different sets of measurements are made on the same object, their averages would be more similar than individual measurements in a single set.

A similar phenomenon exists for predictive models: a weighted average of predictions is often more accurate and more stable than an individual model prediction. Though similar to what happens with measurements, it is less common and more surprising. A model relates inputs to a target. It seems surprising that a better relationship exists than is obtainable with a single model. Combining the models must produce a relationship not obtainable in any individual model.

An algorithm for training a model assumes some form of the relationship between the inputs and the target. Linear regression assumes a linear relation. Tree-based models assume a constant relation within ranges of the inputs. Neural networks assume a nonlinear relationship that depends on the architecture and activation functions chosen for the network.

Combining predictions from two different algorithms might produce a relationship of a different form than either algorithm assumes. If two models specify different relationships and fit the data well, their average is apt to fit the data better. If not, an individual model is apt to be adequate. In practice, the best way to know is to combine some models and compare the results.

For neural networks, applying the same algorithm several times to the same data might produce different results, especially when early stopping is used, since the results might be sensitive to the random initial weights. Averaging the predictions of several networks trained with early stopping often improves the accuracy of predictions.

Combining Models

Ensembles

An ensemble or committee is a collection of models regarded as one combined model. The ensemble predicts a target value as an average or a vote of the predictions of the individual model. The different individual models can give different weights to the average or vote.

For an interval target, an ensemble averages the predictions. For a categorical target, an ensemble might average the posterior probabilities of the target values. Alternatively, the ensemble might classify a case into the class that most of the individual models classify it. The latter method is called voting and is not equivalent to the method of averaging posteriors. Voting produces a predicted target value but does not produce posterior probabilities consistent with combining the individual posteriors.

Unstable Algorithms

Sometimes applying the same algorithm to slightly different data produces very different models. Stepwise regression and tree-based models behave this way when two important inputs have comparable predictive ability. When a tree creates a splitting rule, only one input is chosen. Changing the data slightly might tip the balance in favor of choosing the other input. A split on one input might separate the data very differently than a split on the other input. In this situation, all descendent splits are apt to be different.

The unstable nature of tree-based models renders the interpretation of trees tricky. A business can continually collect new data, and a tree created in June might look very different from one created the previous January. An analyst who depended on the January tree for understanding the data is apt to become distrustful of the tree in June, unless he investigated the January tree for instability. The analyst should check the competing splitting rules in a node. If two splits are comparably predictive and the input variables suggest different explanations, then neither explanation tells the whole story.

Scoring New Data

All the predictive modeling nodes enable you to score the training, validation, test, and scoring data sets in conjunction with training. To score other data sets, especially new data not available at the time of training, use the Score node.

Each predictive modeling node generates SAS DATA step code for computing predicted values. The Score node accumulates the code generated by each modeling node that precedes the Score node in the flow diagram. The Score node then packages all the scoring code into a DATA step that can be executed to score new data sets. The scoring code can be saved for use in the SAS System outside of SAS Enterprise Miner.

The Score node also handles:

- code for transformations generated by the Transform Variables node
- code for missing-value imputation generated by the Impute node

- code for cluster assignment generated by the Cluster node
- code for decision processing.

You can use a SAS Code node following the Score node to do additional processing of the scored data. For example, if you used the Model Comparison node to choose a decision threshold, you could apply the threshold in a SAS Code node.

References

Bishop, C. M. 1995. Neural Networks for Pattern Recognition. Oxford: Oxford University Press.

Breiman, L. 1996. "Bagging Predictors." Machine Learning 24:123-140.

Breiman, L. 1998. "Arcing Classifiers." Annals of Statistics 26:801-824.

Freund, Y. 1995. "Boosting a weak learning algorithm by majority." Information and Computation 121:256-285.

Freund, Y. and R. Schapire. 1996. "Experiments with a new boosting algorithm." In Machine Learning: Proceedings of the Thirteenth International Conference, 148-156.

Friedman, H. J. 1999. "Greedy Function Approximation: A Gradient Boosting Machine." Technical report in postscript file trebst.ps available from: **http://www.stat.stanford.edu/~jhf/ftp**.

Friedman, H. J., T. Hastie, and R. Tibshirani. 1998. "Additive Logistic Regression: a Statistical View of Boosting." Technical report available through **ftp: ftp://stat.stanford.edu/pub/friedman/boost.ps.Z**.

Scott, A. J. and C. J. Wild. 1989. "Selection based on the response variable in logistic regression," In Analysis of Complex Surveys, eds. C. J. Skinner, D. Holt, and T. M. F. Smith, 191-205. New York: John Wiley & Sons.

Chapter 17
Enterprise Miner Target Profiler

Enterprise Miner Target Profiler

Overview of the Target Profiler

The Target Profiler enables you to define target profiles for a target that produce optimal decisions that are based on a user-supplied decision matrix and output from a subsequent modeling procedure. The output from the modeling procedure can be either posterior probabilities for the classes of a categorical target or predicted values of an interval target variable. You can also adjust posterior probabilities for changes in the prior probabilities. Before you try to define a target profile, you should be familiar with the Predictive Modeling document. The following sections from the Predictive Modeling document are especially important:

- Predictive Modeling: Prior Probabilities on page 166
- Predictive Modeling: Decisions on page 171
- Predictive Modeling: Decision Thresholds and Profit Charts on page 182
- Predictive Modeling: Detecting Rare Classes on page 187

Creating Target Profiles for a Data Source

A data source in SAS Enterprise Miner stores all the information that is associated with a SAS data set, for example, name, location of the file, variables roles, and measurement levels. A data source can be used in any diagram within a project. It can also be copied from one project to another. You can create target profiles for a data source and use the target profile in any diagram within a project.

You can create target profiles when you create a data source. For detailed information, see Creating a Data Source in the Data Source Wizard section in the SAS Enterprise Miner 12.3 Data Sources document.

To create or modify target profiles after data sources are defined, use one of the following ways to open the Decision Editor.

- Select the data source in the Project Navigator, and click the ... button of the Decisions property in the Properties Panel.
- Right-click the data source in the Project Navigator and select Edit Decisions.

Note: Target profiles are not used by default in SAS Enterprise Miner 12.3. You must explicitly specify one when you create a data source or in the Input Data node.

Example Data Used to Define Target Profiles

The layout of the target profile depends on the measurement level of the target. To familiarize yourself with the different profile settings that are demonstrated in this document, create a data source from the catalog mailing data set SAMPSIO.DMEXA1 as follows. Then, assign the target model role to PURCHASE, NTITLE, and NUMCARS. Ensure that the measurement level of NUMCARS is set to ordinal.

PURCHASE is a binary target that contains a value of YES if a purchase was made and a value of NO if a purchase was not made. NTITLE is a nominal target that contains a value of Ms, Mr, Mrs, or None. Note that the primary difference in defining a target profile for a nominal target versus a binary target is that the nominal target contains more than two levels. NUMCARS is an ordinal target that contains ordered values of 0, 1, 2, and 3.

Layout of a Target Profile

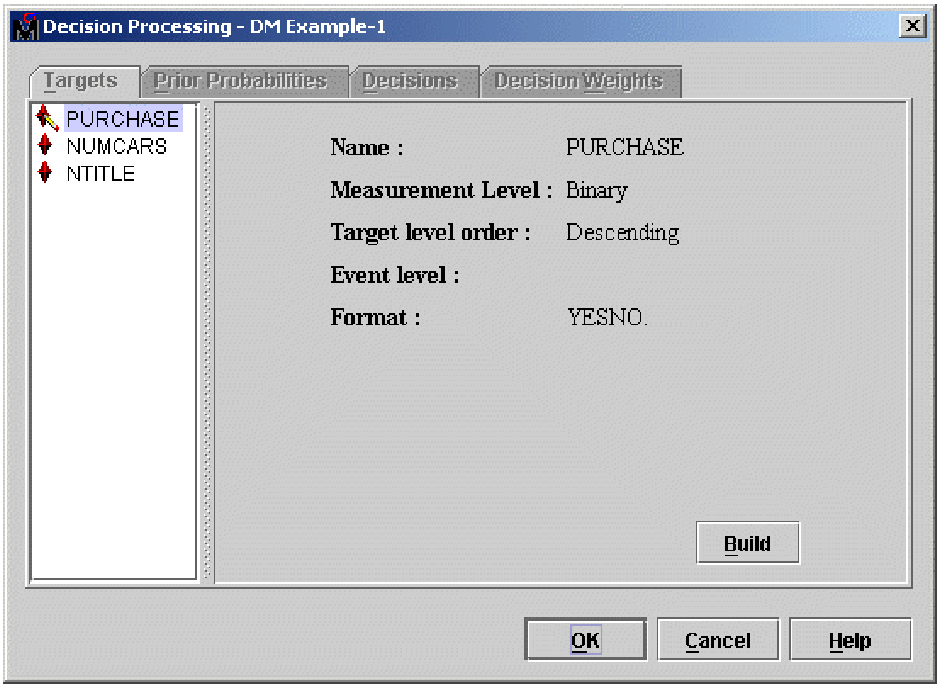
Decision Editor – Decision Configuration Window

The Decision Editor – Decision Configuration window has the following tabs:

- Targets Tab on page 208
- Prior Probabilities Tab on page 210
- Decisions Tab on page 211
- Decision Weights Tab. on page 212

Targets Tab

The left panel of the Targets tab displays the target variables that are defined in the data source. For categorical targets, the right panel of the Targets tab lists the variable name, measurement level, order, and the event level. Decision matrices are no longer automatically built when you open the Decision Editor for new targets. Click the new Build button to create the decision matrix (target profile) for the desired target. If a target profile has already existed for the target, the button will be changed to the Refresh button which sets the target profile to the default one.



Name : PURCHASE

Measurement Level : Binary

Target level order : Descending

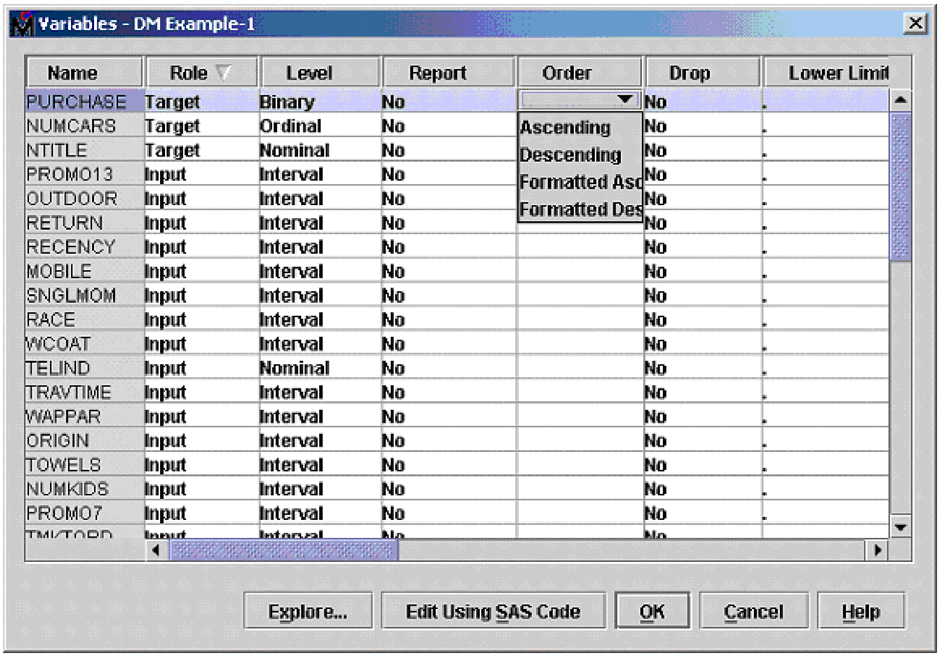
Event level :

Format : YESNO.

Setting the Event Level for a Target Variable

If the target variable is a binary variable (for example, a 0,1, or a No, Yes variable), then the Target tab displays the target event level in the Event field. It is important that you examine the event level and make sure that it is correct for your analysis. The Model Comparison node is dependent on the event level when it calculates assessment statistics, such as lift and profit. For example, to determine the number of buyers (PURCHASE= Yes) in the first decile, you need to make sure that the event is set to Yes. The event level is also used by the Regression node for logistic regression analyses. In this case, you want to make sure that you are modeling the probability of the event rather than the nonevent. Otherwise, it can be confusing when you try to interpret the logistic regression analysis without explicitly setting the event level of interest.

Because the event level is set to Yes in this example, the Regression node will model the probability that a purchase is made. To model the probability that a purchase will not be made, you need to set the order value for PURCHASE to Ascending in the Variable Editor. The following display shows an example of the Variable Editor. To change the event level, select the order from the drop-down list in the Order column.



Name	Role	Level	Report	Order	Drop	Lower Limit
PURCHASE	Target	Binary	No		No	.
NUMCARS	Target	Ordinal	No		No	.
NTITLE	Target	Nominal	No		No	.
PROMO13	Input	Interval	No		No	.
OUTDOOR	Input	Interval	No		No	.
RETURN	Input	Interval	No		No	.
RECENCY	Input	Interval	No		No	.
MOBILE	Input	Interval	No		No	.
SNGLMOM	Input	Interval	No		No	.
RACE	Input	Interval	No		No	.
WCOAT	Input	Interval	No		No	.
TELIND	Input	Nominal	No		No	.
TRAVTIME	Input	Interval	No		No	.
WAPPAR	Input	Interval	No		No	.
ORIGIN	Input	Interval	No		No	.
TOWELS	Input	Interval	No		No	.
NUMKIDS	Input	Interval	No		No	.
PROMO7	Input	Interval	No		No	.

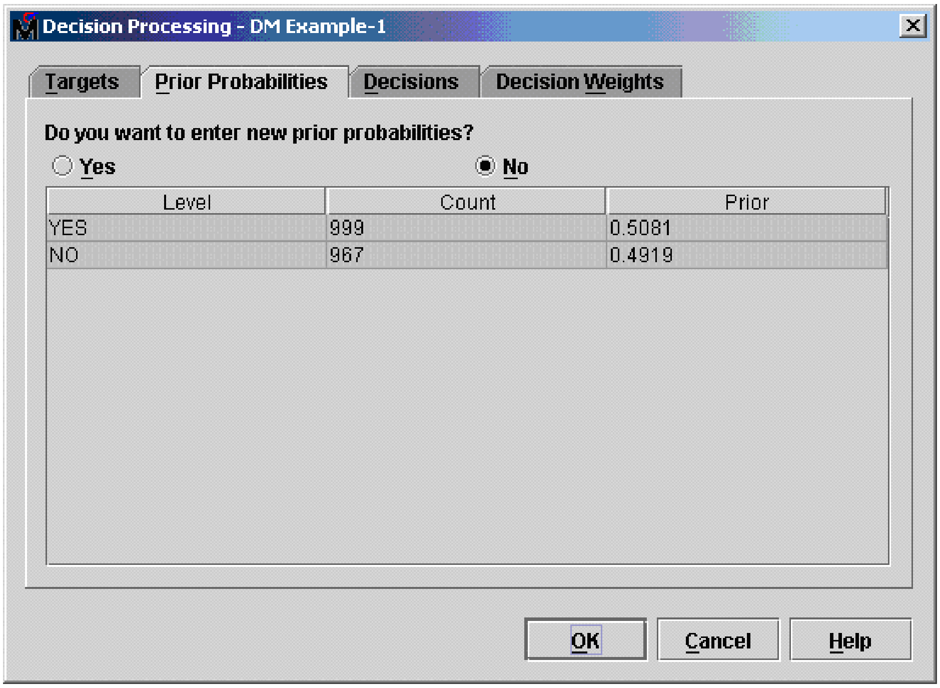
For example, a binary target variable that has values of 0 and 1 and formatted values of No and Yes, where No is mapped to 0 and Yes is mapped to 1, has the following target event level:

Table 17.1 *Binary Target Variable Event Levels*

Order	Target Event Level
Ascending	0
Descending	1
Formatted Ascending	No
Formatted Descending	Yes

Prior Probabilities Tab

You use the Prior Probabilities tab to specify prior probability values to implement prior class probabilities for categorical targets.



Do you want to enter new prior probabilities?

○ Yes ● No

Level	Count	Prior
YES	999	0.5081
NO	967	0.4919

The Prior Probabilities tab has the following columns:

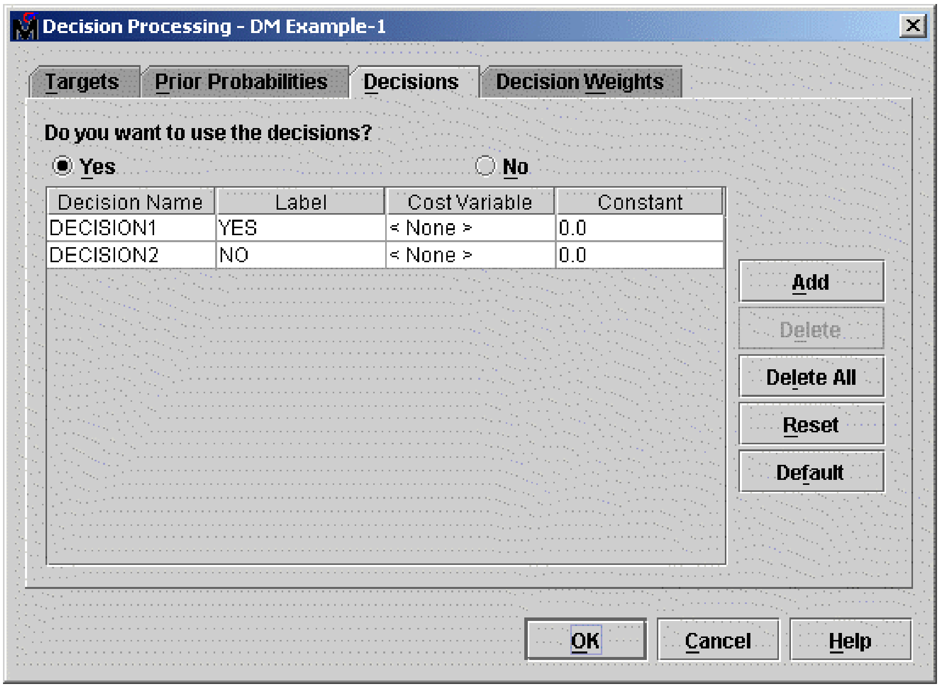
- **Target Level** — displays the levels for target variables.
- **Count** — displays the number of training observations for each target level.
- **Data Prior** — displays the percentage of training observations for each target level.
- **Adjusted Prior** — displays the prior probability values that you specified. The default values are equal probability values for each level of the target.

  To use the prior probabilities that you typed, select the Yes radio button.

Decisions Tab

You use the Decisions tab to add and delete decisions to specify a numeric constant cost or a cost variable for each decision. Cost can be specified only if the decision matrix contains revenue (for maximizing a decision function).

The following display shows an example of the Decisions tab. To specify a cost variable, select the corresponding field for a decision and select the variable from the drop-down list. In order to use a variable as the cost variable, the variable role must be set to Cost in the Variable Editor. To specify a constant cost, select _CONSTANT_ from the drop-down list and enter a value in the corresponding Constant field.



Do you want to use the decisions?

● Yes ○ No

Decision Name	Label	Cost Variable	Constant
DECISION1	YES	< None >	0.0
DECISION2	NO	< None >	0.0

- To add a decision, click **Add**.
- To delete a decision, select a decision, and click **Delete**.
- To delete all decisions, click **Delete All**.
- To reset all the settings in the Decisions and Decision Weights tabs back to the values that you had when you opened the decision editor, click **Reset**.
- To use the default values for all the settings in the Decisions and Decision Weights tabs, click **Default**. The default matrices contain a decision column for each corresponding target level . No cost variable or information is used by default. See Decision Weights Tab for more information about default matrices.

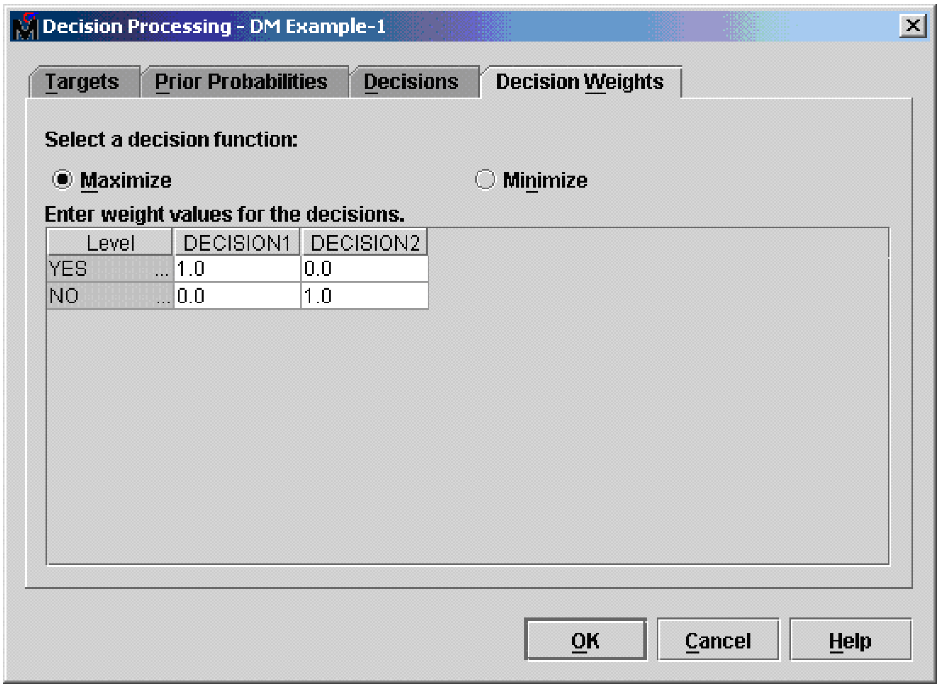
Decision Weights Tab

Target Levels and Decision Weight Matrices

You use the Decisions Weights tab to specify the values that you want to use in your decision matrix. The decision weight matrix contains columns that correspond to each decision, and rows that correspond to target values. The values of the decision variables represent target-specific consequence, which can be PROFIT, LOSS, or REVENUE. Based on the values of the decision variables, select either the **Maximize** or **Minimize** radio button to specify the assessment objective for the matrix. Select **Maximize** if the values in the decision matrix represent profit or revenue. Select **Minimize** if the values in the decision matrix represent loss. **Maximize** is the default setting.

The target levels that are displayed in the default decision weight matrix depend on the order of target levels. By default, it is set to descending for categorical targets.

In the following example, the target variable has two levels, Yes and No. To change the values in the decision weight matrix, select a field in the matrix and type a number.



Select a decision function:

● Maximize ○ Minimize

Enter weight values for the decisions.

Level	DECISION1	DECISION2
YES ...	1.0	0.0
NO ...	0.0	1.0

Default Matrix for Binary Targets

The default matrix for a binary target is a profit matrix that contains a decision column for each target level:

Target Level	Yes	No
Yes	1	0
No	0	1

Default Matrix for Nominal Targets

The default matrix for a nominal target is a profit matrix that contains a decision column for each target level.

Target Level	None	MS	MRS	MR
None	1	0	0	0
MS	0	1	0	0
MRS	0	0	1	0
MR	0	0	0	1

Default Matrix for Ordinal Targets

The default matrix for a nominal target is an unweighted profit matrix that contains a decision column for each target level.

Level	Decision 1	Decision 2	Decision 3	Decision 4
3	3.0	2.0	1.0	0.0
2	2.0	3.0	2.0	1.0
1	1.0	2.0	3.0	2.0
0	0.0	1.0	2.0	3.0

CAUTION:

When applying a decision matrix in the split search for an ordinal target, the number of decisions must be equal to the number of ordinal target levels. If this is not true and you run the Tree node, it displays a message stating that the "Decision alternatives must equal ordinal target values", and then fails. To ensure that each partitioned data set contains all of the ordinal target levels, you should use a stratified sampling method by the ordinal target to create the partitioned data sets (in the Data Partition node, set the method to Stratified and use the ordinal target as the stratification variable). The Neural Network and Regression nodes do not require the number of the decisions to equal the number of ordinal target levels.

%EMTP Macro

You can use the %EMTP macro to create target profile definitions for a data source. Use this macro when you want to create models that are weighted by the values of decisions. For more information about the %EMTP Macro, see SAS Enterprise Miner %EMTP Target Profiler Macro on page 1441 .

Part 7

User Interface

Chapter 18
SAS Enterprise Miner User Interface Help

SAS Enterprise Miner User Interface Help

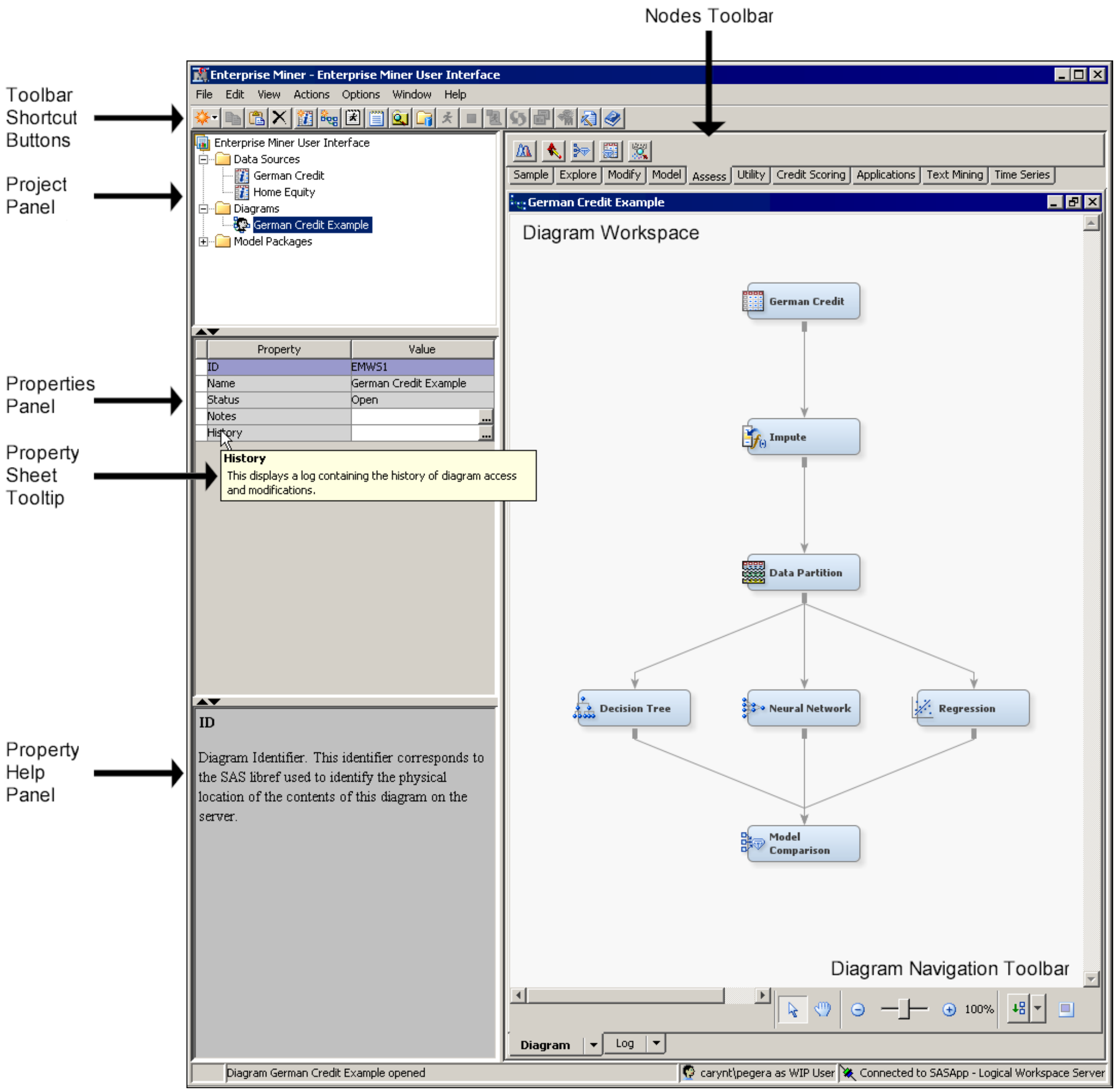
Overview of the SAS Enterprise Miner Workspace

Layout of the SAS Enterprise Miner Workspace

This section covers the user interface components and common window navigation tasks. When you understand the main parts of SAS Enterprise Miner software, you might want to explore “Working with Projects” on page 226 . Working with Projects explains how to create and manage data mining projects. In-depth information is available about SAS Enterprise Miner's node tools and process flow diagrams in the “Using the Diagram Editor” on page 240 section. See the Glossary for definitions of unfamiliar terms.

SAS Enterprise Miner User Interface

Figure 18.1 *SAS Enterprise Miner User Interface*



Property	Value
ID	EMWS1
Name	German Credit Example
Status	Open
Notes	
History	

History
This displays a log containing the history of diagram access and modifications.

ID

Diagram Identifier. This identifier corresponds to the SAS libref used to identify the physical location of the contents of this diagram on the server.

Diagram German Credit Example opened | carynt\pegera as WIP User | Connected to SASApp - Logical Workspace Server

Toolbar

The SAS Enterprise Miner Toolbar is a graphic set of node buttons and tools that you use to build process flow diagrams in the Diagram Workspace. The text name of any node or tool button is displayed when you position your mouse pointer over the button.

Toolbar Shortcut Buttons

The SAS Enterprise Miner toolbar shortcut buttons are a graphic set of user interface tools that you use to perform common computer functions and frequently used SAS Enterprise Miner operations. The text name of any tool button is displayed when you position your mouse pointer over the button.

Project Panel

Use the Project Panel to manage and view data sources, diagrams, results, and project users.

Properties Panel
Use the Properties Panel to view and edit the settings of data sources, diagrams, nodes, results, and users.

Diagram Workspace
Use the Diagram Workspace to build, edit, run, and save process flow diagrams. In this workspace, you graphically build, order, and sequence the nodes that you use to mine your data and generate reports.

Property Help Panel
The Help Panel displays a short description of the property that you select in the Properties Panel. Extended help can be found in the Help main menu.

Common Windows Tasks in SAS Enterprise Miner

SAS Enterprise Miner behaves like most Windows applications:

- Panels can be resized by dragging the edge of the panel.
- Windows within the Diagram Workspace can be resized by dragging window corners.
- Windows can be minimized, maximized, and closed by using the three control buttons in the upper right corner of all windows.
- A single click with the left mouse button selects objects in the Diagram Workspace.
- Right-clicking an object opens a pop-up menu of actions, if any are associated with that object.
- Text-based application menus can be activated by pressing the <ALT> key in combination with the underscored letter in the menu name.
- The name or function of most buttons appears in text if you position your mouse pointer over the button. To change the behavior of tooltips in SAS Enterprise Miner, select **Options** ⇨ **Preferences**. Then use the **Property Sheet Tooltips** and the **Tools Palette Tooltips** properties to configure the tooltip behavior.

  Note: The Nodes Toolbar and the Toolbar Shortcut Buttons together are collectively called the Tools Palette.

SAS Enterprise Miner Menus

Here is a summary of the SAS Enterprise Miner window menus that are available:

File

- **New**
  - **Project** — creates a new project.
  - **Diagram** — creates a new diagram.
  - **Data Source** — creates a new data source using the Data Source wizard.
  - **Library** — creates, modifies, or deletes a Library using the Library wizard.
- **Open Project** — opens an existing project. You can also create a new project from the Open Project window.
- **Recent Projects** — lists the projects on which you were most recently working.
- **Open Model** — opens a model package SPK file that you have previously created.
- **Open Model Package** — opens a window that you can use to view and compare model packages.

- **Register Model** — registers model.
- **Open Diagram** — opens the diagram that you select in the Project Panel.
- **Close Diagram** — closes the open diagram that you select in the Project Panel.
- **Close this Project** — closes the current project.
- **Import Diagram from XML** — imports a diagram that has been defined by an XML file.
- **Save Diagram** — saves a diagram as an image (BMP or GIF) or as an XML file.
- **Print Diagram** — prints the contents of the window that is open in the Diagram Workspace.
- **Print Preview Diagram** — creates an onscreen preview of the contents of the window that is selected for printing.
- **Delete this Project** — deletes the current project.
- **Exit** — ends the SAS Enterprise Miner session and closes the window.

Edit

- **Cut** — deletes the selected item and copies it to the clipboard.
- **Copy** — copies the selected node to the clipboard.
- **Paste** — pastes a copied object from the clipboard.
- **Delete** — deletes the selected diagram, data source, or node.
- **Rename** — renames the selected diagram, data source, or node.
- **Duplicate** — creates a copy of the selected data source.
- **Select All** — selects all of the nodes in the open diagram.
- **Clear All** — clears text from the Program Editor.
- **Find/Replace** — opens the Find/Replace window so that you can search for and replace text in the Program Editor, Log, and Results windows
- **Go To Line** — opens the Go To Line window. Enter the line number on which you want to enter text.
- **Layout**
  - **Horizontally** — creates an ordered, horizontal arrangement of the layout of nodes that you have placed in the Diagram Workspace.
  - **Vertically** — creates an ordered, vertical arrangement of the layout of nodes that you have placed in the Diagram Workspace.
- **Zoom** — increases or decreases the size of the process flow diagram within the diagram window by the amount that you choose.
- **Copy Diagram to Clipboard** — copies an image of the current process flow diagram in the Diagram Workspace to the Windows clipboard.

View

- **Program Editor** — opens a SAS Program Editor window in which you can enter SAS code.
- **Project Log** — opens a Project Log window.
- **Explorer** — opens a SAS Explorer window. You use the Explorer window to view SAS Libraries and their tables.

- **Metadata** — opens a SAS Explorer window that you can use to open metadata information for projects and results.
- **Refresh Project** — updates the selected project tree to incorporate any changes that were made to the project from outside the SAS Enterprise Miner user interface.

Actions

- **Add Node** — adds a node that you select to the Diagram Workspace.
- **Select Nodes** — opens the Select Nodes Window.
- **Connect Nodes** — opens the Connect Nodes Window. You must select a node in the Diagram Workspace to make this menu item available. You can connect the node that you select to any nodes that have been placed in your Diagram Workspace.
- **Disconnect Nodes** — opens the Disconnect Nodes window. You must select a node in the Diagram Workspace to make this menu item available. You can disconnect the node that you select from any nodes that are connected to the selected node in your Diagram Workspace.
- **Update** — updates the selected node to incorporate any changes that you have made.
- **Run** — runs the selected node and any predecessor nodes in the process flow that have not been executed, or submits any code that you type in the Program Editor window.
- **Stop Run** — interrupts a currently running process flow.
- **Stop Code** — interrupts currently running code.
- **View Results** — opens the Results window for the selected node.
- **Create Model Package** — generates a mining model package. See "Exporting the Results" on page 1445 in the Model Repository documentation for more information.
- **Export Path as SAS Program** — saves the path that you select as a SAS program. In the window that opens, you can specify the location to which you want to save the file, and whether you want the code to run the path or create a model package.

Options

- **Preferences** — opens the Preferences window. Use the following options to change the user interface:

  User Interface

  - **Property Sheet Tooltips** — This setting specifies whether tooltips are displayed on various property sheets that appear throughout the user interface.
  - **Tools Palette Tooltips** — This setting specifies whether tooltips display the tool name only, display the tool name and a description, or suppress tooltips for the tool buttons in the tools palette.
  - **Open Last Opened Project Automatically** — specifies whether SAS Enterprise Miner should open the last opened project when you log on.
  - **Open Last Viewed Diagram Automatically** — specifies whether SAS Enterprise Miner should open the last viewed diagram when you log on. You must specify **Yes** in both this property and **Open Last Opened Project Automatically** to automatically open the last viewed diagram.
  - **Number of Recent Projects** — specifies the number of projects that appears in the **Recent Projects** list.

Interactive Sampling

- **Sample Method** — specifies whether default samples are made from the top of the data set or are drawn at random when exploring data.
- **Fetch Size** — specifies the amount of data to fetch during interactive sampling. You can choose from **Default** (2000 observations) or **Max** (all observations).
- **Random Seed** — lets you specify the default value for Random Seed entries.

Model Package Options

- **Generate C Score Code** — specifies whether to generate and add C Score code to reports and Results packages
- **Generate Java Score Code** — specifies whether to generate and add Java Score code to Results packages
- **Java Score Code Package** — specifies the Java package name to be used when SAS Enterprise Miner is configured to generate and add Java Score code to Results packages.

Run Options

- **Grid processing** — specifies whether to use grid processing when available or to never use grid processing. If you enable Grid processing for a SAS Enterprise Miner project, you must specify a project location that is common to all Grid nodes.

Results Options

- **Log/Output Line Numbers** — specifies the number of lines that are displayed in any logs or outputs.

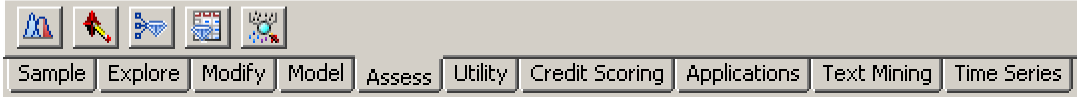
Window

- **Tile** — displays windows in the Diagram Workspace so that all windows are visible at the same time.
- **Cascade** — displays windows in the Diagram Workspace so that windows overlap.

Help

- **Contents** — opens the SAS Enterprise Miner Help window, which enables you to view all the SAS Enterprise Miner Reference Help.
- **Component Properties** — opens a table that displays the component properties of each tool.
- **Generate Sample Data Sources** — creates sample data sources that you can access from the Data Sources folder.
- **Generate Diagram Templates** — creates sample process flow diagrams that introduce some common data mining tasks.
- **Configuration** — displays the current system configuration of your SAS Enterprise Miner session.
- **About** — displays information about the version of SAS Enterprise Miner that you are using.

The SAS Enterprise Miner Node Toolbar



The SAS Enterprise Miner Node Toolbar is located directly above the Diagram Workspace. It is a tabbed graphic collection of SAS Enterprise Miner nodes and tools that you use to build process flow diagrams.

If you position your mouse pointer over a Toolbar button, a text ToolTip appears and displays the node or tool name.

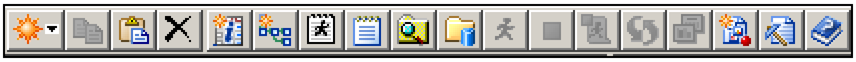
To use a Toolbar node in a process flow diagram, click the node button and drag it into the Diagram Workspace. The Toolbar button remains in place and the node in the Diagram Workspace is ready to be connected and configured for use in your process flow diagram. For more information about manipulating the nodes, see "Using the Diagram Editor" on page 240.

Note: SAS Text Miner and Credit Scoring for SAS Enterprise Miner are not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner and SAS Text Miner, those data mining tools do not appear in your SAS Enterprise Miner software.

Toolbar Shortcut Buttons

Shortcut buttons for the Toolbar are found at the top left portion of the SAS Enterprise Miner User Interface window, above the Project Navigator:

Display 18.1 *Toolbar Shortcut Buttons*



TIP Tooltips appear when you position your pointer over a button.

- **New** — opens a pop-up menu to create a new SAS Enterprise Miner Project, Diagram, Data Source, or Library
- **Copy** — copies the selected item to the Windows clipboard.
- **Paste** — pastes the contents of the Windows clipboard to the selected area.
- **Delete** — deletes the selected item.
- **Create Data Source** — creates a new data source.
- **Create Diagram** — creates a new diagram.
- **Program Editor** — opens the Program Editor window.
- **Log** — opens the Log window.
- **Explorer** — opens a SAS Explorer window that you can use to browse SAS libraries and tables.
- **Metadata** — opens a SAS Explorer window that you can use to open metadata files.

- **Run** — runs the process flow from the selected node, or submits the code in the Program Editor window.
- **Stop run** — stops a running process.
- **Stop code** — stops running code.
- **Update** — updates the selected path and the SAS log.
- **Results** — opens the Results window for the selected node.
- **Create Model Package** — generates a model package from any results that were created previously.
- **Open Model** — opens a file browse window that you can use to locate and open model packages.
- **Contents** — opens the SAS Enterprise Miner Reference Help.

The Project Panel

Use the Project Panel to manage and view data sources, diagrams, model packages, and list users. The Project Panel contains the following folders:

- **Data Sources** — The Data Sources folder displays the data sources that have been defined for the project. Selecting a data source in the directory displays the data source properties in the Properties Panel. To use a data source in a process flow diagram, drag the data source from the Data Sources directory into the Diagram Workspace.
- **Diagrams** — The Diagrams folder lists the various diagrams that are associated with the project. To open a diagram in the Diagram Workspace, select a diagram in the Project Panel and select **File** ⇨ **Open Diagram** from the main menu. There is no limit on the number of diagrams that you can open at one time.
- **Model Packages** — The Model Packages folder lists all the results that you have generated by creating a report. You can right-click the results to register the results to the model repository or to save the results to a local directory.

For more information, see “Working with Projects” on page 226 .

The Properties Panel

Use the Properties Panel to view and edit the settings of data sources, diagrams, nodes, results, and users.

You can edit most of the properties of each node. Some properties display information only and cannot be edited. To edit the settings in the Properties Panel, click the appropriate row in the Value column. If the value can be edited, enter the appropriate value, or select a value from the drop-down list.

The Properties Panel also responds to some common keyboard navigation techniques.

- For properties that have an ellipsis button, pressing the spacebar emulates left-clicking on the ellipsis button. These properties lead to additional property groups, interactive applications, and other features. Pressing the spacebar to start the appropriate action.

- Property groups that are collapsible are indicated by a [+] or [-] next to the property name. You can expand or collapse a property group by pressing the spacebar.
- Hold the control key and press the up arrow to move to the top of the Properties Panel. Similarly, hold the control key and press the down arrow to move to the bottom of the Properties Panel.
- Use the Page Up and Page Down buttons to jump to the next property group.

The Diagram Workspace

Use the Diagram Workspace to build, edit, and run process flow diagrams. Note the following features of the Diagram Workspace:

- You can resize any of the Diagram Workspace windows by placing the mouse pointer on the edge of the window, and dragging the window to the size that you want.
- To view a large process flow diagram, use the scroll bars at the right side and the bottom of the window. Alternatively, you can decrease the magnification of the diagram by selecting Zoom from the Edit menu.
- To perform various tasks, use the menus, the toolbar at the top of the window, or the Diagram Workspace pop-up menus.

Using the Diagram Workspace Pop-Up Menus

To open a pop-up menu, right-click in an open area of the Diagram Workspace. Note that you can also perform many of these tasks by using the menus. The pop-up menu contains the following items:

- **Add node** — adds a node from the toolbar to the Diagram Workspace.
- **Paste** — pastes a copied node from the clipboard to the Diagram Workspace.
- **Select All** — selects all nodes in the process flow diagram.
- **Select Nodes** — opens the Select Nodes window in which you can select from a list of nodes that are in your diagram workspace.
- **Layout** — creates an ordered, horizontal or vertical arrangement of the nodes in the Diagram Workspace.
- **Zoom** — increases or decreases the size of the process flow diagram within the diagram window by the amount that you choose.
- **Copy Diagram to Clipboard** — copies the selected process flow diagram in the Diagram Workspace to the Windows clipboard.
- **Reset Diagram** — forces SAS Enterprise Miner to reset the property sheet for any node that was running when SAS Enterprise Miner was terminated. If SAS Enterprise Miner was inadvertently terminated while a node was running, it was impossible edit the properties of that node until the diagram lock expired. This selection manually resets the diagram lock and enables editing of the node's properties. For more information, see "Locked Data Sources and Diagrams " on page 283 .

Working with Projects

Overview

A project is a collection of SAS Enterprise Miner process flow diagrams and information that pertains to them. Projects are often differentiated by the type of data that you intend to analyze. In other words, it is a good idea to create a separate project for each major data mining problem that you intend to investigate. The following sections provide information about your projects and describe the steps that you take to create, modify, and delete projects.

Project Directory Structure

SAS Enterprise Miner uses a predefined directory structure and stores all of the project files in the project location (the root project directory).

Project Location

The directory that is created to contain your project contains four subdirectories:

DataSources Subdirectory

The DataSources subdirectory contains all available data source definitions. The actual data sets do not reside in this subdirectory, but in a separate directory. The DataSources subdirectory contains table and column metadata about each data set that you define. In addition, the subdirectory contains files that describe the data set properties and any notes that you entered about the data set.

Reports Subdirectory

The Reports subdirectory contains any reports that you created. Each report contains three files:

- **`miningresult.sas7bcat`**
- **`miningResult.spk`**
- **`miningResult.xml`**

System Subdirectory

The System subdirectory contains the start and exit SAS code that you entered when you created the project.

Workspaces Subdirectory

The Workspaces subdirectory contains all process flow diagrams that you created in the project.

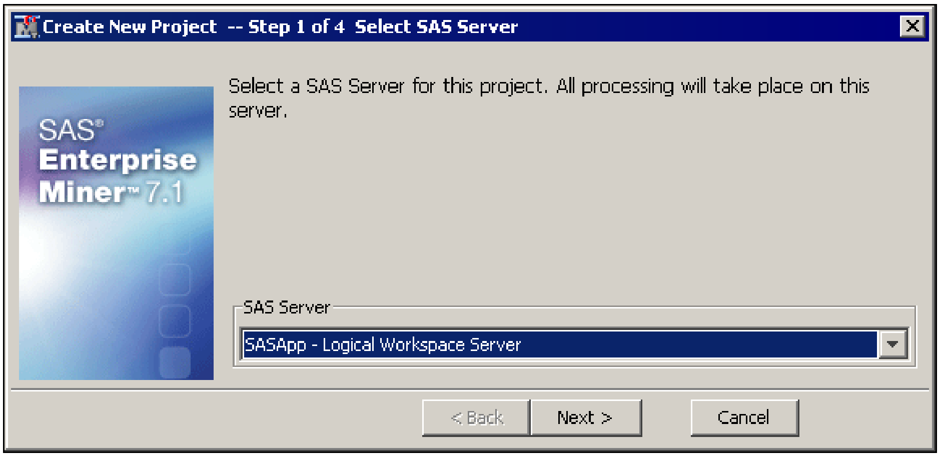
Each diagram ID folder contains all files that were created within a single process flow diagram, including all SAS data sets from all nodes. Each diagram ID folder also includes folders for each node in the process flow diagram. For example, a folder that is named **`Emws/Reg`** contains all files that are created by the Regression node, such as the training code, the scoring code, the log, and the output.

Creating a New Project

To create a new project, follow these steps:

1. From the main menu, select **File** ⇨ **New** ⇨ **Project**.

   The Create New Project window appears:

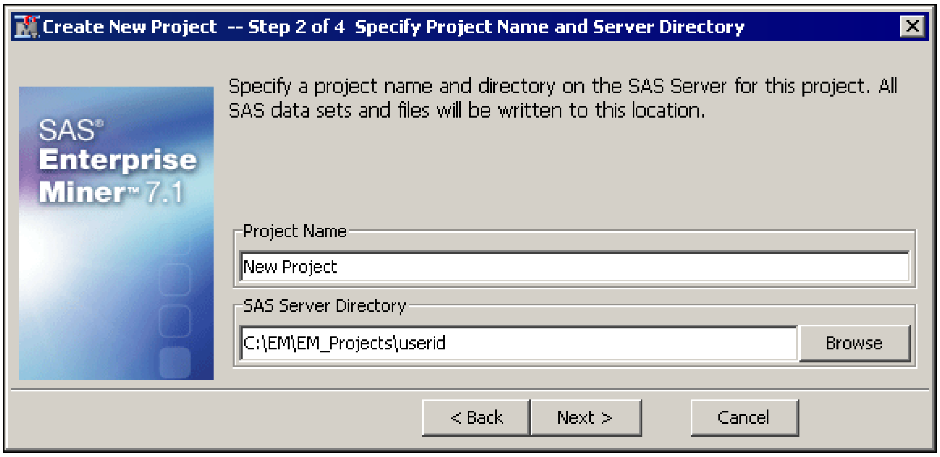


Create New Project -- Step 1 of 4 Select SAS Server

Select a SAS Server for this project. All processing will take place on this server.

SAS Server: SASApp - Logical Workspace Server

2. Select an available server for your project. Click **Next**.



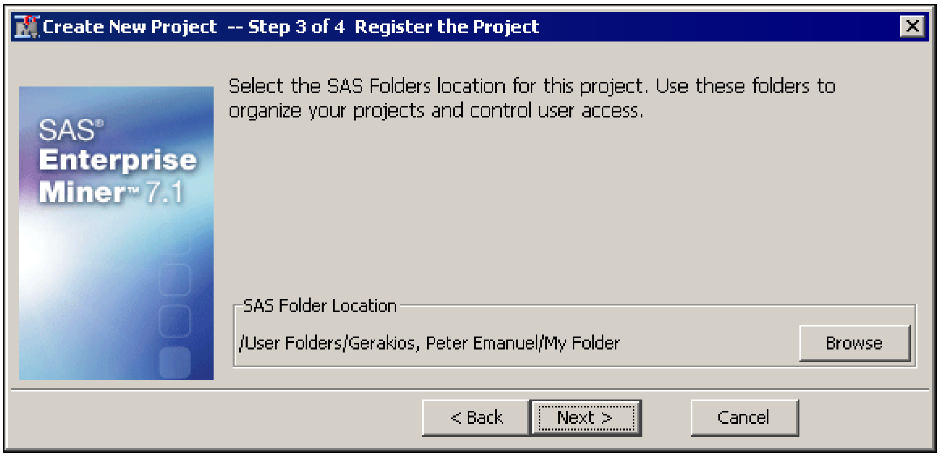
Create New Project -- Step 2 of 4 Specify Project Name and Server Directory

Specify a project name and directory on the SAS Server for this project. All SAS data sets and files will be written to this location.

Project Name: New Project

SAS Server Directory: C:\EM\EM_Projects\userid

3. Enter the name of your project in the Project Name field, and specify the server directory in the SAS Server Directory field. Click **Next**.

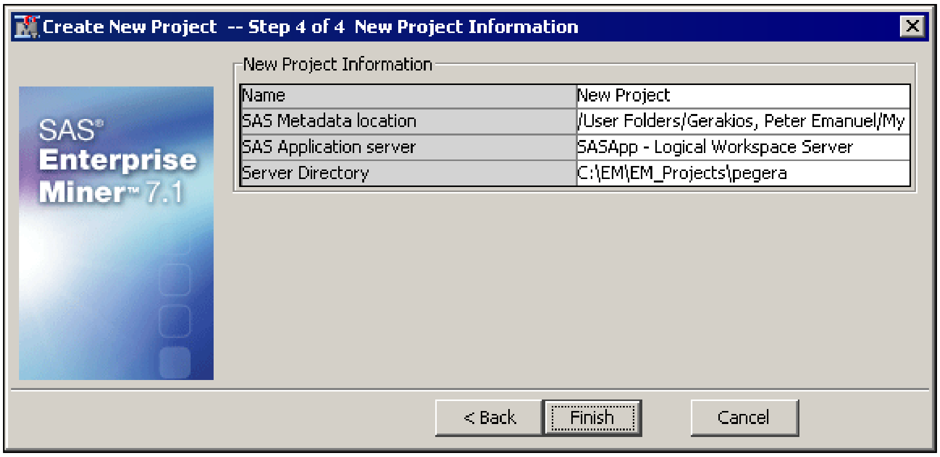


Create New Project -- Step 3 of 4 Register the Project

Select the SAS Folders location for this project. Use these folders to organize your projects and control user access.

SAS Folder Location: /User Folders/Gerakios, Peter Emanuel/My Folder

4. Select a location for the project. Click **Next**.


Create New Project -- Step 4 of 4 New Project Information
SAS® Enterprise Miner™ 7.1
< Back
Finish
Cancel

New Project Information

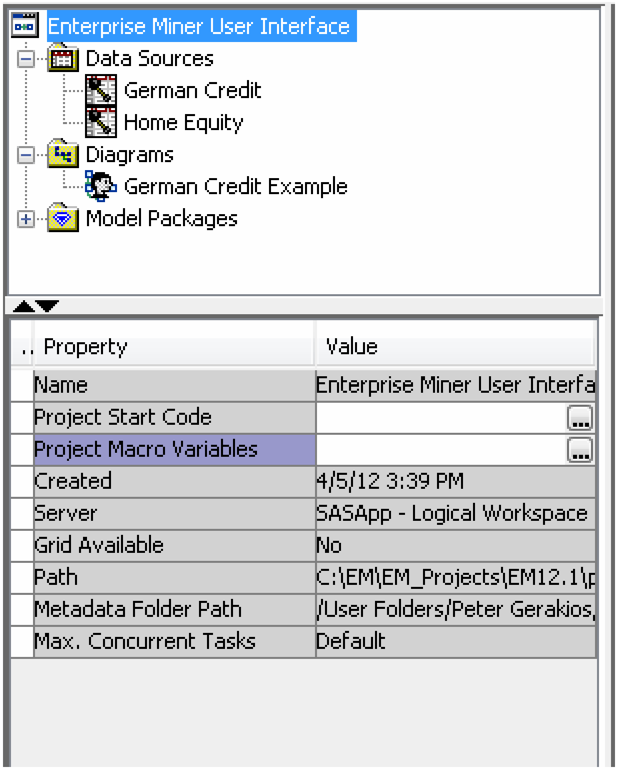
Name	New Project
SAS Metadata location	/User Folders/Gerakios, Peter Emanuel/My
SAS Application server	SASApp - Logical Workspace Server
Server Directory	C:\EM\EM_Projects\pegera

5. Review the information that you specified. Click **Finish**.

Project Start Code

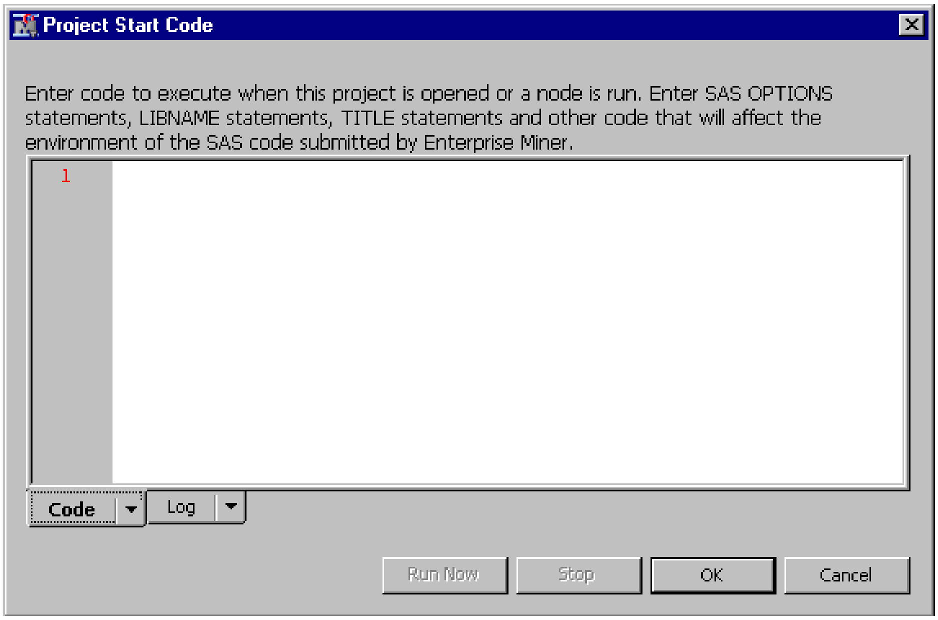
You can specify SAS code that executes every time you start the project.

1. Click the project tree name (Enterprise Miner User Interface is the project name in this example) to display the project properties panel.



Property	Value
Name	Enterprise Miner User Interfa
Project Start Code	...
Project Macro Variables	...
Created	4/5/12 3:39 PM
Server	SASApp - Logical Workspace
Grid Available	No
Path	C:\EM\EM_Projects\EM12.1\p
Metadata Folder Path	/User Folders/Peter Gerakios,
Max. Concurrent Tasks	Default

2. Click the ... button to the right of the **Project Start Code** property to open the Project Start Code window.



Enter code to execute when this project is opened or a node is run. Enter SAS OPTIONS statements, LIBNAME statements, TITLE statements and other code that will affect the environment of the SAS code submitted by Enterprise Miner.

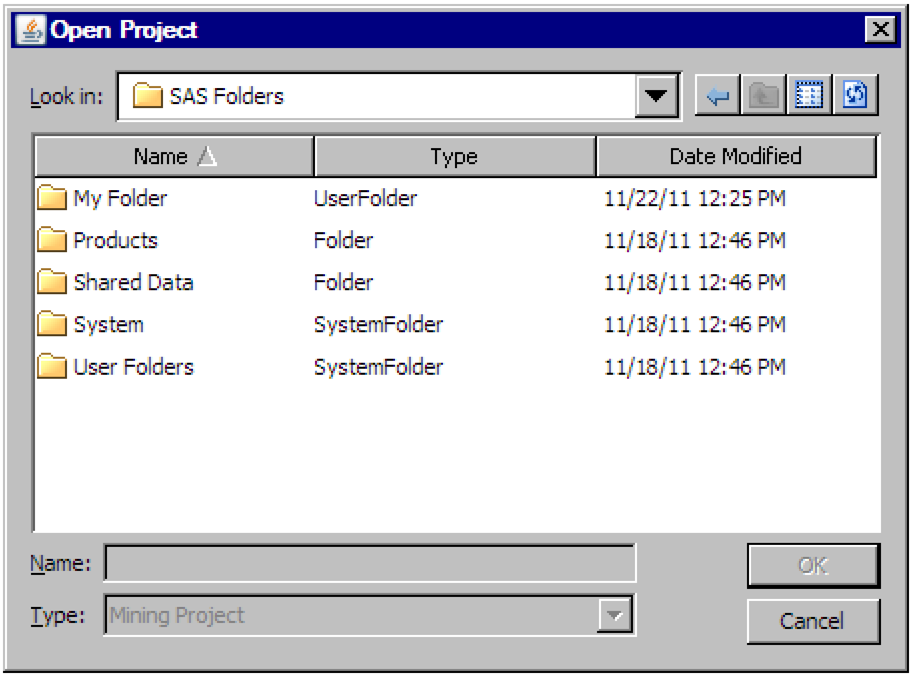
3. Enter the SAS code that you want to run each time the project starts, such as LIBNAME statements. When you have finished entering all of the project start code that you want, click **Run Now**. Click **OK** to close the Project Start Code window.

Note: You can modify your project start code by repeating the previous three steps.

Opening an Existing Project

If you want to open a project on which you have recently worked, you can select **File** ⇨ **Recent Projects**. Then choose the project that you want to open from the list.

To open an existing project, select **File** ⇨ **Open Project** from the main menu. The Open Project window appears.



Look in: SAS Folders

Name	Type	Date Modified
My Folder	UserFolder	11/22/11 12:25 PM
Products	Folder	11/18/11 12:46 PM
Shared Data	Folder	11/18/11 12:46 PM
System	SystemFolder	11/18/11 12:46 PM
User Folders	SystemFolder	11/18/11 12:46 PM

Name:

Type: Mining Project

Use the **My Folder** drop-down list to navigate to the project that you want to open. To open a project, click the project, and click **OK**.

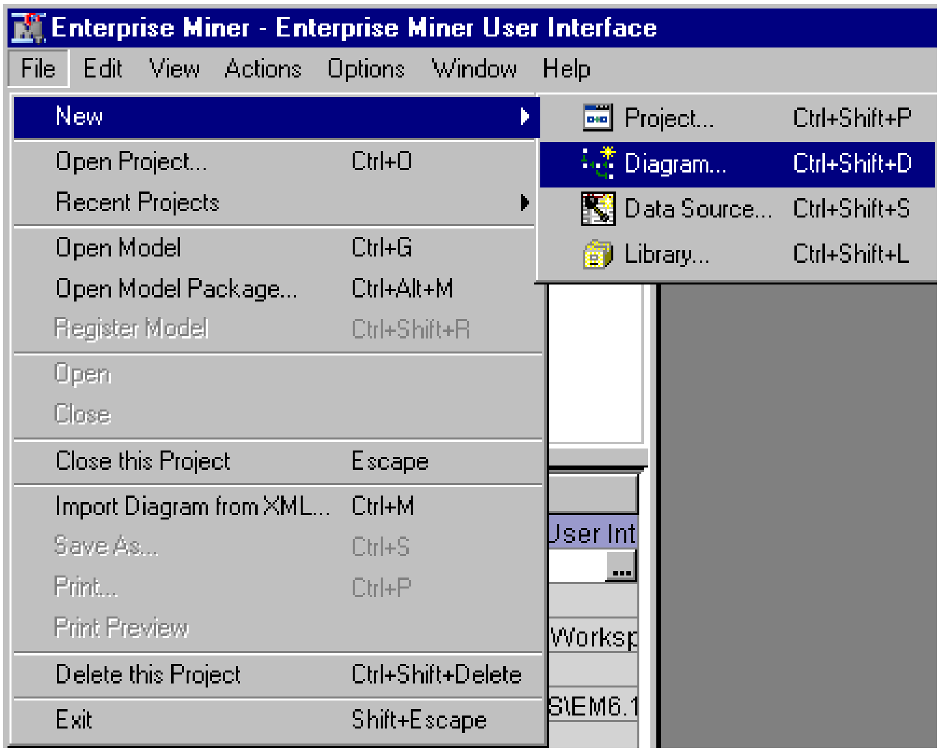
The Open Project window contains the following options:

- Back — select to go to the previous view in the Open Project window.
- Up one level — select to go up one level in the project directory.
- Views — select to choose whether to view project directory entries as a list or with details added. Select **List** to view just the names of projects or folders. Select **Details** to view the type of an entry and the date it was last modified in addition to its name.
- Refresh — select to refresh the file and folder display in the project directory.

Create a New Project Diagram

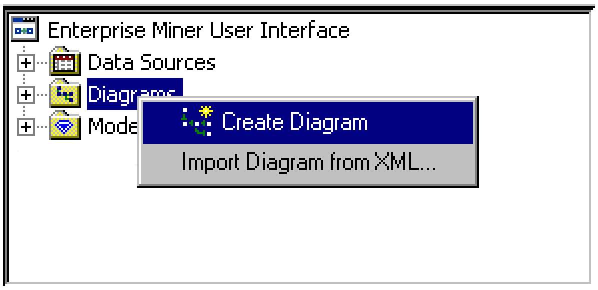
To create a new project diagram, select **File** ⇨ **New** ⇨ **Diagram** from the main menu.

Display 18.2 *New Project Diagram*



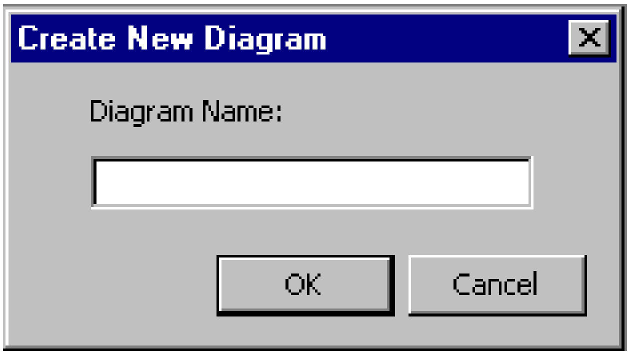
Alternatively, right-click the Diagrams folder in the project tree, and select **Create Diagram**.

Display 18.3 *Alternative New Project Diagram*



The Create New Diagram window appears.

Display 18.4 *Create New Diagram Window*



Enter a name for your diagram in the Diagram Name field.

Display 18.5 *Enter a New Name*

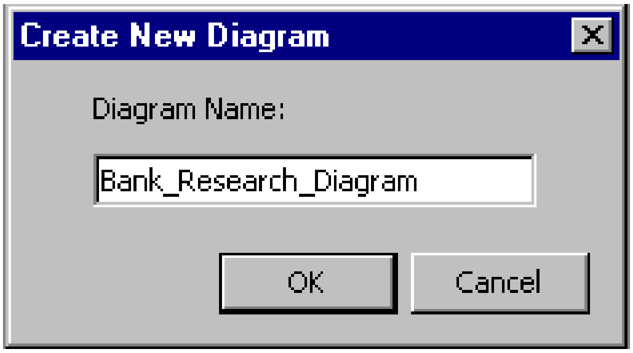
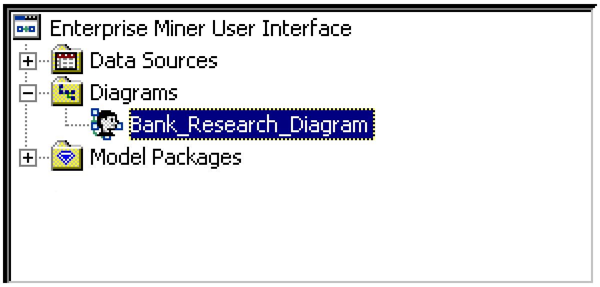


Diagram Name: Bank_Research_Diagram

Click **OK** to create the diagram. The new diagram opens in the Diagram Workspace, and appears in the project tree under Diagrams.

***Display 18.6** Created Diagram*

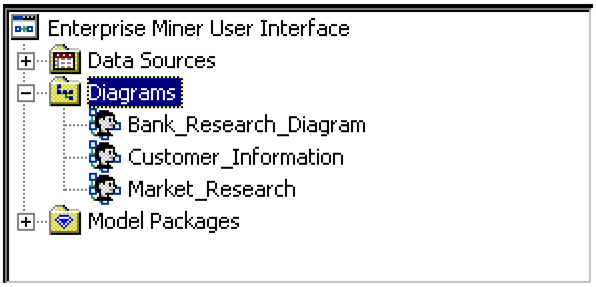


- Enterprise Miner User Interface
  - Data Sources
  - Diagrams
    - Bank_Research_Diagram
  - Model Packages

Opening an Existing Project Diagram

To open a project diagram, expand the Diagrams folder in the Project Panel.

***Display 18.7** Diagrams Folder*



- Enterprise Miner User Interface
  - Data Sources
  - Diagrams
    - Bank_Research_Diagram
    - Customer_Information
    - Market_Research
  - Model Packages

Use any of the following methods to open a diagram:

- Double-click the diagram name.
- Right-click the diagram name and select **Open** from the menu.

Reordering Project Diagrams

To rearrange the order of the diagrams in the project panel, click and drag a project from its original position to its new position in the list of diagrams. This enables you to arrange the diagrams in your project in the order is most efficient for you. By default, diagrams are sorted alphabetically.

Running a Project Diagram

To generate results from a process flow diagram in your Diagram Workspace, you must run the process flow path to execute the code that is associated with each node. The paths can be run in several ways:

- Right-click the node that you want to run and click **Run**. All previous nodes in the process flow diagram will run if they have not already run, or will run if they are set to re-run.
- Select the node you that want to run and click 🏃 from the toolbar shortcut buttons. All previous nodes in the process flow diagram will run if they have not already run, or will run if they are set to re-run.
- Select the node that you want to run, and from the main menu select **Actions** ⇨ **Run**. All previous nodes in the process flow diagram will run if they have not already run, or will run if they are set to rerun.

Saving a Project Diagram

You do not need to save your project diagrams. Changes that you make to your process flows and nodes are saved automatically.

Closing a Project Diagram

To close an open project diagram, select the Diagram window in the Diagram Workspace. Then from the main menu select **File** ⇨ **Close Diagram "Diagram-name"**. You can also close a diagram by closing the Diagram window.

Deleting Projects and Diagrams

To delete a project, open the project you want to delete, and then select **File** ⇨ **Delete this Project** from the main menu. A message window asks you to confirm your choice. To delete a diagram, right-click the diagram name in the Project Panel and select **Delete**.

Project Metadata

Using the Metadata Explorer, you can view, delete, and rename projects and results on the metadata server. To access the Metadata Explorer, select **View** ⇨ **Metadata** from the main menu. The Metadata Explorer communicates only with the metadata server and displays only mining projects and mining results.

To open a project from its metadata, right-click on a mining project and select **Open** from the pop-up menu. Additionally, you can select **File** ⇨ **Open Project** from the main menu to open a project.

To open the metadata for a mining result, right-click on a mining result and select **Open** from the pop-up menu. This opens a new window with all the metadata for that set of results. Inside this window, there are five windows each containing information created or needed by the scoring process. These windows are:

- **Properties** — contains project information. This includes the project name, location on the server, node or nodes scored, user name, and date of creation.
- **Output Table** — contains the output variables, a brief description of each, the length of each column, the SAS Format of each variable, the SAS Informat of each variable, and the SAS Column Type of each variable.
- **SAS Score Code** — contains the SAS code for the scoring process.
- **Input Table** — contains the input variables, a brief description of each, the length of each column, the SAS Format of each variable, the SAS Informat of each variable, and the SAS Column Type of each variable.
- **Target Table** — contains the target variables, a brief description of each, the length of each column, the SAS Format of each variable, the SAS Informat of each variable, and the SAS Column Type of each variable.

To rename a project or result, right-click on the item and choose **Rename** from the pop-up menu. This makes project or result name editable.

Similarly, to delete a project or result, right-click on the item and choose **Delete** from the pop-up menu. You will then be prompted to confirm that you want to delete the metadata for that project or result. Note that this will delete only the project from the metadata. Your project workspace files will still be located on your SAS server.

Project Panel Pop-Up Menus

You can perform a number of actions in the Project Panel by using the pop-up menus that are accessible with the right mouse button. The following table describes the menus that are available for each item in the Project Panel.

Right-click these items.	The following menu items are then available.
Project Name	• **Create data source** — opens the **Data Source** wizard. • **Create diagram** — opens the Create New Diagram window, where you specify a name of a diagram to be added to the Diagrams folder. • **Import diagram from XML** — opens a browsing window where you can select a diagram that had been previously saved as an XML file to import. • **Refresh Project Tree** — updates the project tree display.
Data Sources	**Create Data Source** — opens the **Data Source** wizard.
Data Source Name	• **Rename** — enables you to enter a new name for the data source. • **Duplicate** — creates a copy of the data source. • **Delete** — deletes the data source. • **Edit Variables** — opens the Variables table in which you can modify the various aspects of the variables and access the Explore functionality. • **Refresh Metadata** — refreshes the metadata that is associated with a data source. • **Edit Decisions** — opens the Decision Processing window. You can change any decisions that you have previously defined. • **Explore** — opens the table for browsing and creating graphs within the Explore window.
Diagrams	• **Create Diagram** — creates a new diagram in the folder. • **Import Diagram from XML** — enables you to import a diagram that had been previously saved as an XML file.

Right-click these items.	The following menu items are then available.
Diagram Name	• **Open** — opens the diagram in the diagram workspace. • **Create Diagram** — creates a new diagram in the folder. • **Import Diagram from XML** — enables you to import a diagram that had been previously saved as an XML file. • **Rename** — enables you to assign a new name to the diagram. • **Delete** — deletes the selected diagram. • **Close** — closes the open, selected diagram. • **Save As** — enables you to save the diagram as an **Image** or as an **XML Document**. • **Print** — prints the diagram image. • **Print Preview** — displays a preview of the diagram image prior to printing.
Model Packages	• **Open Model** — opens the Open Model window.
Model Package Name	• **Open** — opens the model package in the Package window. • **Delete** — deletes the model package from the project. • **Register** — registers the model package to the model repository. • **Recreate Diagram** — opens a new diagram that is a copy of the diagram in the model package. • **Save As** — enables you to save the SPK file to a new directory.

Viewing Project Panel Properties

Each of the items in the Project panel has associated properties. Click the item to view the properties in the Properties panel.

- Project Properties on page 235
- Data Source Properties on page 236
- Diagram Properties on page 237
- Model Packages Properties on page 237

Project Properties

Click the project name in the Project panel to view the following properties:

- **Name** — the project name.

- **Project Start Code** — the code that is executed when the project is opened.
- **Project Macro Variables** — the project macro variables. For more information, see SAS Enterprise Miner Project Macro Variables on page 238 .
- **Created** — when the project was created.
- **Server** — the server where the project is located and where the SAS data mining procedures are run.
- **Grid Available** — whether the server is configured to enable you to use grid processing.
- **Path** — the physical location of the project directory on the server.
- **Metadata Folder Path** — the physical location of the metadata folder.
- **Max. Concurrent Tasks** — the maximum number of parallel processes that can be implemented when process flows are run.

Data Source Properties

Click a data source name to view the following properties:

- **ID** — the data source identifier. The ID value is the assigned libref to the SAS library in which the metadata tables are stored.
- **Name** — the data source name.
- **Variables** — characteristics of the columns of the physical table. The Explore functionality to view variable distribution is available here.
- **Decisions** — decision processing. For more information, see the "Enterprise Miner Target Profiler" on page 207 documentation.
- **Role** — the role of the table.
- **Notes** — a window in which you enter notes about the data source.
- **Library** — the SAS library that is used to access the table.
- **Table** — the name of the physical table that is used by the data source.
- **Sample Data Set** — the name of the sample data set created from the datasource table.
- **Size Type** — indicates the sample size type. This can either be a percentage of the data table or a discrete number of observations.
- **Sample Size** — either the proportion of observations for the number of observations included in the sample.
- **Type** — the member type. Valid type values are DATA or VIEW.
- **No. Obs.** — the number of rows in the table.
- **No. Cols**. — the number of columns in the table.
- **No. Bytes** — the number of bytes used for the data source.
- **Segment** — the segment that you define for the data source.
- **Created by** — the name of the user who created the data source.
- **Create Date** — the date on which the data source was created.
- **Modified by** — the name of the user who last modified the data source.
- **Modify Date** — the date on which the data source was last modified.

- **Scope** — indicates whether the data source is **local** (exists only in the project) or **global** (exists independently of the project and can be shared by multiple projects).

Diagram Properties

Click a diagram name to view the following properties:

- **ID** — the diagram identifier. The identifier corresponds to the SAS libref that is used to identify the physical location of the diagram contents on the server.
- **Name** — the diagram name.
- **Status** — the diagram status. Diagram states are:
  - **Available**
  - **Open**
  - **Locked**
- **Notes** — a window in which you enter notes about the diagram.
- **History** — the diagram history. The history is a log of all actions and modifications that were performed on the diagram.

Model Packages Properties

Click a model package name to view the following properties:

- **ID** — the ID that was generated when you created the model package.
- **Name** — the name of the mining model package.
- **Filename** — the physical location of the model package SPK file.
- **Mining Function** — the mining function. The available mining functions are as follows:
  - **None**
  - **Segmentation**
  - **Transformation**
  - **Classification**
  - **Prediction**
- **Mining Algorithm** — the mining algorithm that the model uses.
- **Target** — the target variable.
- **Version** — the version of SAS Enterprise Miner.
- **Diagram ID** — the ID that was assigned to the workspace that contains the model.
- **Node Description** — the name of the node from which the model package was generated.
- **Repositories** — the repositories to which the model has been registered. This field is not updated if the model has been deleted from a repository. This field indicates only that the model has been registered.
- **Created By** — the user ID of the user who created the report.
- **Created Date** — the date and time on which the report was created.

SAS Enterprise Miner Start Code

Start code is any SAS code that you want to run before SAS Enterprise Miner processing. This code might include any SAS system options, titles and footnotes, LIBNAME and filename statements, or SAS Enterprise Miner macro variables. Use this code to customize your SAS session. You should not enter code that runs SAS procedures or other potentially time-consuming operations. In particular, you should avoid the creation of unneeded SAS libraries that slow the creation of SAS sessions and thus reduce system responsiveness for SAS Enterprise Miner users.

There are two levels of start code, **server start code** and **project start code**.

- **Server start code** — Your server administrator manages this code, which is executed for all SAS Enterprise Miner users for any server. This code is stored in a code file accessible on the SAS server. The filename and location is entered into the SAS Enterprise Miner plug-in for the SAS Management Console.
- **Project start code** — Users can enter code that is executed for any process within the current project. This code is stored in the SAS Enterprise Miner project and is edited through the property sheet for the project.

Start code is executed in the following order:

- The SAS system `autoexec.sas` file.
- The SAS Enterprise Miner **server start code** file.
- The SAS Enterprise Miner **project start code** file.

SAS Enterprise Miner Project Macro Variables

The **Project Macro Variables** enables you to set the values of several SAS Enterprise Miner macro variables. The variables are separated into two categories, **EM General** and **HPDM**. Click the ... button in the **Project Macro Variables** property to open the Project Macro Variables window.

In the Project Macro Variables window, you can configure the following macro variables:

- **EM General**
  - **EM_NUMTASK** — controls the number of nodes that are run concurrently. Use this to control the compute load processed by your system. Possible values are nonnegative integers, SINGLE, MAX or MAXIMUM, and DEFAULT.

    If you specify 0, 1, or SINGLE, then the nodes in a process flow diagram will run sequentially. That is, parallel processing is not used.

    When you specify MAX or MAXIMUM, parallel processing is used and SAS Enterprise Miner attempts to run as many nodes as is possible in parallel. For example, if you have four nodes that can be run in parallel, the system will attempt to run all four nodes concurrently.

    If you specify DEFAULT, parallel processing is used to run nodes concurrently. If the SAS grid is not used, the maximum number of concurrent nodes is determined by the value of the automatic system macro variable SYSNCPU. If the SAS grid is used, the maximum number of concurrent nodes is the value returned by the GRDSVC_NNODES function.

  - **EM_VBUFSIZE** — This macro variable sets the buffer size when processing a view. The default value is 64M.
  - **EM_EXPLOREOBS_MAX** — controls the number of rows of data downloaded to the client for interactive graphics in the explore data actions. The value should be large enough to select a sample that accurately represents the population but not so large to overload your network and computer memory. The default value is dynamically determined by the record length of your selected data set. Set this value when you think the default value is not appropriate.
  - **EM_TRAIN_MAXLEVELS** — specifies the maximum number of levels that are allowed when computing the fit statistics with PROC DMDB. The default value is 512.
  - **EM_DECMETA_MAXLEVELS** — controls the maximum number of levels for any target variable for which a target profile can be built. The default value is 32.
  - **EM_PMML** — enables the generation of PMML score code by the nodes that support PMML. Valid values are Y, Yes, N, and No, and the default value is No.
  - **EM_ASSESS** — controls the execution of model assessment functions for modeling nodes. Valid values are Y or N, and Y is the default value. Specify N to suppress all assessment reports.
  - **EM_GROUPASSESS** — controls the execution of model assessment functions for model nodes within a group processing segment. The possible values are Y or N. The default value is Y. Use this setting to speed processing when you only want to see the model assessment details of the final population model, rather than the individual models.
  - **EM_MAXGROUPASSESS** — controls the maximum number of groups for which predictive model assessment is executed for a modeling node within a group processing segment. The default value is blank.
- **HPDM**
  - **HPDM_COMMIT** — determines whether the SAS log is updated and how often that it is updated. Specify a value here to indicate that the SAS log will produce a message whenever the number of observations sent from the client to the appliance for distributed processing exceeds an integer multiple of this value.
  - **HPDM_CPUCOUNTS** — specifies how many processors the High Performance procedures assume are available on each host in the computing environment. Valid values are any integer from 1 to 256.
  - **HPDM_NODES** — specifies the number of nodes in the distributed computing environment, provided that the data is not processed alongside the database. If you specify **0**, you indicate that you want to process the data in symmetric multiprocessing mode on the client. If the data is not read alongside the database, this option specifies the number of nodes on the appliance involved in the analysis.
  - **HPDM_NTHREADS** — specifies the number of threads that are used for analytic computations and overrides the SAS system option. If not specified, the number of threads is determined based on the number of CPUs on the host where the analytic computations execute.
  - **HPDM_SAMPLESIZE** — Specifies the sample size for the sample generated with HPDM data sources. The default is 10,000.

Using the Diagram Editor

Adding a Node

You create a process flow diagram by adding and connecting nodes in the Diagram Workspace. You connect nodes in a logical order that follows the SEMMA data mining methodology.

You can add nodes to a process flow diagram in the following ways:

- Drag node icons from the toolbar to the Diagram Workspace.
- From the main menu, select **Actions** ⇨ **Add node** to open a menu of nodes to add.
- Right-click the Diagram Workspace and select **Add node** from the menu.

Opening Node Pop-Up Menus

To open a node pop-up menu, right-click a node icon in the Diagram Workspace. The node pop-up menu contains the following items:

- Edit Variables... — opens the Variables window in which you can view and modify variable values. The values that are available to modify will vary from node to node. See documentation for the node whose variables you want to modify for information about which variable values you can modify in this window. In general, variables with a gray background are read-only. The Explore functionality for viewing variable distribution is available here.
- Update — updates the node and all previous nodes in the path to incorporate any changes that you made. Updating the path does not run the node.
- Run — runs the selected node and all predecessor nodes in the process flow diagram that have not been run, or are set to rerun.
- Create Model Package... — creates a model package that incorporates the results of the node and the previous nodes in the path.
- Results... — opens the Results window.
- Export Path as SAS Program — saves the path from the selected node and all previous nodes in the process flow as a SAS program.
- Cut — cuts the selected node and copies it to the clipboard.
- Copy — copies the selected node to the clipboard.
- Delete — deletes the selected node.
- Rename — renames the selected node.
- Select All — selects all nodes in the Diagram Workspace.

-

— opens the Select Nodes window, in which you can select any of the nodes in the Diagram Workspace. For more information, see "Using the Diagram Editor" on page 240.

-

— opens the Connect Nodes window, in which you can connect the selected nodes to any other nodes in the Diagram Workspace. For more information, see "Connecting Nodes" on page 242.

-

— opens the Disconnect Nodes window, in which you can disconnect the selected nodes to the nodes to which they are attached in the Diagram Workspace. For more information, see "Disconnecting Nodes" on page 244.

Variables Table

If you select the **Edit Variables** option in the node pop-up menu, you will open the variables table for that node. For all nodes other than input data source nodes, the variables table contains a column named **Use**. This column determines whether the variable is used as an input variable.

- **Yes** — This is the default value when the role of the variables is **Frequency** or the variable is the first target variable.
- **No** — This is the default value when there is more than one target variable, and the current variable is not the first target variable.
- **Default** — If a variable has a role of **Input**, then this is treated as **Yes** and if the variable has a role of **Rejected**, then this is treated as **No**. If there is more than one target variable, then this option is not available.

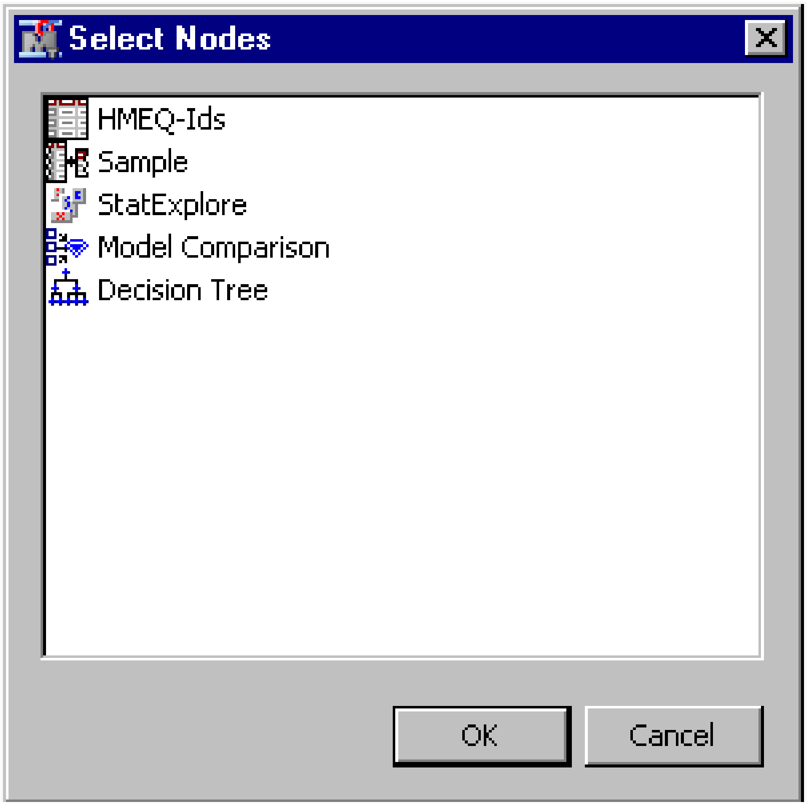
Selecting Nodes

You can select a node in the Diagram Workspace by clicking the node. Select multiple nodes by holding the CTRL key down and clicking the nodes that you want to select. You can select all of the nodes in the Diagram Workspace by right-clicking the Diagram Workspace and choosing **Select All** from the pop-up menu.

You can also select nodes by using the Select Nodes window. To open the **Select Nodes** window, right-click in the Diagram Workspace and choose **Select Nodes** from the pop-up menu.

You can select single or multiple nodes by using the Select Nodes window.

Display 18.8 *Select Nodes Window*



Click the node name in the window to select the node. You can select multiple nodes by holding the CTRL key down and then clicking on the nodes that you want to select. Also, you can select one node, hold the SHIFT key down, the click any other node to select all nodes between and including the two you clicked.

You can also use the up and down arrows on your keyboard to choose a node that you want to select. To select multiple nodes with this method, highlight the first node that you want to select, hold down the CTRL key, use the up and down arrows to move to another node, and then select that node by pressing the space bar. If you hold the SHIFT key while using the up and down arrow keys, you can select all nodes between your first and last node.

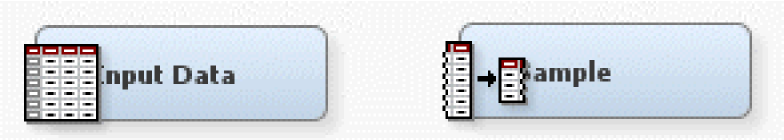
Click **OK** to select the nodes.

Connecting Nodes

Connect nodes in the Diagram Workspace to create a logical process flow. When you create a process flow, follow the SEMMA data mining methodology. Information flows from node to node following the connecting arrows.

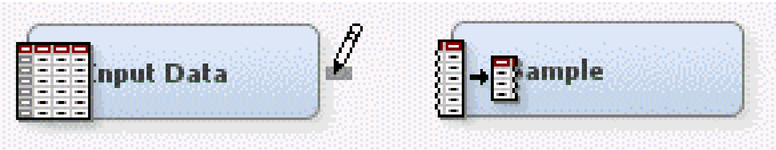
The following example connects the Input Data node to the Sampling node.

Display 18.9 *Input Data Node to the Sampling Node*



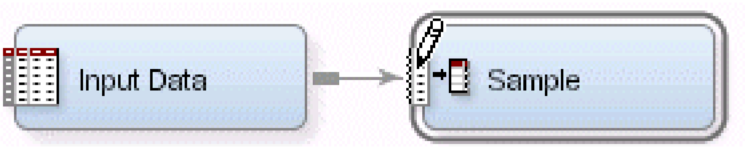
To connect the nodes, move the pointer to the right side of the Input Data node icon. The pointer icon changes to a pencil.

Display 18.10 *Connect Nodes*



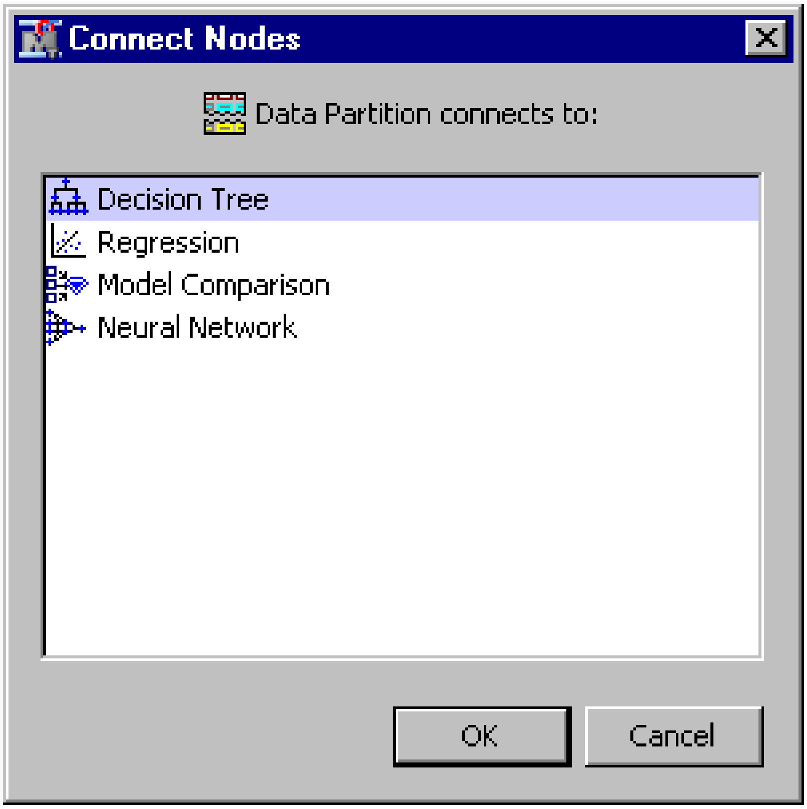
Drag the pointer to the Sample node.

Display 18.11 *Drag Pointer*



Release the pointer, and the nodes are connected.

Display 18.12 *Connect Nodes Window*



You can connect nodes by using the Connect Nodes window. Right-click the node that you want to connect to other nodes and select **Connect Nodes** from the pop-up menu. Note that the arrows in your process flow diagram point away from the node that you use to open the Connect Nodes window

Click the node name in the window to select the node. You can select multiple nodes by CTRL-clicking or SHIFT-clicking a range of nodes in the window.

You can also use the up and down arrows on your keyboard to select nodes that you want to connect.

Click **OK** to connect the nodes.

Disconnecting Nodes

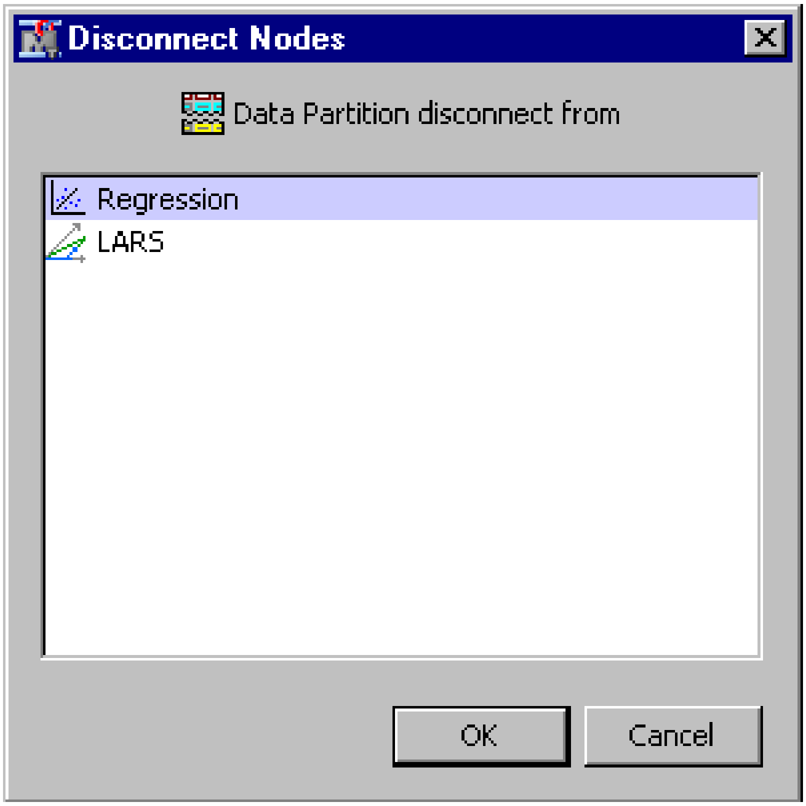
You can disconnect nodes in the Diagram Workspace by clicking the arrow between two nodes so that it is highlighted, and then pressing **Delete** on the keyboard. You can also disconnect two nodes by right-clicking an arrow that connects two nodes in the Diagram Workspace, and selecting Delete from the pop-up menu.

You can disconnect multiple nodes in the Diagram Workspace by holding CTRL down, selecting the arrows that you want to delete, and then either pressing **Delete** on the keyboard, or right-clicking a selected arrow, and choosing **Delete**.

You can also disconnect nodes by using the Disconnect Nodes window. To open the Disconnect Nodes window, right-click a node and select **Disconnect Nodes** from the pop-up menu.

You can select single or multiple nodes by using the Disconnect Nodes window.

Display 18.13 *Disconnect Nodes Window*



Click the node name in the window to select the node. You can select multiple nodes by holding the CTRL key down and then clicking on the nodes that you want to select. Also, you can select one node, hold the SHIFT key down, the click any other node to select all nodes between and including the two you clicked.

You can also use the up and down arrows on your keyboard to choose a node that you want to select. To select multiple nodes with this method, highlight the first node that you want to select, hold down the CTRL key, use the up and down arrows to move to another node, and then select that node by pressing the space bar. If you hold the SHIFT key while using the up and down arrows that you can select all nodes between your first and last node.

Click **OK** to disconnect the nodes.

Deleting Single Nodes

There are two ways in which you can delete a node from a process flow diagram. Here is the first:

1. Right-click the node icon to open a pop-up menu.
2. Select **Delete**. The Confirm Delete dialog box appears and asks you to verify your choice. Click **Yes** to remove the node from the process flow diagram. Click **No** to keep the node.

Here is the second:

1. Click the node to highlight it.

2. Select **Edit** ⇨ **Delete** to delete the node. You can also press the Delete key to delete the selected node.

CAUTION:
You cannot undelete a deleted node.

Deleting Multiple Nodes

To delete multiple nodes, follow these steps:

1. Select a node icon that you want to delete, and then hold down the Control key and click to select the remaining node icons that you want to delete. The selected nodes become highlighted. You can also select multiple nodes by dragging your pointer around the nodes that you want to delete.
2. Right-click a selected node.
3. Select **Delete**. A Confirm Delete window appears that asks you to verify your choice. If you click **Yes**, then the nodes are removed from the process flow diagram. If you click **No**, then the nodes remain in the process flow diagram.

To delete all of the nodes in the Diagram Workspace, select **Edit** ⇨ **Select All** from the main menu, and then choose **Edit** ⇨ **Delete**.

Moving or Repositioning Nodes

To move a node, follow these steps:

1. Click the node to select it.
2. Drag the node icon to a new position in the Diagram Workspace. The connections stretch to accommodate the move.

   Note: You can move multiple nodes by selecting the nodes that you want to move and dragging them to a new position.

You can arrange your diagram nodes by selecting **Edit** ⇨ **Layout** from the main menu, and then choosing **Horizontally** or **Vertically**.

Copying and Pasting Nodes

You can copy an existing node to the clipboard and paste it to the Diagram Workspace. The same node properties of the node that you copy are used for the new node. You can copy a node and paste it to the same diagram or to a different diagram.

To copy an existing node:

1. Right-click the node that you want to copy in the Diagram Workspace and select **Copy**.
2. Right-click the location in the diagram where you want to paste the node and select **Paste**.

Alternatively, after you select a node in the Diagram Workspace, you can select **Copy** and then **Paste** from the main menu under **Edit**. To determine where the copied nodes will be pasted, left-click in the diagram where you want to paste the nodes.

Using the Results Window

Results Window Overview

You can view the results from any node in SAS Enterprise Miner by using the Results window. Use the Results window to do the following tasks:

- View the log, output, and training code of your node.
- View the flow and publish score code.
- Examine assessment statistics, fit statistics, and residual statistics.
- View graphical output.

Results Window Buttons

You can perform actions in the Results window by using the following tool icons:

- Log — opens the Log window.
- Output — opens the Output window.
- Print — opens the Print window.
- Table — opens a table of the data that is associated with the graph that is selected.
- Plot — opens the Graph Wizard, using the data in the table that you open.

Results Window Main Menus

The Results window contains menus that enable you to perform various administrative tasks.

Note: Not all of the following menus are available for each node, and menus that are available in some nodes are not listed here.

The menu items are as follows:

File

- **Save As** — opens the Save As dialog box.
- **Print** — prints the contents of the active window.
- **Close** — closes the Results window.

Edit — The submenus that are displayed depend on the window that is active within the Results window

- **Edit (Text Output)**
  - **Undo** — cancels the last action that you performed.
  - **Redo** — repeats the last action that you performed.
  - **Cut** — deletes the text that you select and copies the text to the clipboard.
  - **Copy** — copies text that you select.
  - **Paste** — copies the contents of the clipboard to the active window.

  - **Select All** — marks all of the text in the active window.
  - **Clear All** — removes the output from the window.
  - **Find** — opens the Find/Replace dialog box in which you enter text that you want to find.
  - **Go To Line** — opens the Go To Line dialog box. Enter the line number on which you want to enter text.
- **Edit (Graph Output)**
  - **Graph Properties** — opens the Properties page for the selected graph.
  - **Action Mode** — enables you to choose the graph interaction mode.
    - **Select** — choose additional elements by pressing CTRL and clicking the desired elements.
    - **Brush** — use of a selection rectangle that persists on screen after mouse buttons are released enables you to move or reshape multiple or different graph elements.
    - **Viewport** — Enables you to select a region of the diagram to enlarge
    - **Reset** — resets a perspective change to a diagram.
  - **Data Options** — opens the Data Options Dialog box.
  - **Copy** — copies the graph image to the clipboard.
  - **Focus on Chart** — enables the user to expand one of the charts so that it occupies the entire lattice space and can be examined in more detail.
  - **Reset Focus** — restores a lattice chart view from a magnified Focus on Chart perspective to the original scale of the lattice chart.
  - **Clone** — opens a copy of the selected graph.
  - **Get Saved Chart** — opens the saved version of the graph

View — opens separate windows within the Results window that display the following:

- **Properties**
  - **Settings** — displays a window with a read-only table of the node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — opens the Notes window, which displays user-generated notes.
- **SAS Results**
  - **Log** — the log for the selected tool. The Log window displays the log only for the associated tool, and does not display the log from the previous tools.
  - **Output** — the SAS output for the selected tool.

  - **Flow Code** — the SAS code used to produce the output that the node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Predictive Model Markup Language (PMML) code that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 . Not all SAS Enterprise Miner nodes can produce PMML code.
- **Assessment**
  - **Fit Statistics** — the “Fit Statistics Table ” on page 252 for the selected modeling tool.
  - **Statistics Comparison** — the statistics comparison table for the selected modeling tool.
  - **Classification Chart** — the classification chart and graphs for the selected modeling tool.
  - **Score Rankings Overlay** — opens or highlights the Score Rankings Overlay window that can display the following assessment statistics from a drop-down menu:
    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response
  - **Score Rankings Matrix** — opens the Score Rankings Matrix chart, which compares assessment statistics options by data role.
  - **Score Distribution** — opens the Score Distribution chart, which compares events by data role.
  - **ROC Chart** — opens the ROC chart, which compares models as a function of specificity and sensitivity. For more information, see “Score Rankings Chart and Score Rankings Table” on page 254.
- **Model**
  - **Effects Plot** — opens the Effects Plot window, which plots positive and negative effects against a response variable.
  - **Estimate Selection Plot** — opens the Estimate Selection Plot window.
- **Table** — opens a table of the data that is associated with the graph that you have open.
- **Plot** — opens the graph wizard, using the data in the table that you have opened.

Window

- **Tile** — arranges all of the opened windows in the Results window so that all windows are visible at the same time.
- **Cascade** — displays all of the opened windows in the Results window so that windows overlap.

Results Window Contents

The Enterprise Miner Results window contains one or more elements in sub-windows. The Output window is always displayed. It contains the SAS output that is generated by running the SAS Enterprise Miner node. Depending on the node type, additional tables or plots related to the node's functionality are displayed in the Results window.

Results Window Output Window

The Output window displays the SAS output that is generated when a node is run.

The Variable Summary section in the Output window is displayed for all nodes. It displays the frequency counts of variables for each combination of variable role and measurement level.

The sections in the Output window that are common to modeling nodes include Model Event, Decision Matrix, Predicted and Decision Variables, Fit Statistics, Classification Chart, Decision Table, Event Classification Chart, Assessment Score Rankings, and Assessment Score Distribution.

Other sections in the Output window include the following:

- **Model Event** — displays the target variable, event level, measurement level, number of levels, order, and target label.
- **Decision Matrix** — displays the decision matrix.
- **Predicted and Decision Variables** — displays the predicted and decision variables and their labels.
- **Fit Statistics** — displays the statistics of a model. Different modeling nodes display different types of fit statistics. The Output window transposes the columns of the fit statistics that are displayed in the Results Fit Statistics window. For more information, see "Fit Statistics Table " on page 252.
- **Classification Chart** — opens a classification bar chart when you model a non-interval target variable. For more information, see "Classification Plot and Table" on page 251.
- **Decision Table** — displays the decision, percent of target, percent of decision, percent of total observations, and frequency counts of observations for each target and decision combination.
- **Event Classification Table** — displays the frequency counts of false positive, false negative, true positive, and true negative predictions of observations.
- **Assessment Score Rankings** — displays assessment statistics such as gain, lift, and % response that are used to create the score rankings plot.
- **Assessment Score Distribution** — displays the number of events and nonevents and percentage of events over bins of posterior probabilities. These statistics are used to create the score distribution plot. For more information, see "Using the Diagram Editor" on page 240.

Results Window Plots and Tables

Overview of the Results Window Plots and Tables

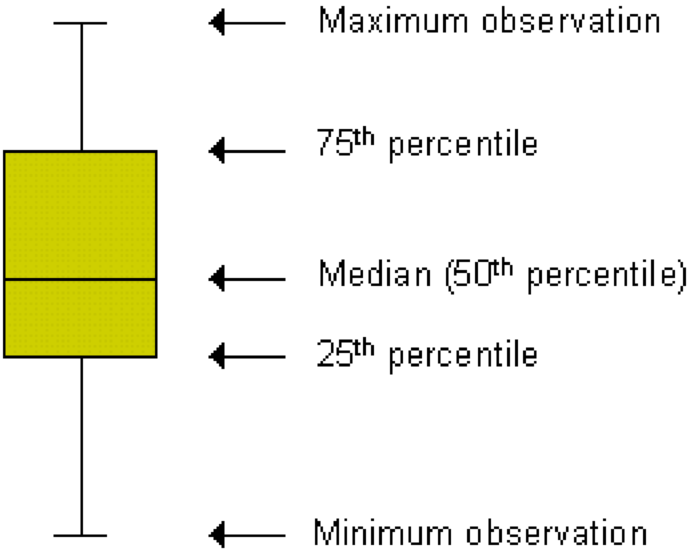
The plots and tables that appear in the Enterprise Miner Results window vary according to the node that produced the results. The following plots and tables are a representative sample of some Enterprise Miner Results window plots and data tables.

Residual Statistics Plot and Table

When you model an interval target, the Results window produces a Residual Statistics plot.

To view the Residual Statistics plot, select **View** ⇨ **Assessment** ⇨ **Residual Statistics**. The Residual Statistics plot displays a box plot for the residual measurements of the interval target. This plot enables you to examine the distribution of the residuals. The following display illustrates the elements of the plot:

Display 18.14 *Elements*



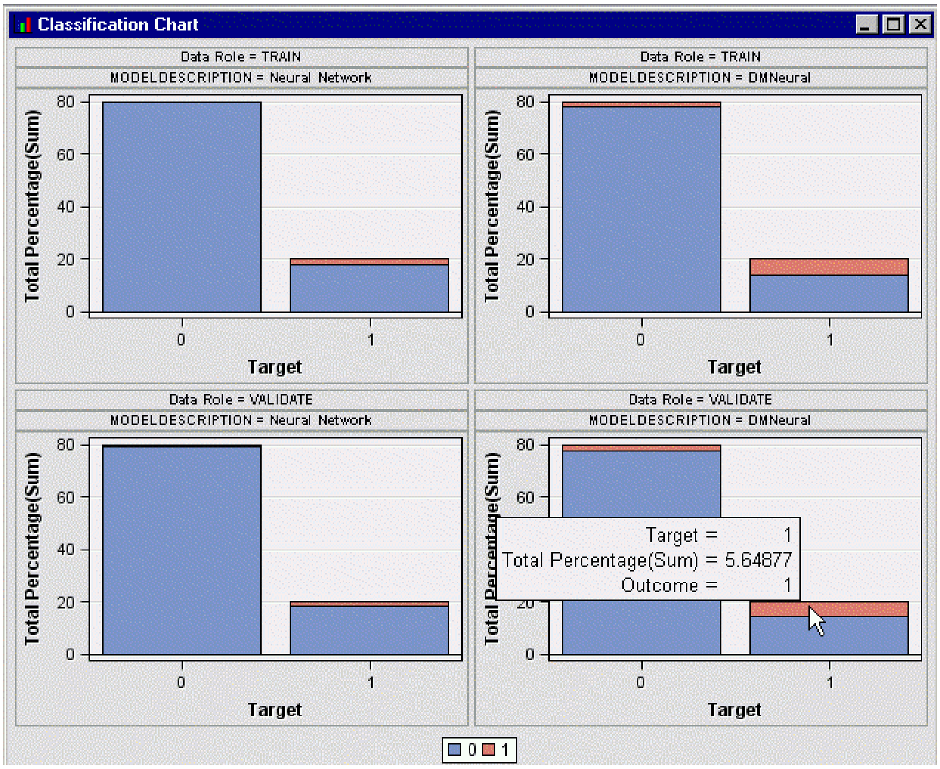
To view the data table for a Residual Statistics plot in the Results window, click the plot to select it. Then from the Results window main menu, select **View** ⇨ **Table**. You can right-click any column cell in the Residual Statistics plot table and select **Show** ⇨ **Variable Names** to change the column headings from the default descriptive variable labels to the variable names that are used in SAS code. To change back to variable labels from variable names, right-click any column cell and select **Show Variable Labels**.

Classification Plot and Table

When you model a non-interval target, the modeling node generates a classification bar chart.

The stacked bars of the Classification Chart show the relationships between the actual and predicted values of the target variable. Charts are produced for the training, validation, and test data sets of each modeling node that is compared. The following display shows an example of the Classification Chart for a Model Comparison run that compares Neural Network node and DMNeural node models:

Display 18.15 *Classification Chart Comparing Neural Network Node and DMNeural Node*



The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represents the percentage of total observations. Move the cursor over plot bars to display additional statistics.

- **Target** — the value of the target variable.
- **Total Percentage(Sum)** — the proportion of all observations that predicted the specified value for the level of the target.
- **Outcome** — the value of the target variable that was predicted.

While the Classification Chart is active, you can go to the Results window menu and select ⇨ **View** ⇨ **Table** to see the table of data that SAS Enterprise Miner uses to generate the plot.

Fit Statistics Table

The Fit Statistics table displays a wide variety of fit statistics for models. The list of fit statistics below is output by the Model Comparison node. Other modeling nodes output a subset of the fit statistics below. If you are interested in viewing one of the fit statistics below and it is not output by your modeling node, you can add a Model Comparison node behind your modeling node and rerun your process flow diagram. Then you can view the complete set of fit statistics.

You can right-click any column cell in the Fit Statistics table and select **Show Variable Names** to change the column headings from the default descriptive variable labels to the variable names that are used in SAS code. To change back to variable labels from variable names, right-click any column cell and select **Show Variable Labels**.

The Fit Statistics table contains the following columns for train, validate, and test data sets.

- **Selected Model** — the status of the model.
- **Predecessor Node** — the label name of the predecessor modeling node.
- **Model Node** — the default name or output model type of the predecessor modeling node.
- **Model Description** — the name of the model node used.
- **Target Variable** — the name of the target variable.
- **Total Degrees of Freedom** — total degrees of freedom for the training data set.
- **Degrees of Freedom for Error**— degrees of freedom for error in the training data set.
- **Model Degrees of Freedom** — total degrees of freedom for the training data set.
- **Number of Estimated Weights** — the number of estimated weights in the data set.
- **Akaike's Information Criterion** — Akaike's Information Criterion from the training data set.
- **Schwarz's Bayesian Criterion** — Schwarz's Bayesian Criterion from the training data set.
- **Average Squared Error** — average squared error from the training data set.
- **Maximum Absolute Error** — maximum absolute error from the training data set.
- **Divisor of ASE** — divisor of the Average Squared Error for the training data set. It equals the number of training observations.
- **Sum of Frequencies** — the weighted number of training observations.
- **Root Average Squared Error** — square root of average squared error from the training data set.
- **Sum of Squared Errors** — sum of squared errors from the training data set.
- **Sum of Case Weights Times Freq** — sum of case weights times frequency from the training data set.
- **Final Prediction Error** — final prediction error from the training data set.
- **Mean Squared Error** — mean squared error from the training data set.
- **Root Final Prediction Error** — square root of the final prediction error from the training data set.
- **Root Mean Square Error** — square root of the mean squared error from the training data set.
- **Average Error Function** — the averaged error function value from the training data set.
- **Error Function** — the error function value from the training data set.
- **Misclassification Rate** — misclassification rate from the training data set.
- **Number of Wrong Classifications** — the number of wrong classifications from the training data set.
- **Frequency of Classified Cases** — the number of classified cases in the data.
- **Gini Coefficient** — two times the area between the training data Lorenz curve and the baseline.

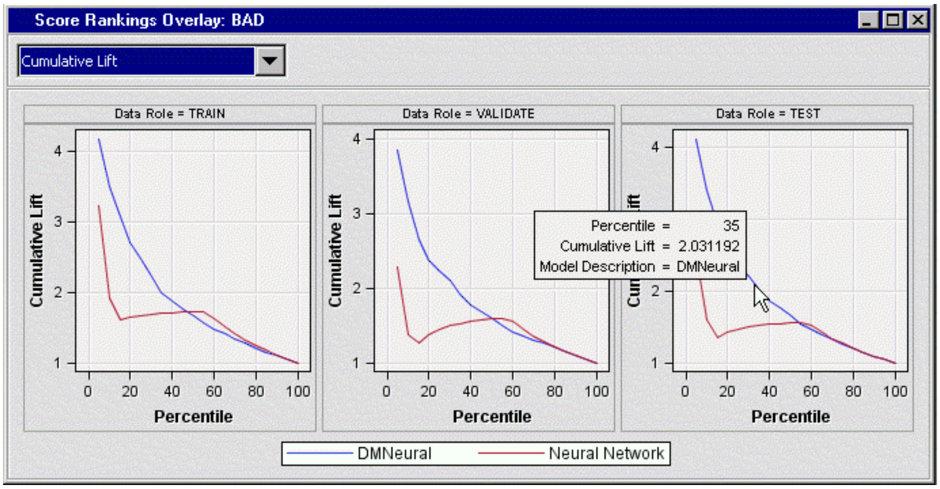
- **Kolmogorov-Smirnov Probability Cutoff** — the data Kolmogorov-Smirnov probability cutoff for a binary target, which measures the maximum vertical separation between the cumulative distributions of event and non-event scores.
- **ROC Index** — the area under the test data Receiver Operating Characteristic curve.
- **Gain** — ((% of events in decile / random % of events in decile)-1).
- **Lift** — the ratio of percent captured response within each decile to the baseline percent response.
- **Percent Response** — the proportion of true responders in each decile.
- **Cumulative Lift** — the cumulative ratio of percent captured responses within each decile to the baseline percent response.
- **Cumulative Percent Response** — the cumulative proportion of responders.
- **Percent Captured Response** — the proportion of total responders captured in a decile or demi-decile.
- **Cumulative Percent Captured Response** — the cumulative proportion of total responders captured in a decile or demi-decile.
- **Bin-Based Two-Way Kolmogorov-Smirnov Statistic** — the Kolmogorov-Smirnov statistic for a binned target, which measures the maximum vertical separation between binned distributions of event and non-event scores.
- **Bin-Based Two-Way Kolmogorov-Smirnov Probability Cutoff** — the Kolmogorov-Smirnov probability cutoff for a binned target, which measures the maximum vertical separation between binned distributions of event and non-event scores.
- **Selection Criterion** — the selection criterion that is specified.

Score Rankings Chart and Score Rankings Table

Score Rankings charts enable you to plot the following statistics on the vertical axis. These statistics are computed based on the model that you create, the random baseline model, and the exact model. The random baseline model assumes that you target the events at random. When the exact model for a nonbinary target is computed, observations are sorted in descending order by actual profit. The exact model captures all of the events in the first few deciles. For a binary target, the exact model is computed from the model results; the data set does not have to be sorted.

- Cumulative Lift
- Lift
- Gain
- % Response
- Cumulative % Response
- % Captured Response
- Cumulative % Captured Response

Display 18.16 *Score Rankings Chart*



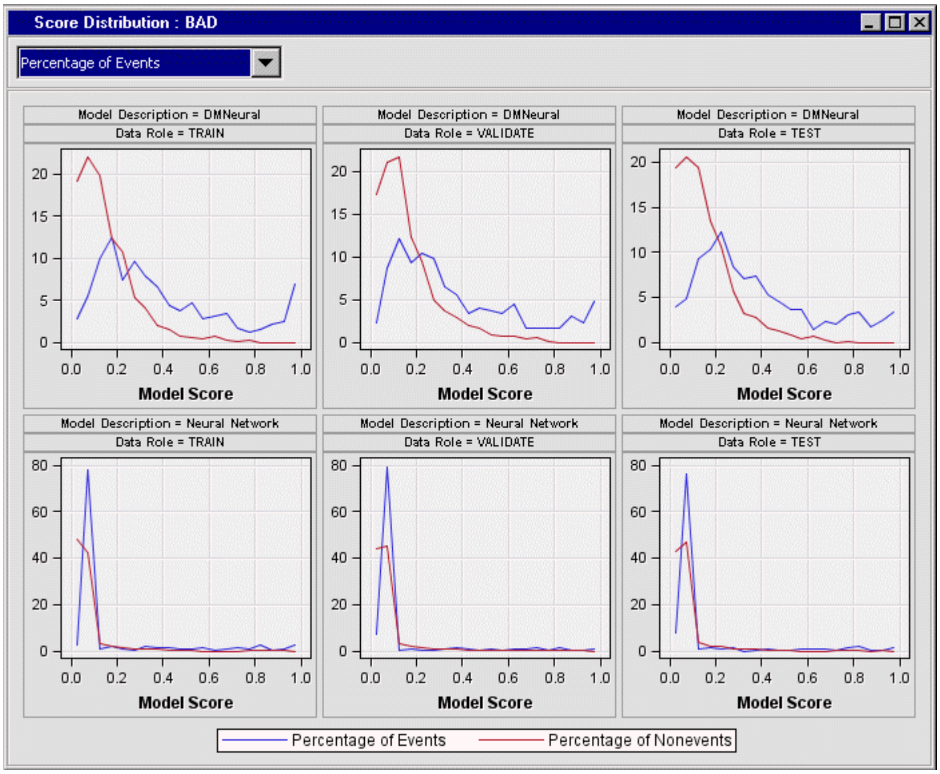
The horizontal axis of a Score Rankings chart displays the groups of observations. The grouping of observations depends on the value of the Number of Bins property. The default value of the Number of Bins property is 20. That is, the observations are grouped into 20 bins. When you move the cursor over the curve, a text plot displays the percentile value, cumulative lift value (if cumulative lift is selected in the drop-down menu), and the modeling tool that creates the model.

To view the Score Rankings table for a Score Rankings chart, click on the chart to select it, and then from the Results window main menu, select **View** ⇨ **Tools**. For more information, see "Score Rankings Chart and Score Rankings Table" on page 254.

Score Distribution Chart and Score Distribution Table

The Score Distribution chart displays the percentage of events and non-events in a bin across the range of model scores for a binary target. Observations in the scored data set are grouped into bins. A model that fits the data well assigns high scores to the events and low scores to the non-events.

Display 18.17 *Score Distribution*



To view the Score Distributions table for a Score Distribution chart, click on the chart to select it, and then from the Results window main menu, select **View** ⇨ **Table**.

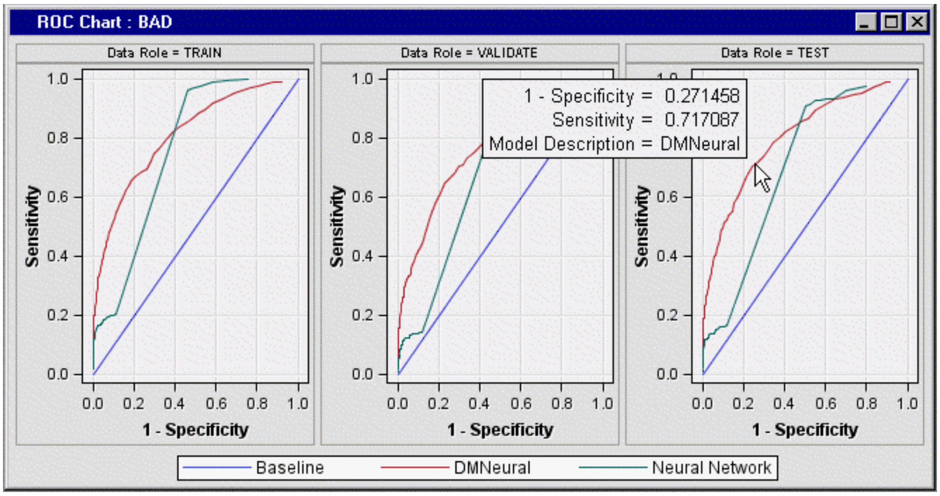
ROC Chart and ROC Chart Table

ROC charts are useful for comparing the global performance of a model. ROC charts require a binary target.

The receiver operating characteristic (ROC) chart displays the sensitivity and 1-(specificity measures) for a model over a range of cutoff values. Sensitivity is a measure of accuracy for predicting events. It is equal to: true positives / (true positives + false positives) Specificity is a measure of accuracy for predicting nonevents. It is equal to: true negatives / (true negatives + false negatives). One minus specificity is simply the number of false positives (the number of nonevent observations that the model incorrectly predicts as events for a given probability cutoff point) divided by the number of nonevents.

Each point on the curve represents a cutoff probability. The cutoff choice represents a trade-off between sensitivity and specificity. Ideally, you would like to have high values for both sensitivity and specificity, so that your model can accurately predict both events and nonevents. A lower cutoff typically gives more false positives. A high cutoff gives more false negatives, a low sensitivity, and a high specificity. For more information, see “ROC Chart and ROC Chart Table” on page 256.\

Display 18.18 *Receiver Operating Characteristic Chart*



1 - Specificity = 0.271458
Sensitivity = 0.717087
Model Description = DMNeural

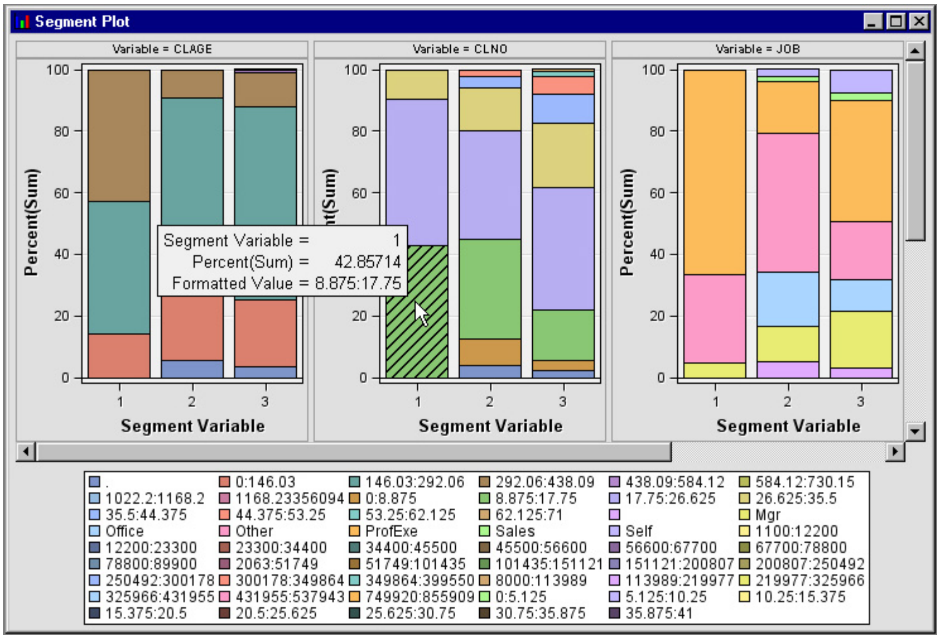
You can use the Data Options dialog box to change the plotting variable on the vertical axis.

To view the data table for a ROC Chart, click on the chart to select it. Then from the Results window main menu, select **View** ⇨ **Table**.

Segment Size Plot and Table

The cluster segment plot shows the varying sizes of the clusters generated in the output data:

Display 18.19 *Cluster Segment Plot*



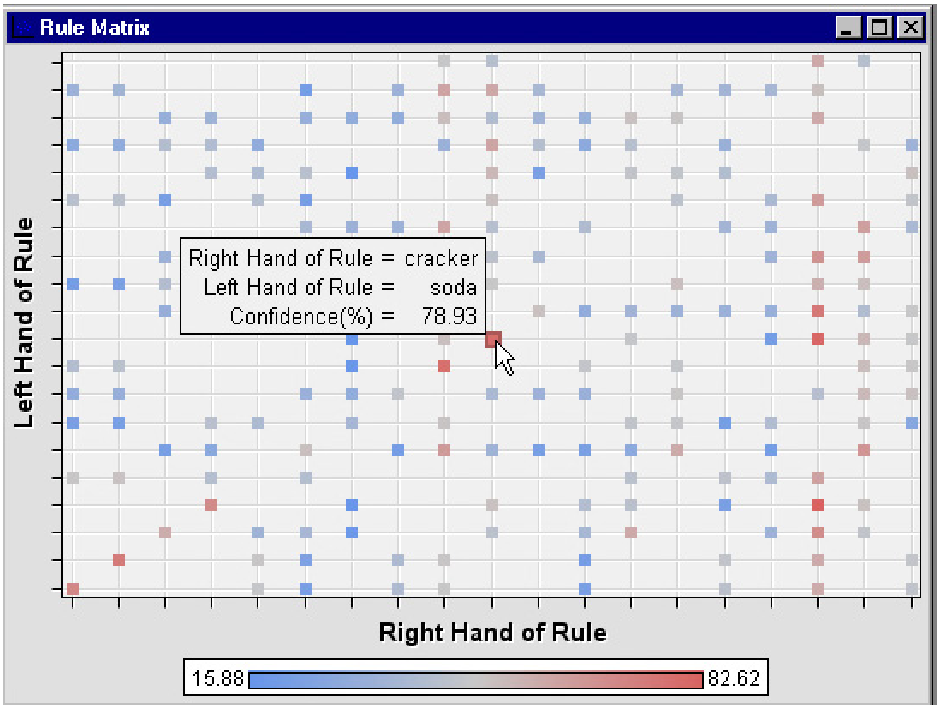
Segment Variable = 1
Percent(Sum) = 42.85714
Formatted Value = 8.875:17.75

To view the data table for Segment Plot, click on the chart to select it, and then from the Results window main menu, select **View** ⇨ **Table**.

Rule Matrix Plot

The association analysis rule matrix displays a grid plot of the items in the Right Hand of Rule on the X-axis and the Left Hand of Rule on the Y-axis.

Display 18.20 *Rule Matrix Plot*



Right Hand of Rule = cracker
Left Hand of Rule = soda
Confidence(%) = 78.93

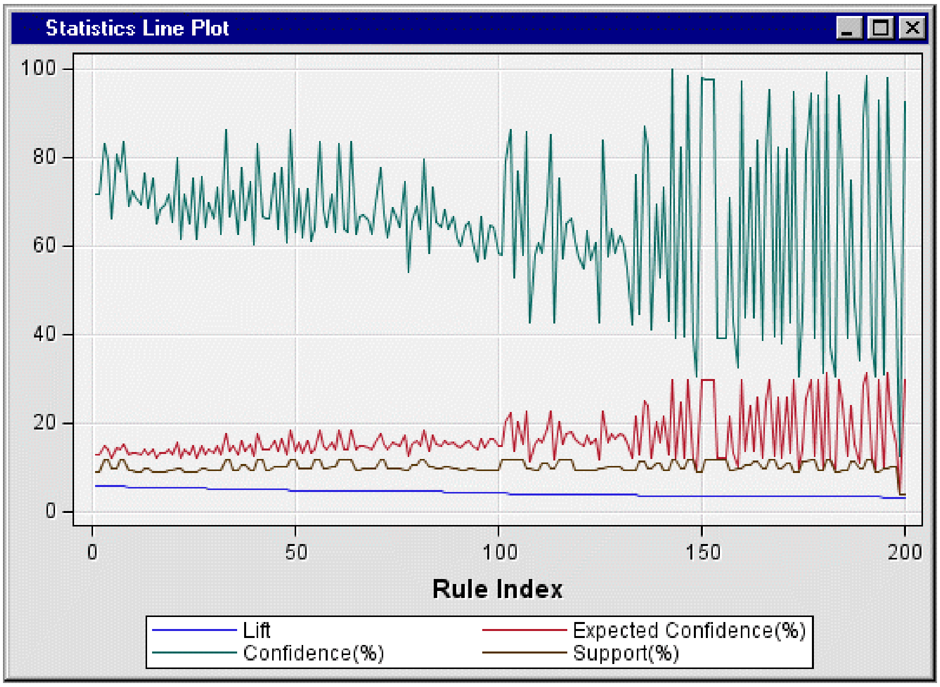
A data point in the plot corresponds to a transaction rule. The ToolTip window displays the rule and the Confidence (%) value of the rule.

To view the data table for a Rule Matrix Plot, click on the plot to select it. Then from the Results window main menu, select **View** ⇨ **Table**.

Statistics Line Plot

The association analysis statistics line plot displays a line plot of various values for each rule on the Y-axis. Here is an example of the statistics line plot:

Display 18.21 *Statistics Line Plot*



The X-axis represents the values of Rule Index.

- lift — not available in a sequence analysis
- expected confidence (%) — not available in a sequence analysis
- confidence (%)
- support (%)

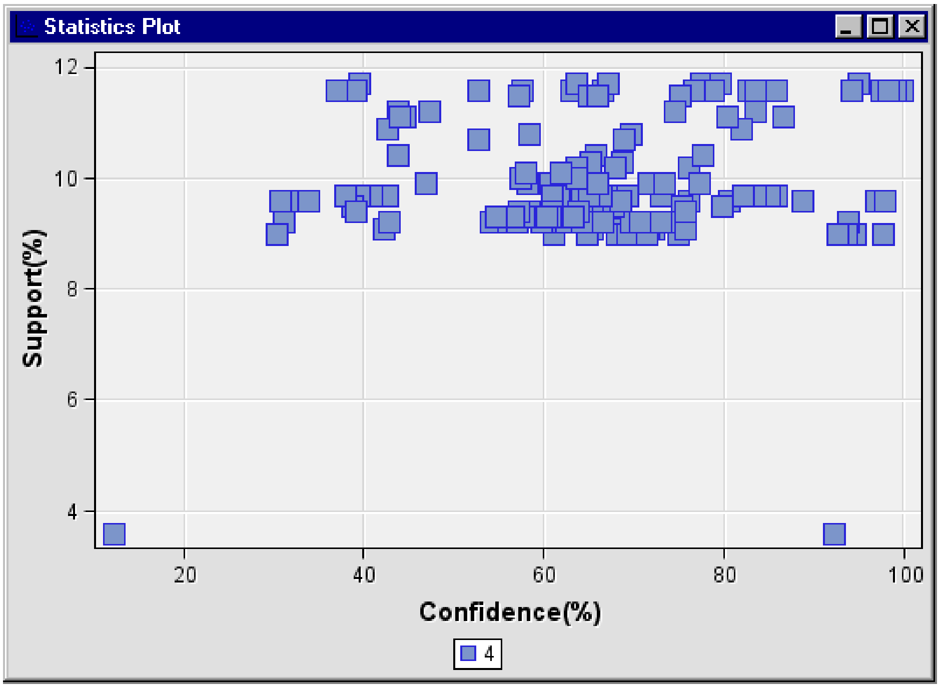
The Statistics Line Plot and Rule Matrix windows are linked. When you select a data point in the rule matrix, the associated data points in the statistics line plot are highlighted.

To view the data table for a Statistics Line Plot, click on the plot to select it. Then from the Results window main menu, select **View** ⇨ **Table**.

Statistics Plot

The statistics plot displays a plot of the frequency counts for given ranges of support and confidence levels in an association analysis. The following is an example of a statistics plot:

Display 18.22 *Statistics Plot*



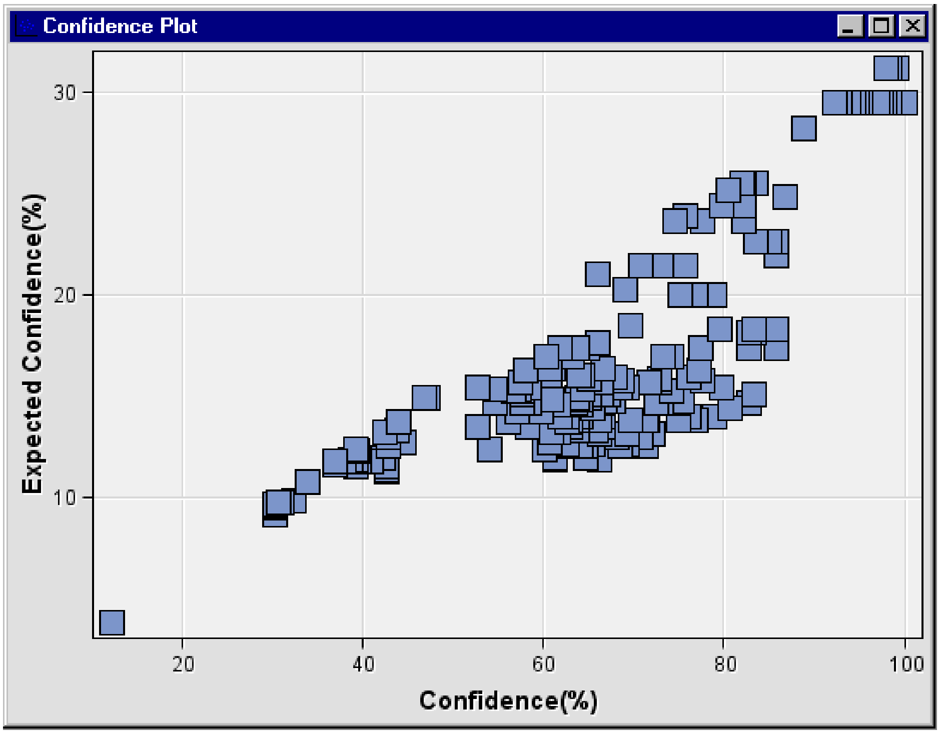
The X-axis and Y-axis represent the confidence (%) and the support (%), respectively.

To view the data table for a Statistics Plot, click on the plot to select it. Then from the Results window main menu, select **View** ⇨ **Table**.

Confidence Plot

The confidence plot displays a scatter plot of the Confidence (%) on the X-axis versus the Expected Confidence (%) on the Y-axis. If the confidence values are close to the expected confidence, the data points will be close to a 45-degree line that starts from the origin. Nodes such as Association Analysis and Path Analysis use confidence plots in their Results windows. The following is an example of a confidence plot:

Display 18.23 *Confidence Plot*



The data underlying the Confidence plot can be viewed in the “Rules Table” on page 426 . To view the Rules Table for a Confidence plot, click on the plot to select it. Then from the Results window main menu, select **View** ⇨ **Table**.

Modifying Results Graphs

SAS Enterprise Miner enables you to modify and enhance Results graphs. You can graphically explore the effects of different variable roles, add or change response variables, or modify graph attributes to emphasize particular results or enhance appearance for presentation purposes.

- Data Options Dialog Box on page 261
- Graph Properties Window on page 251

Data Options Dialog Box

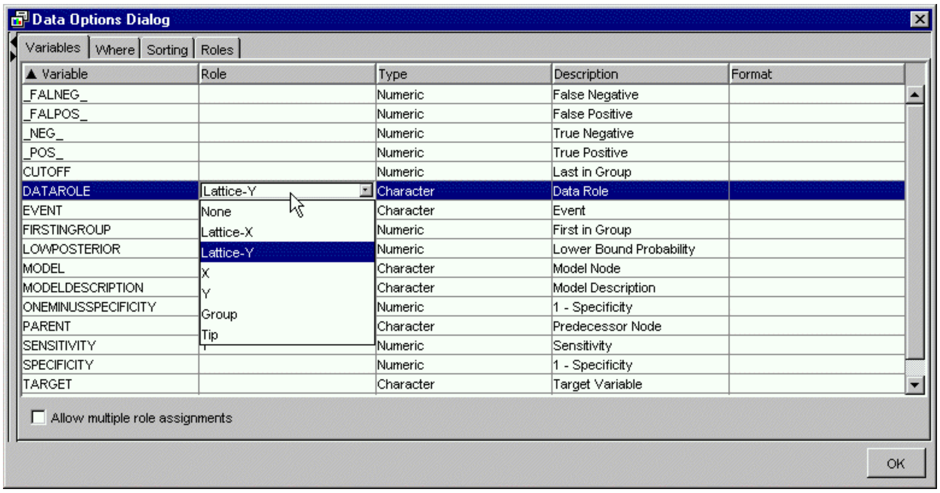
The Data Options Dialog box enables you to modify or create your own graphs using existing plots. For example, you can use the Data Options Dialog box to change the response variables used in a results plot.

To open the Data Options Dialog box, right-click inside a graph that you want to modify and select **Data Options** from the pop-up menu.

The Data Options Dialog box contains four tabs for configuring data plots.

- The **Variables** tab of the Data Options Dialog box contains a table that you can use to configure response variables to use in a graph or plot. The Variable column lists a variety of SAS variables that you can choose as your response variables. Columns for Type, Description, and Format provide additional information about the available SAS variable.

Display 18.24 *Variables Tab*



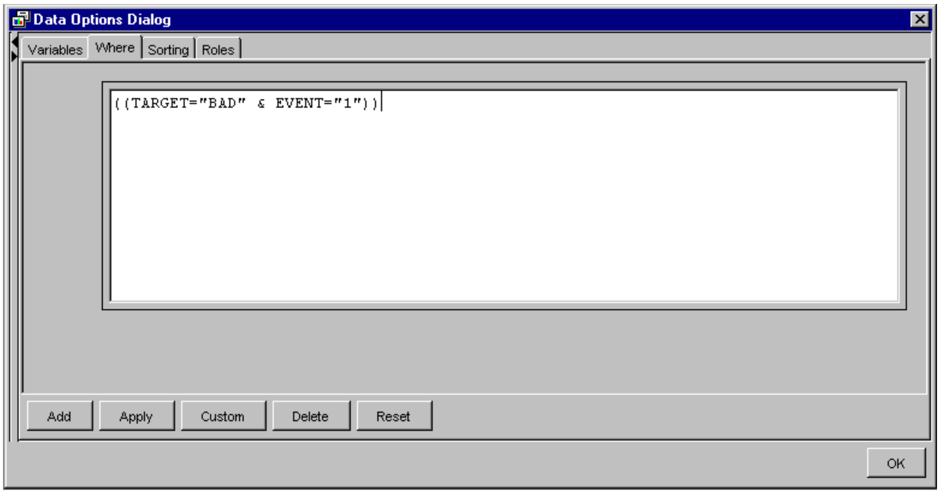
▲ Variable	Role	Type	Description	Format
FALNEG		Numeric	False Negative	
FALPOS		Numeric	False Positive	
NEG		Numeric	True Negative	
POS		Numeric	True Positive	
CUTOFF		Numeric	Last in Group	
DATAROLE	Lattice-Y	Character	Data Role	
EVENT		Character	Event	
FIRSTINGROUP		Numeric	First in Group	
LOWPOSTERIOR		Numeric	Lower Bound Probability	
MODEL		Character	Model Node	
MODELDESCRIPTION		Character	Model Description	
ONEMINUSSPECIFICITY		Numeric	1 - Specificity	
PARENT		Character	Predecessor Node	
SENSITIVITY		Numeric	Sensitivity	
SPECIFICITY		Numeric	1 - Specificity	
TARGET		Character	Target Variable	

□ Allow multiple role assignments

Click a cell in the Role column to open a selection list that contains graphical role options. You can find it useful to change the Y response variable to different fit statistics. If the Allow multiple role assignments check box is selected, more than one Y response variable can be selected. You can click on column headings to toggle between ascending and descending sorts. In the example above, the Variable column is sorted in ascending order. Click **OK** to see your changes updated in the graph plot.

- The **Where** tab of the Where Options Dialog box contains an expression builder that you can use to build a WHERE-clause or a Boolean expression to subset the data that you want to plot.

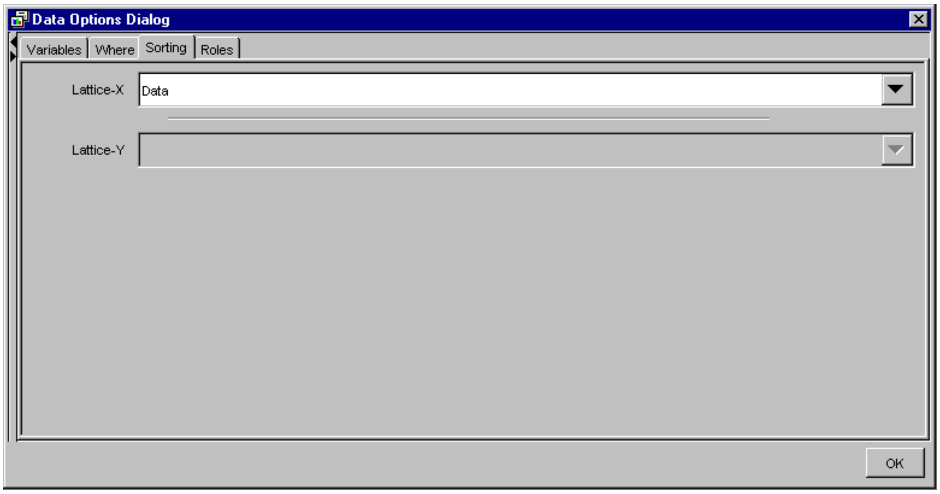
Display 18.25 *Where Tab*



```
((TARGET="BAD" & EVENT="1"))
```

- The **Sorting** tab of the Data Options Dialog box can be used to sort the X-axis data points.

Display 18.26 *Sorting Tab*



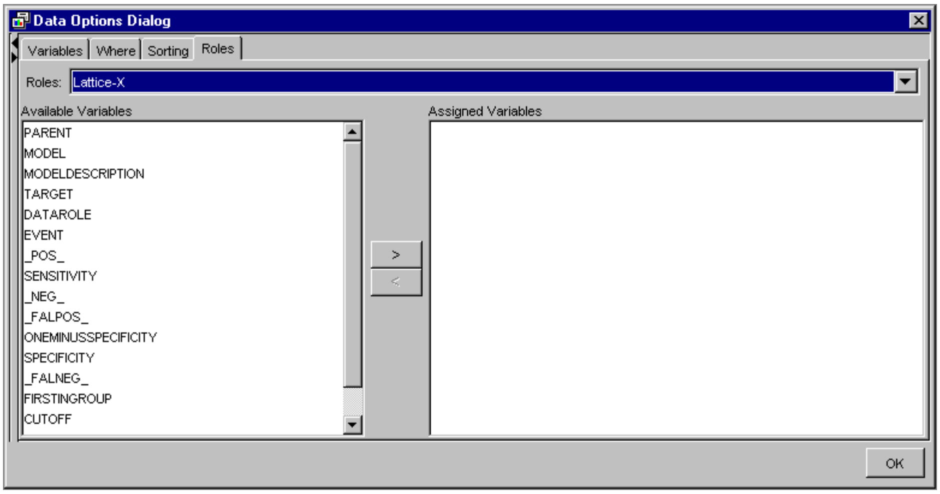
Lattice-X: Data

Lattice-Y:

The drop-down list choices are Ascending, Descending, and Data. You can use the ascending and descending ordering controls, for example, to look at descending frequency counts without having to sort the data table on the server. The data ordering applies to ordering bars in bar charts where nominal or ordinal groupings are used.

- The **Roles** tab of the Data Options Dialog box enables you to assign roles to a list of available variables.

Display 18.27 *Roles Tab*



Roles: Lattice-X

Available Variables

PARENT
MODEL
MODELDESCRIPTION
TARGET
DATAROLE
EVENT
POS
SENSITIVITY
NEG
FALPOS
ONEMINUSSPECIFICITY
SPECIFICITY
FALNEG
FIRSTINGROUP
CUTOFF

Assigned Variables

Graph Properties Window

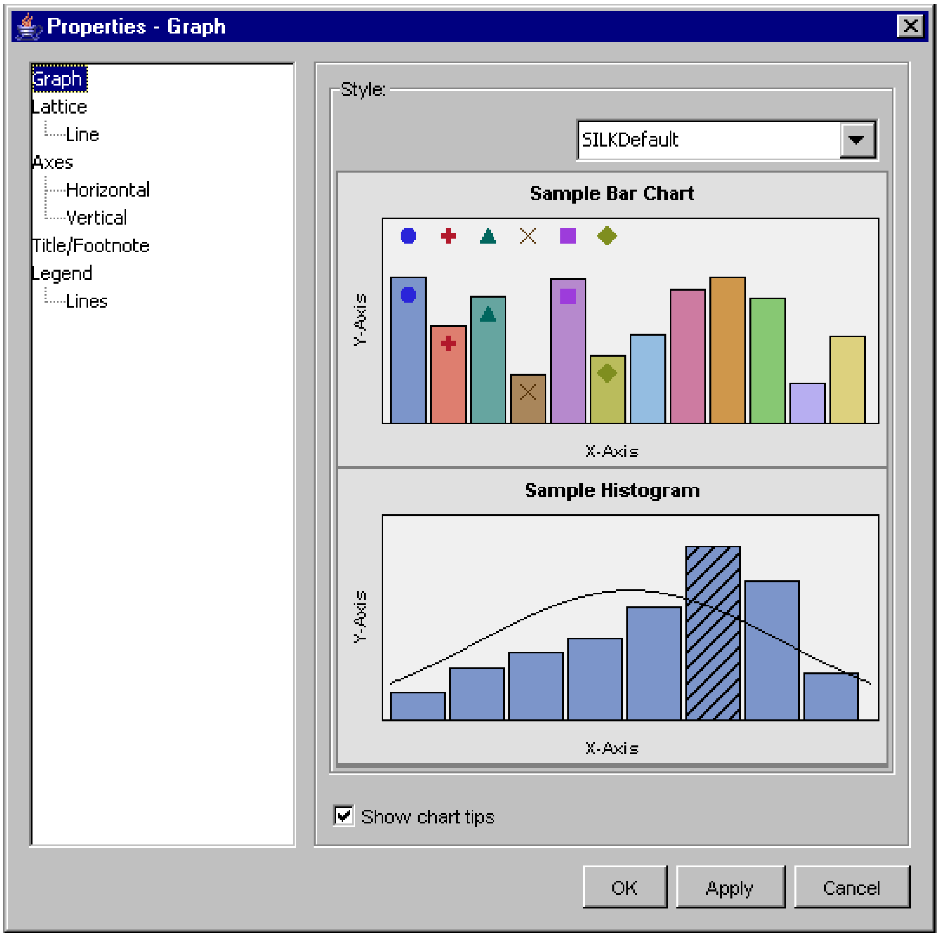
The Graph Properties window enables you to modify attributes of an existing Results graph to enhance certain features that you want to call attention to, or to prepare a results graph for use in a presentation that uses external software.

The Graph Properties window offers configuration controls that vary according to the type of chart that is selected. The window name reflects the type of graph that you are modifying and contains several tabs that group the plot entities that you can modify.

For example, to change the display properties of a ROC Chart, right-click inside the chart and select **Graph Properties**. A Graph Properties window with four sections appears. The following example Graph Properties windows appears for SAS Enterprise Miner ROC charts.

- **Graph**

Display 18.28 *Graph*



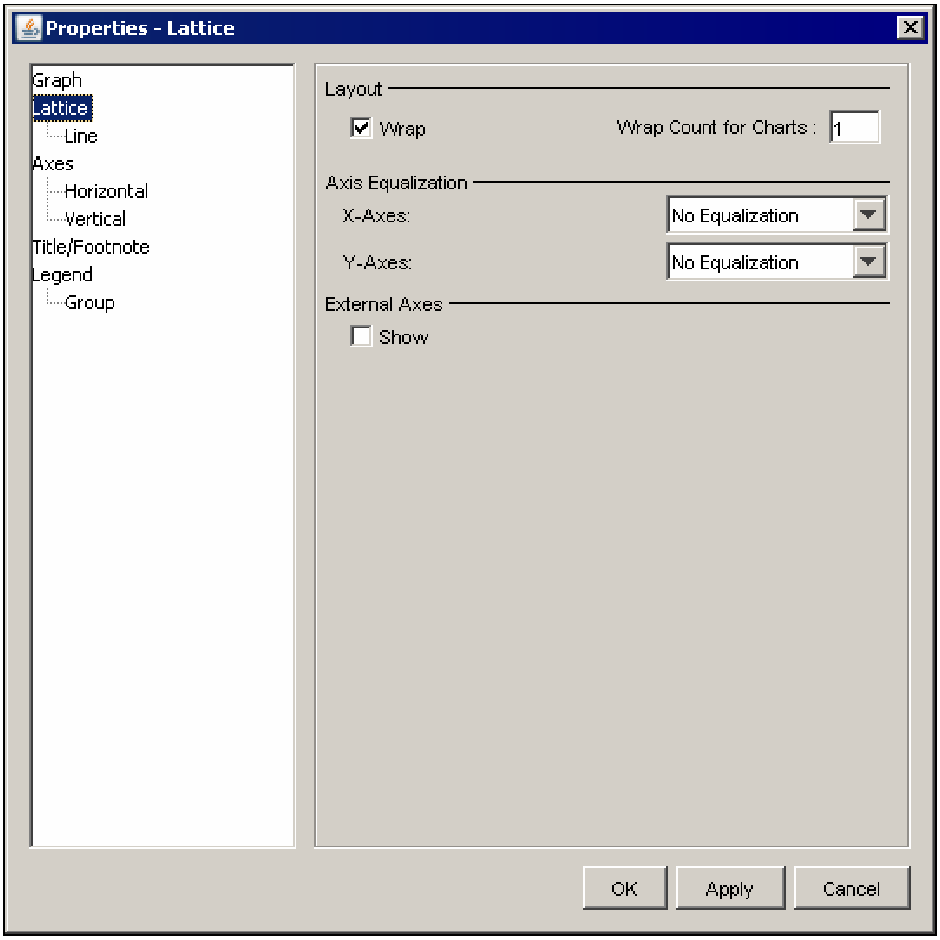
Style: SILKDefault

☑ Show chart tips

The **Graph** section enables you to specify a default style. You can pick from Analysis, Default, Journal, Listing, Statistical, or SILKDefault. You can also select whether to show chart tips.

- **Lattice**

Display 18.29 *Lattice*



Layout

☑ Wrap

Wrap Count for Charts : 1

Axis Equalization

X-Axes: No Equalization

Y-Axes: No Equalization

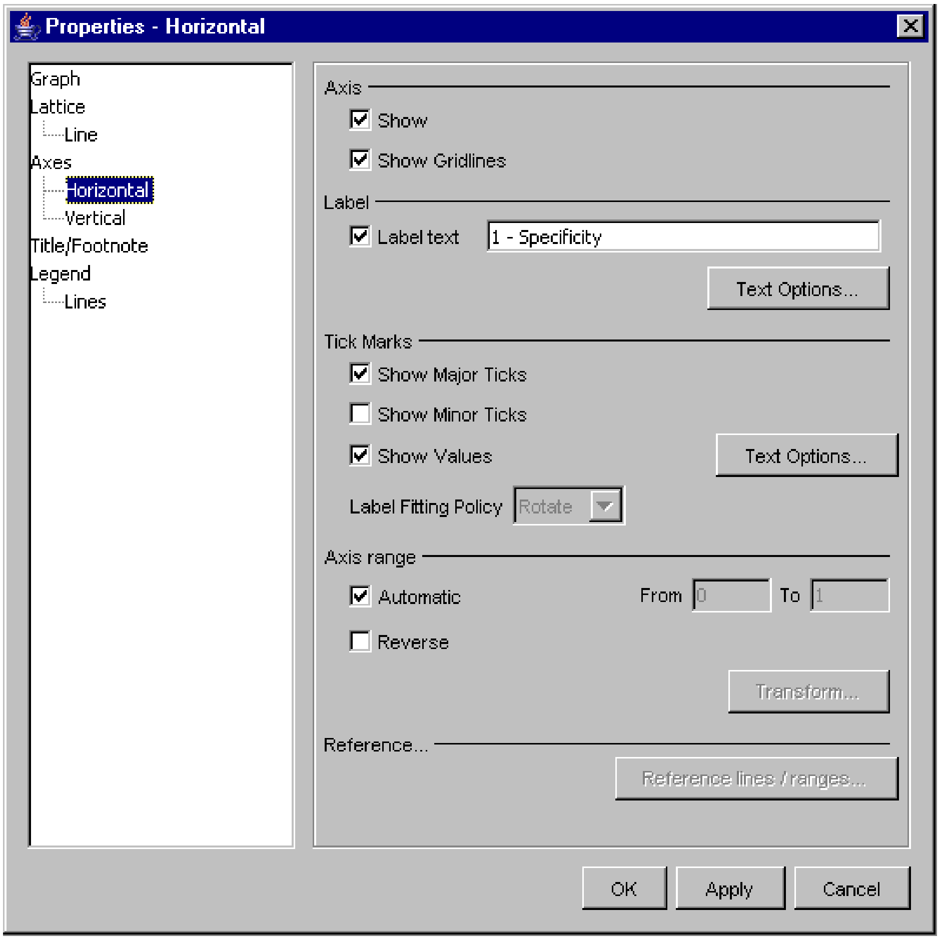
External Axes

□ Show

The **Lattice** section enables you to specify layout and axis equalization information. Click on a subgroup, such as Line, to specify additional information in the **Join** and **Markers** tabs. You can specify whether you want to skip missing values, or show markers.

- **Axes**

Display 18.30 *Axes*



☑ Show

☑ Show Gridlines

☑ Label text: 1 - Specificity

☑ Show Major Ticks

□ Show Minor Ticks

☑ Show Values

Label Fitting Policy: Rotate

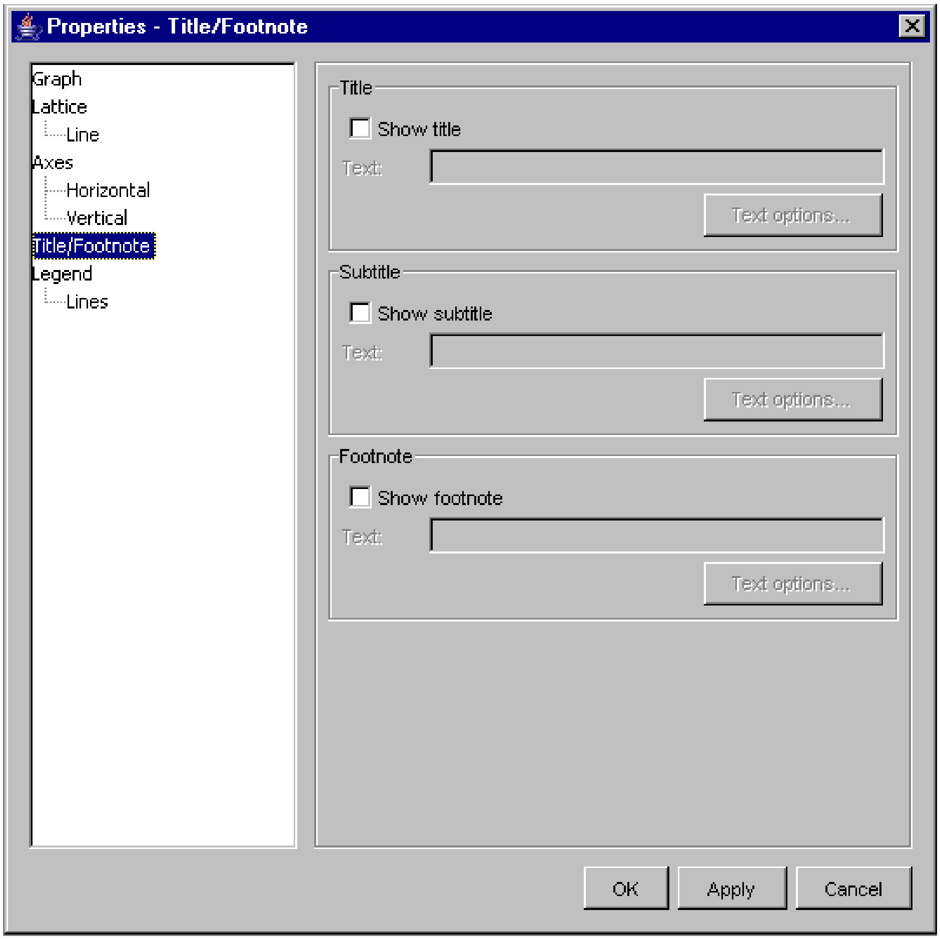
☑ Automatic From 0 To 1

□ Reverse

The **Axes** section enables you to specify a variety of formatting options for the horizontal and vertical axes.

- **Title/Footnote**

Display 18.31 *Title/Footnote*



Title

□ Show title

Text:

Subtitle

□ Show subtitle

Text:

Footnote

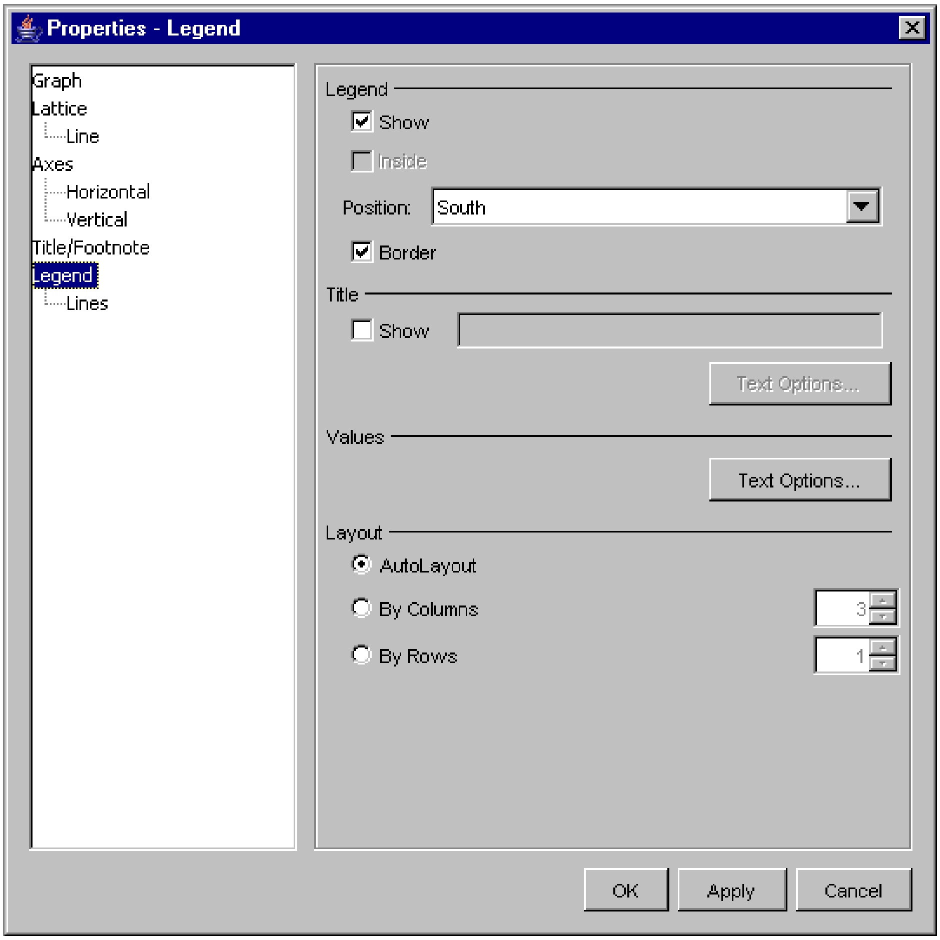
□ Show footnote

Text:

The **Title/Footnote** section enables you to add a Title, Subtitle, and Footnotes for the selected chart. When you enable title, subtitle, or footnote entities, you can compose the text that is displayed and the characteristics of the font that you want to use to display the text.

- **Legend**

Display 18.32 *Legend*



Legend

☑ Show

□ Inside

Position: South

☑ Border

Title

□ Show

Values

Layout

● AutoLayout

○ By Columns 3

○ By Rows 1

The **Legend** section enables you to control whether the legend is displayed for the selected chart, and, if so, where and how it is displayed.

Exploring SAS Tables

SAS Enterprise Miner enables you to view tables and graphs like those that you view with SAS/INSIGHT software. This feature is designed for the exploration of your data through graphs and analyses that are linked across multiple windows. You can perform tasks such as these:

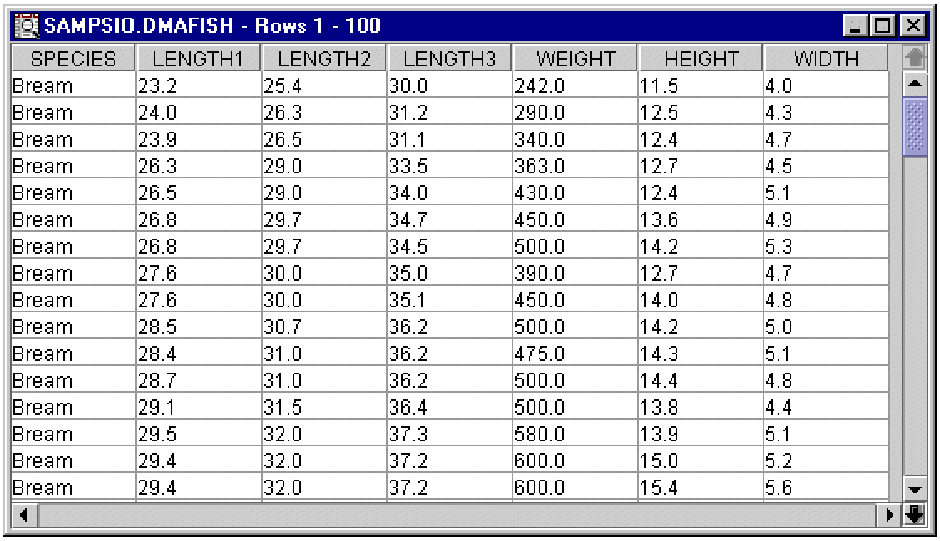
- analyze univariate distributions
- investigate multivariate distributions
- create scatter plots
- display pie charts

Opening a SAS Table

Follow these steps to open a SAS table:

1. Select **View** ⇨ **Explorer** from the main menu. The SAS Explorer window appears.
2. Select a library and a table in the window. In the example below, the data set SAMPSIO.DMAFISH is used.
3. Double-click a table to view the contents. Alternatively, right-click and select **Open** to browse the contents of a table.

Display 18.33 *SAS Table*

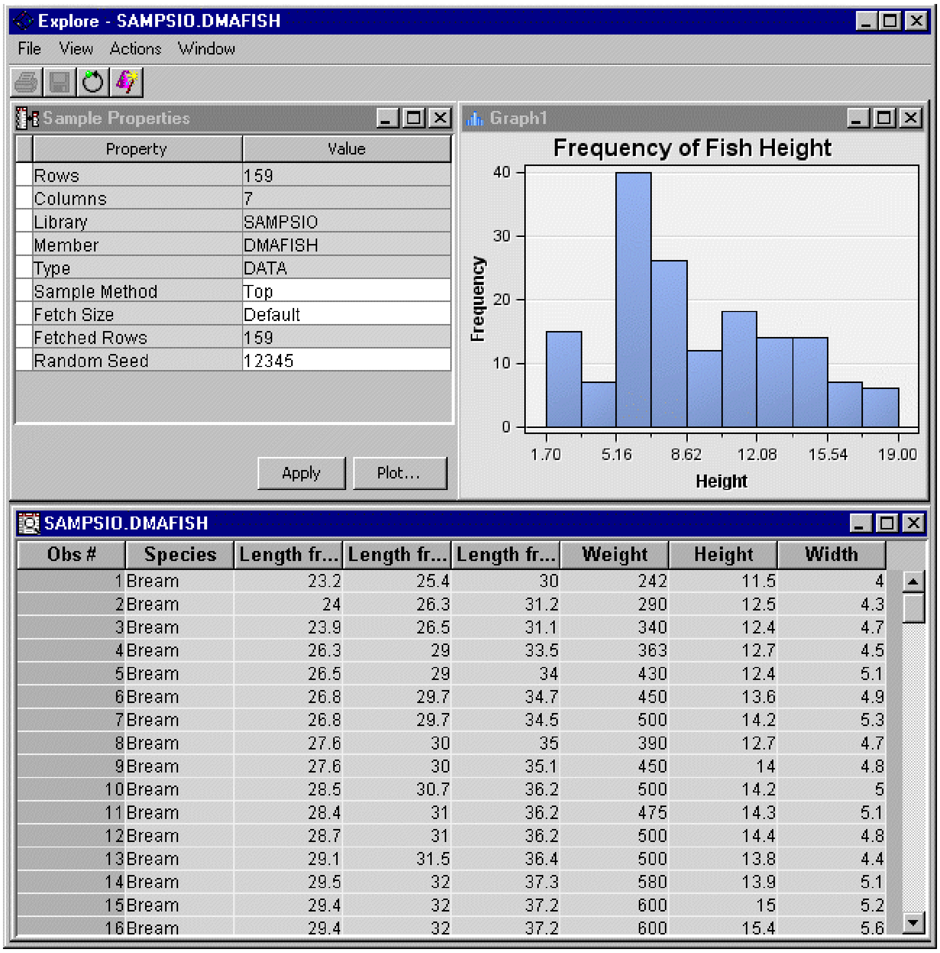


SAMPSIO.DMAFISH - Rows 1 - 100

SPECIES	LENGTH1	LENGTH2	LENGTH3	WEIGHT	HEIGHT	WIDTH
Bream	23.2	25.4	30.0	242.0	11.5	4.0
Bream	24.0	26.3	31.2	290.0	12.5	4.3
Bream	23.9	26.5	31.1	340.0	12.4	4.7
Bream	26.3	29.0	33.5	363.0	12.7	4.5
Bream	26.5	29.0	34.0	430.0	12.4	5.1
Bream	26.8	29.7	34.7	450.0	13.6	4.9
Bream	26.8	29.7	34.5	500.0	14.2	5.3
Bream	27.6	30.0	35.0	390.0	12.7	4.7
Bream	27.6	30.0	35.1	450.0	14.0	4.8
Bream	28.5	30.7	36.2	500.0	14.2	5.0
Bream	28.4	31.0	36.2	475.0	14.3	5.1
Bream	28.7	31.0	36.2	500.0	14.4	4.8
Bream	29.1	31.5	36.4	500.0	13.8	4.4
Bream	29.5	32.0	37.3	580.0	13.9	5.1
Bream	29.4	32.0	37.2	600.0	15.0	5.2
Bream	29.4	32.0	37.2	600.0	15.4	5.6

4. Right-click a table in the Explore window, and select **Explore**. Use the Explore window to view variable distributions. For more information, see "Exploring Variables" on page 280.

Display 18.34 *Explore Window*



Property	Value
Rows	159
Columns	7
Library	SAMPSIO
Member	DMAFISH
Type	DATA
Sample Method	Top
Fetch Size	Default
Fetched Rows	159
Random Seed	12345

SAMPSIO.DMAFISH

Obs #	Species	Length fr...	Length fr...	Length fr...	Weight	Height	Width
1	Bream	23.2	25.4	30	242	11.5	4
2	Bream	24	26.3	31.2	290	12.5	4.3
3	Bream	23.9	26.5	31.1	340	12.4	4.7
4	Bream	26.3	29	33.5	363	12.7	4.5
5	Bream	26.5	29	34	430	12.4	5.1
6	Bream	26.8	29.7	34.7	450	13.6	4.9
7	Bream	26.8	29.7	34.5	500	14.2	5.3
8	Bream	27.6	30	35	390	12.7	4.7
9	Bream	27.6	30	35.1	450	14	4.8
10	Bream	28.5	30.7	36.2	500	14.2	5
11	Bream	28.4	31	36.2	475	14.3	5.1
12	Bream	28.7	31	36.2	500	14.4	4.8
13	Bream	29.1	31.5	36.4	500	13.8	4.4
14	Bream	29.5	32	37.3	580	13.9	5.1
15	Bream	29.4	32	37.2	600	15	5.2
16	Bream	29.4	32	37.2	600	15.4	5.6

Setting the Sample Properties

Before creating graphs, you should sample the data set. Sampling reduces the processing time that is required to create the graphs and is especially important if you are creating graphs from a large data set. You can choose to sample a data set during data source creation on page 293 or by using the Sampling Node. For more information about Sampling, see the "Sample Node" on page 387 documentation.

In the Explore window shown above, the Sample Properties window is located in the upper left. You can view or modify the following properties in the Sample Properties window.

- **Data Properties** — You can view the following Data properties:
  - **Rows** — the number of rows in the data set.
  - **Columns** — the number of columns in the data set.
  - **Library** — the name of the SAS library that contains the data set.
  - **Member** — the name of the table that you selected.
  - **Type** — the data type of the table. The type can be DATA or VIEW.
- **Sample Method** — You can choose the sampling method that you want to apply to the data set. After you finish modifying the sample properties, click **Apply**. The table is updated with the new sample.
  - **Top** — selects the top (first) rows of the data set.
  - **Random** — uses the random-sampling method.
- **Fetch Size** — the number of observations that are read in from memory when you apply the sample properties. Valid values are **Default** and **Max**. If a data set contains n observations, the Default fetch size is the lesser of n and 2000 rows, and the **Max** fetch size is the lesser of n and 20,000 rows.

# Obs in Data Set	Default Fetch Observations	Max Fetch Observations
159	159	159
2,500	2,000	2,500
30,000	2,000	20,000
100,000	2,000	20,000

  If you want to specify a custom fetch size (such as 50,000 observations) to be used in Explore windows during an Enterprise Miner session, you can use the EM_EXPLOREOBS_MAX macro variable to submit a statement via Program Manager or your start file:

  ```
  %let EM_EXPLOREOBS_MAX=50000;
  ```

  Note: Using the EM_EXPLOREOBS_MAX macro variable to specify very large fetch sizes can cause sluggish performance.
- **Fetched Rows** — the actual size of the sample that is currently displayed in the Explore window.
- **Random Seed** — the random seed that is used to generate the sample.

Creating Graphs

Use one of the following methods to open the **Graph** wizard.

- Select **Actions** ⇨ **Plot** from the main menu of the Explore window.
- Click the icon on the Explore or Results window toolbar.
- Select **View** ⇨ **Plot** from the main menu of the Results window.

The Select a Chart Type window appears.

Types of Graphs

You can create different types of graphs, depending on the structure of your data.

- Scatter Plots on page 271
- Line Plots on page 272
- Histograms on page 273
- Density Plots on page 273
- Box Plots on page 273
- Tables on page 274
- Matrix Plots on page 274
- Lattice Plots on page 274
- Parallel Axis Plots on page 276
- Constellation Plots on page 276
- 3D Charts on page 276
- Contour Plots on page 276
- Bar Charts on page 276
- Pie Charts on page 277
- Needle Charts on page 277
- Vector Plots on page 277
- Band Plots on page 277

Scatter Plots

A scatter plot displays data points on a two-dimensional graph or in three dimensions, and is considered a preliminary analysis tool for fitting a regression curve. A scatter plot is particularly useful when you have a large number of data points and you want to see the relationship between an explanatory variable that is plotted on the x-axis and a response variable that is plotted on the y-axis.

There are four common ways to use scatter plots for analysis. First, you can judge the strength of the relationship between the explanatory and the response variable. Second, you can judge the shape of the best fit curve for the group of points. Third, you can estimate the direction of the curve, and whether it is positive or negative. Finally, you can observe the presence of outliers.

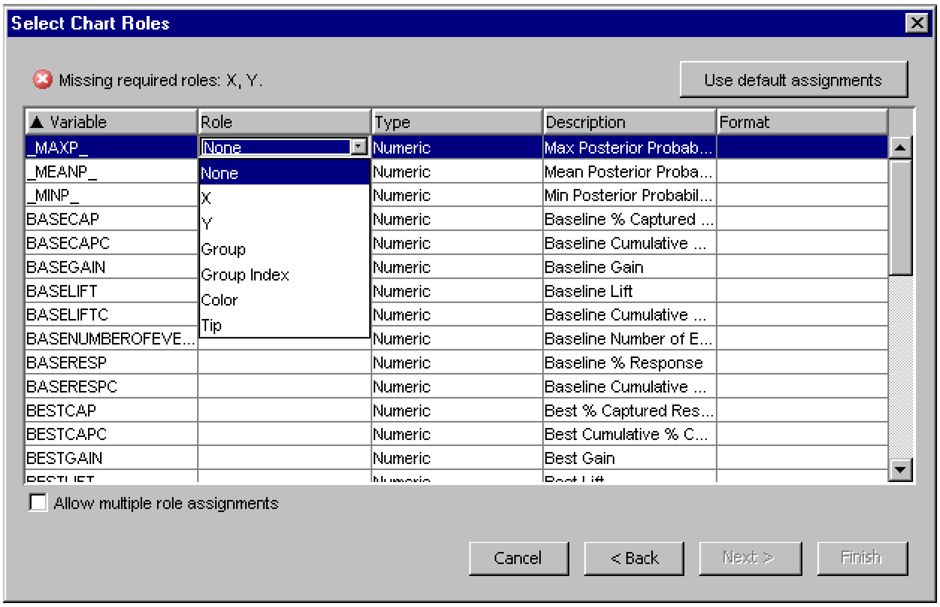
To create a scatter plot, follow these steps:

1. Select **Scatter** from the list of options in the Select a Chart Type window.
2. Choose from the following options:

- plot values of two variables against each other
- plot values of two variables against each other and connect data values with a line
- generate a scatter plot in three dimensions

3. Click **Next** to open the Select Chart Roles window. Depending on which type of plot you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your plot. See the following screen capture for an illustration of the available data roles:

***Display 18.35** Select Chart Roles*



Select Chart Roles

Missing required roles: X, Y.

▲ Variable	Role	Type	Description	Format
MAXP	None	Numeric	Max Posterior Probab...	
MEANP		Numeric	Mean Posterior Proba...	
MINP		Numeric	Min Posterior Probabil...	
BASECAP		Numeric	Baseline % Captured ...	
BASECAPC		Numeric	Baseline Cumulative ...	
BASEGAIN		Numeric	Baseline Gain	
BASELIFT		Numeric	Baseline Lift	
BASELIFTC		Numeric	Baseline Cumulative ...	
BASENUMBEROFEVE...		Numeric	Baseline Number of E...	
BASERESP		Numeric	Baseline % Response	
BASERESPC		Numeric	Baseline Cumulative ...	
BESTCAP		Numeric	Best % Captured Res...	
BESTCAPC		Numeric	Best Cumulative % C...	
BESTGAIN		Numeric	Best Gain	
BESTLIFT		Numeric	Best Lift	

□ Allow multiple role assignments

4. For the remaining steps, see "Completing Your Chart" on page 278.

Line Plots

A line plot illustrates the relationship between two variables with a connected line on a two-dimensional graph, and is considered a simple analysis tool to observe the relationship between two variables.

To create a line plot, follow these steps:

1. Select **Line** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - plot values of two variables against each other and connect data values with a line
   - plot the summarized y value against each x value and connect data values with a line
3. Click **Next** to open the Select Chart Roles window. Depending on which type of plot you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your plot.
4. For the remaining steps, see "Completing Your Chart" on page 278.

Histograms

A histogram contains bars that show the distribution of data values by the area of a bar. Histograms are useful to observe the frequency of certain values and trends.

To create a histogram, follow these steps:

1. Select **Histogram** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - a histogram that provides statistical relations between X values that are grouped as bins or discrete
   - a parameterized histogram, which assumes that data is already pre-binned, provides statistical relations between X values that are grouped as bins or discrete, and only processes one observation per bin
   - a histogram that provides statistical relations between X and Y values that are grouped as bins or discrete
3. Click **Next** to open the Select Chart Roles window. Depending on which type of plot you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your plot.
4. For the remaining steps, see “Completing Your Chart” on page 278.

Density Plots

A density plot shows the distribution of one or more variables.

To create a density plot, follow these steps:

1. Select **Density** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - show the distribution of one variable on the x-axis
   - show the distribution of one variable on the y-axis
   - show the distribution of two variables plotted against each other
3. Click **Next** to open the Select Chart Roles window. Depending on which type of plot you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your plot.
4. For the remaining steps, see “Completing Your Chart” on page 278.

Box Plots

A box plot illustrates the five-number summary of a data set, which includes (1) the smallest observation, (2) the lower quartile, (3) the median, (4) the upper quartile, and (5) the largest observation. In addition, outliers are also typically included on box plots.

To create a box plot, follow these steps:

1. Select **Box** from the list of options in the Select a Chart Type window.
2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.
3. For the remaining steps, see “Completing Your Chart” on page 278.

Tables

You can obtain a table of generated statistics for each model that you are comparing.

To create a table, follow these steps:

1. Select **Tables** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - simple table.
   - GTable, which is a standard Table component with some graphical features added to it. The GTable component supports several ways of rendering cells for different types of data.
3. Click **Finish** to create your table.

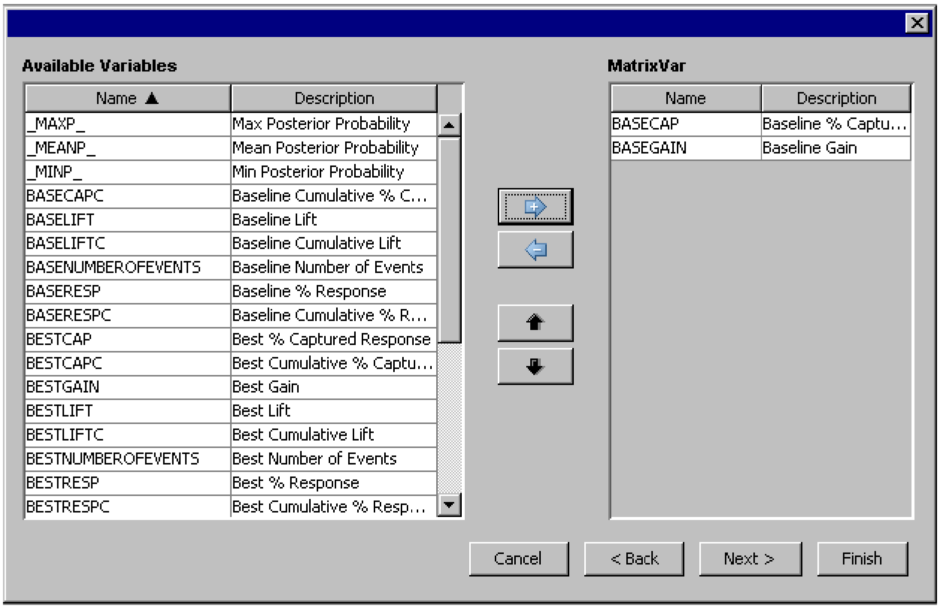
Matrix Plots

Matrix plots enable you to plot multiple interval variables at one time, resulting in a matrix of plots within a single graph window.

To create a matrix plot, follow these steps:

1. Select **Matrix** from the list of options in the Select a Chart Type window.
2. Click **Next** to open a window that enables you to select available variables. You need to select at least two variables from the available variables pane to include in the MatrixVar pane. The chart uses the assigned variables to build a plot matrix such that each variable is assigned a row and column.

   Display 18.36 Select Variables



Available Variables

Name ▲	Description
MAXP	Max Posterior Probability
MEANP	Mean Posterior Probability
MINP	Min Posterior Probability
BASECAPC	Baseline Cumulative % C...
BASELIFT	Baseline Lift
BASELIFTC	Baseline Cumulative Lift
BASENUMBEROFEVENTS	Baseline Number of Events
BASERESP	Baseline % Response
BASERESPC	Baseline Cumulative % R...
BESTCAP	Best % Captured Response
BESTCAPC	Best Cumulative % Captu...
BESTGAIN	Best Gain
BESTLIFT	Best Lift
BESTLIFTC	Best Cumulative Lift
BESTNUMBEROFEVENTS	Best Number of Events
BESTRESP	Best % Response
BESTRESPC	Best Cumulative % Resp...

MatrixVar

Name	Description
BASECAP	Baseline % Captu...
BASEGAIN	Baseline Gain

3. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.
4. Click **Next** in the Select Chart Roles window. The Data WHERE Clause opens. Use this window to create a WHERE clause that you can use to subset your data.
5. Click **Finish** to create your matrix plot.

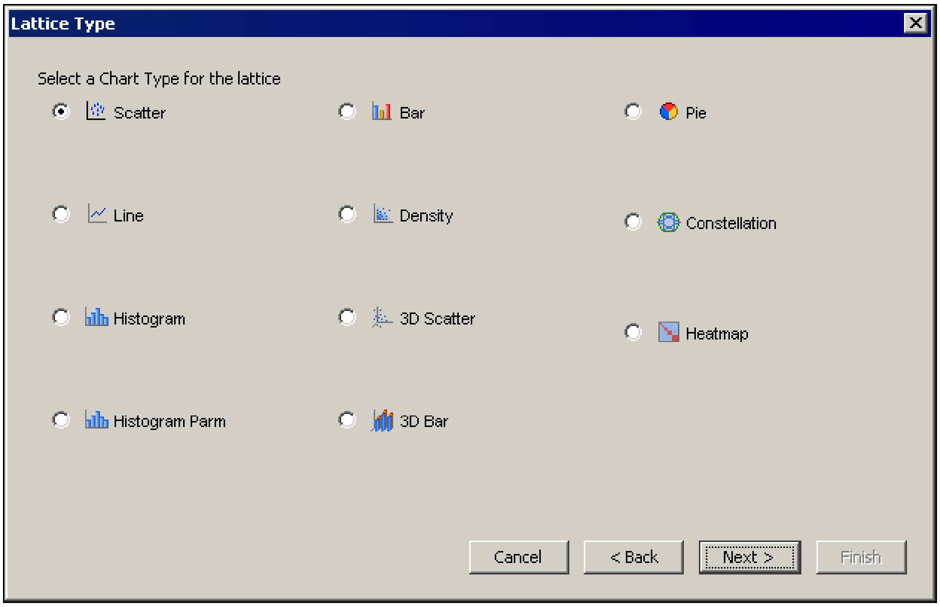
Lattice Plots

Lattice plots build a separate chart for each unique value of a BY variable.

To create a lattice plot, follow these steps:

1. Select **Lattice** from the list of options in the Select a Chart Type window.
2. Choose from among the following lattice plot options:
   - The X-Lattice lays the charts out horizontally.
   - The Y-Lattice lays the charts out vertically.
   - The 2-D Lattice has two by-variables, and builds a separate chart for each combination of the unique by-variable values.
3. Click **Next** to open the Lattice Type window. Select either a scatter, line, histogram, histogram parm, bar, density, 3-D scatter, 3-D bar, or pie type.

Display 18.37 *Lattice Type*



Lattice Type

Select a Chart Type for the lattice

● Scatter ○ Bar ○ Pie

○ Line ○ Density ○ Constellation

○ Histogram ○ 3D Scatter ○ Heatmap

○ Histogram Parm ○ 3D Bar

4. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.
5. Click **Next** in the Select Chart Roles window. The Data WHERE Clause opens. Use this window to create a WHERE clause that you can use to subset your data.
6. Click **Finish** to create your lattice plot.

Parallel Axis Plots

A parallel axis plot is used to plot large multivariate data sets. In a parallel axis plot, each axis represents one variable in the data set, and each observation is plotted as a line connecting its points on each axis.

To create a parallel axis plot, follow these steps:

1. Select **Parallel Axis** from the list of options in the Select a Chart Type window.
2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.
3. For the remaining steps, see "Completing Your Chart" on page 278.

Constellation Plots

A parallel axis plot is used to plot large multivariate data sets. In a parallel axis plot, each axis represents one variable in the data set, and each observation is plotted as a line connecting its points on each axis.

To create a parallel axis plot, follow these steps:

1. Select **Constellation** from the list of options in the Select a Chart Type window.
2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot
3. For the remaining steps, see "Completing Your Chart" on page 278.

3-D Charts

A 3-D chart shows a three dimensional illustration of statistical results.

To create a 3-D chart, follow these steps:

1. Select **3-D Charts** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - a surface of gridded 3-D data
   - a bar chart with both a category and series variable
   - a scatter plot in three dimensions
3. Click **Next** to open the Select Chart Roles window. Depending on which type of chart you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your chart.
4. For the remaining steps, see "Completing Your Chart" on page 278.

Contour Plots

Contour plots are two-dimensional plots that show three-dimensional relationships. They use contour lines or patterns to represent levels of magnitude for a contour variable that is plotted on the horizontal and vertical axes.

To create a contour plot, follow these steps:

1. Select **Contour** from the list of options in the Select a Chart Type window.
2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.
3. For the remaining steps, see "Completing Your Chart" on page 278.

Bar Charts

A bar chart provides statistical relations between categorical values. A bar can be subdivided or grouped by using a subgroup variable or a group variable respectively.

To create a bar chart, follow these steps:

1. Select **Bar** from the list of options in the Select a Chart Type window.
2. Choose from the following options:
   - a bar chart on the x-axis
   - a bar chart on the y-axis
   - a bar chart with both a category and series variable

3. Click **Next** to open the Select Chart Roles window. Depending on which type of chart you selected in the Select a Chart Type window, you might be asked to assign different data roles. Assign the roles for the statistics that you want to include on your chart.

4. For the remaining steps, see "Completing Your Chart" on page 278.

Pie Charts

A pie chart shows each category's contribution to a sum.

To create a pie chart, follow these steps:

1. Select **Pie** from the list of options in the Select a Chart Type window.

2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your chart.

3. For the remaining steps, see "Completing Your Chart" on page 278.

Needle Charts

A needle chart draws a vertical line from the XY point to a baseline.

To create a needle chart:

1. Select **Needle** from the list of options in the Select a Chart Type window.

2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your chart.

3. For the remaining steps, see "Completing Your Chart" on page 278.

Needle Plots

A needle chart draws a vertical line from the XY point to a baseline.

To create a need chart, follow these steps:

1. Select **Needle** from the list of options in the Select a Chart Type window.

2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your chart.

3. For the remaining steps, see "Completing Your Chart" on page 278.

Vector Plots

A vector plot draws vectors to the XY point. The vectors can be drawn either from a common origin or from a start x and y variable.

To create a vector plot, follow these steps:

1. Select **Vector** from the list of options in the Select a Chart Type window.

2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.

3. For the remaining steps, see "Completing Your Chart" on page 278.

Band Plots

A band plot fills a region that is defined by an upper and lower variable.

To create a band plot, follow these steps:

1. Select **Band** from the list of options in the Select a Chart Type window.

2. Click **Next** to open the Select Chart Roles window. Assign the roles for the statistics that you want to include on your plot.

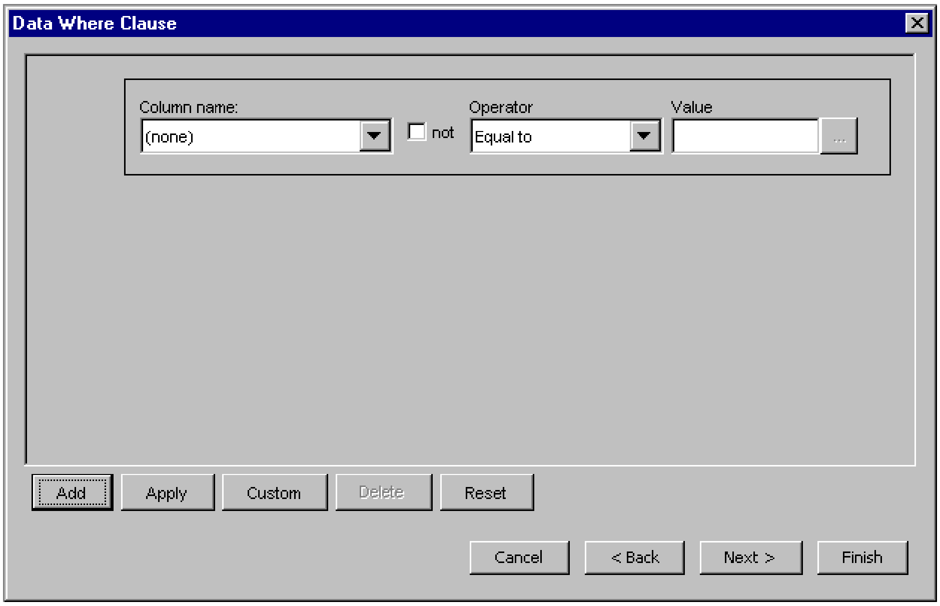
3. For the remaining steps, see "Completing Your Chart" on page 278.

Completing Your Chart

Follow these steps to subset your data, and to add titles and legend information to you chart.

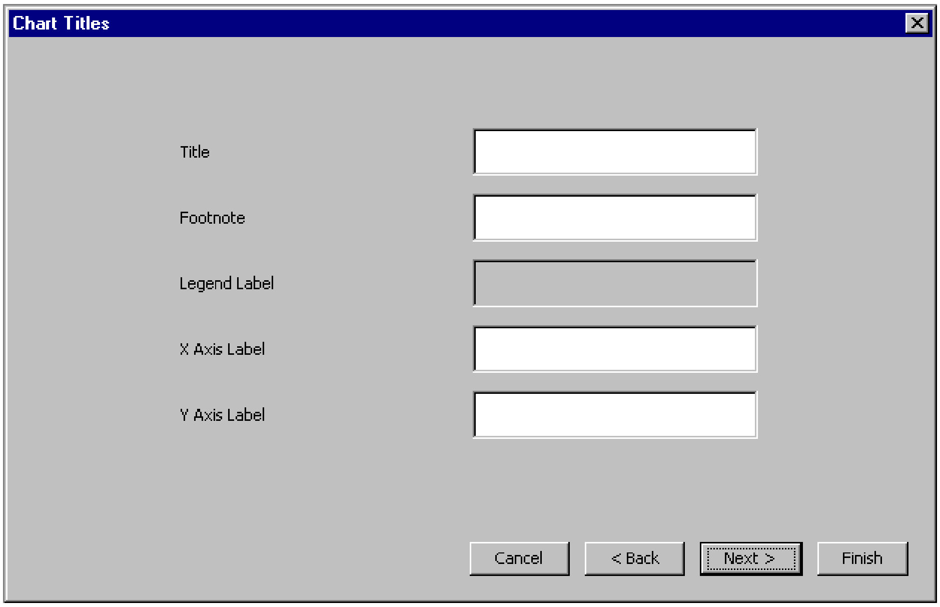
1. Click **Next** in the Select Chart Roles window. The Data Where Clause window appears.

Display 18.38 *Data WHERE Clause Window*



2. Use the Data Where Clause window to create a WHERE clause that you can use to subset your data. Click **Next**. The Chart Titles window appears.

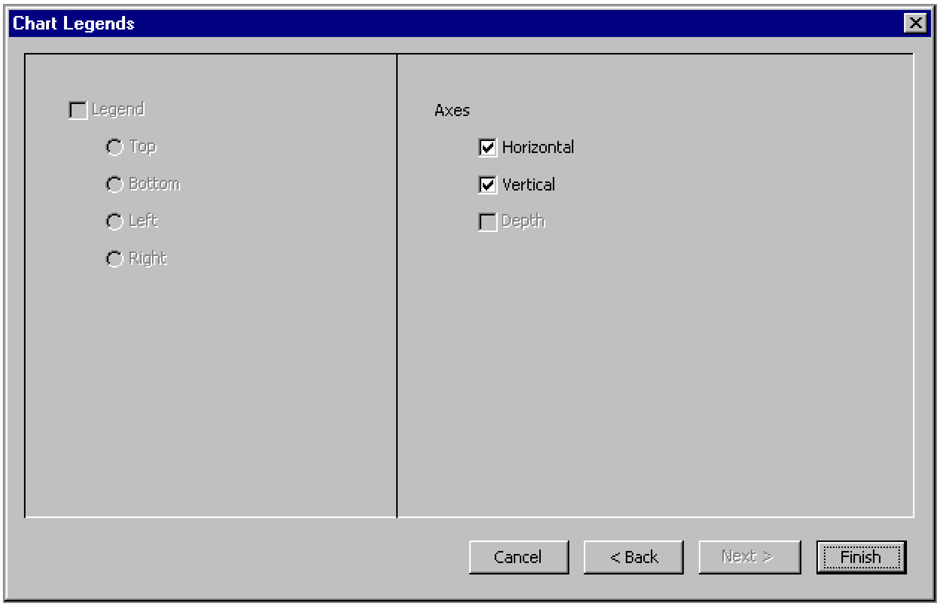
Display 18.39 *Chart Title Window*



3. Use the Chart Titles window to specify the following:
   - title
   - footnote
   - legend label
   - x axis label
   - y axis label

   Click **Next**. The Chart Legends window appears.

   Display 18.40 *Chart Legend*



4. Use the Chart Legends window to do the following tasks:
   - determine whether a legend is displayed
   - determine the placement of the legend
   - determine whether the horizontal and vertical axes are displayed
5. Click **Finish**.

Viewing Multiple Graphs at the Same Time

You can view a number of graphs at the same time. In addition, you can select data points in a graph or a table to highlight them. The data points that you select are displayed in the remaining graphs.

To view multiple graphs at the same time, select **Window** ⇨ **Tile** from the Explore or Results window main menu. Any graphs that have not been minimized are displayed so that all windows are displayed at the same time.

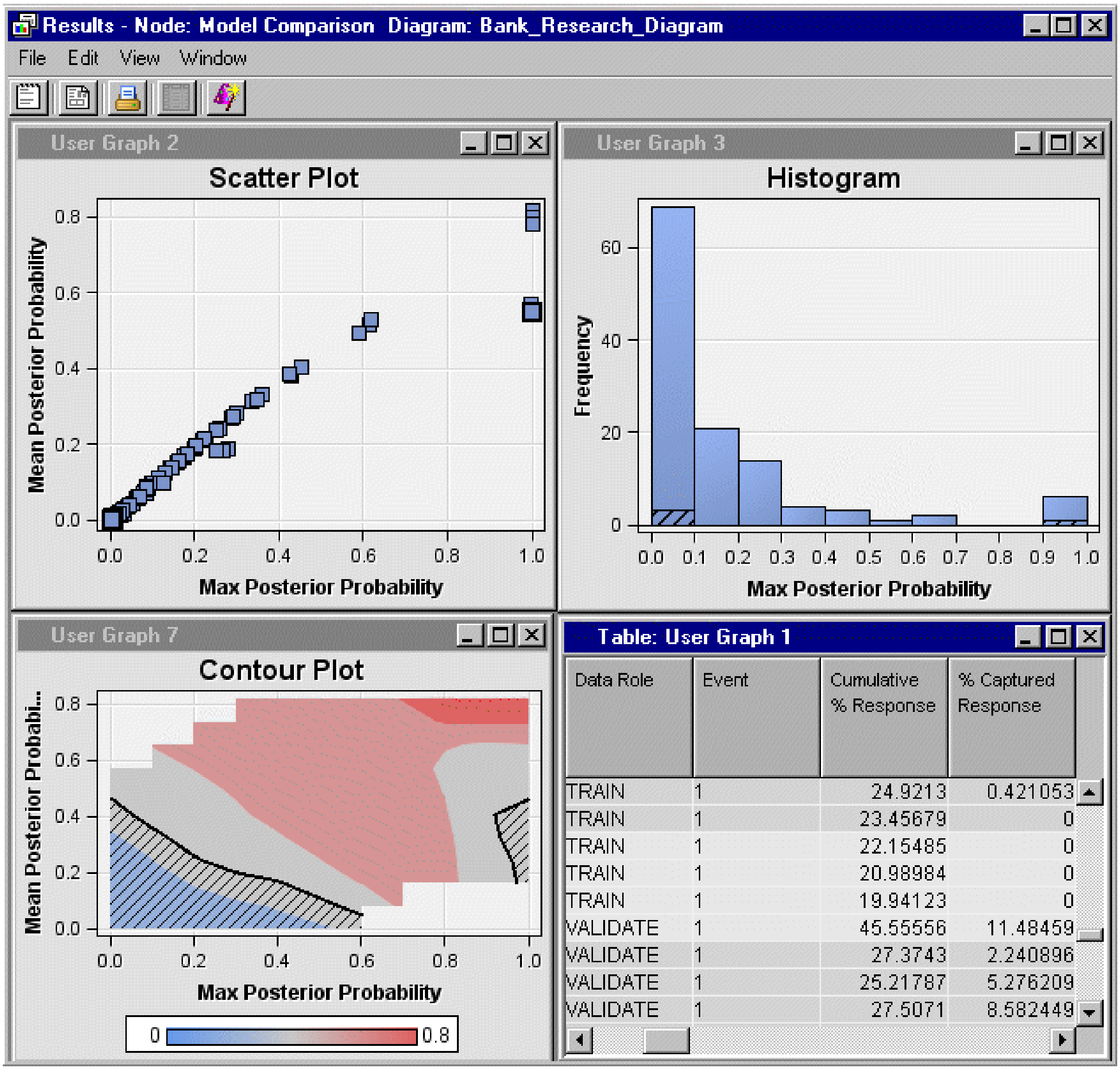
You can select observations by taking the following actions:

- Click a data point on a graph.
- Create a selection box on a graph to highlight multiple observations.
- Click a row in the table.
- Hold down CTRL while you select multiple rows in the table.

- Hold down SHIFT and select a group of rows in the table.

The following illustrates how selecting multiple rows in a table can highlight information in multiple graphs with a subset of graphs and tables that have been created.

Display 18.41 *View Multiple Graphs*



Data Role	Event	Cumulative % Response	% Captured Response
TRAIN	1	24.9213	0.421053
TRAIN	1	23.45679	0
TRAIN	1	22.15485	0
TRAIN	1	20.98984	0
TRAIN	1	19.94123	0
VALIDATE	1	45.55556	11.48459
VALIDATE	1	27.3743	2.240896
VALIDATE	1	25.21787	5.276209
VALIDATE	1	27.5071	8.582449

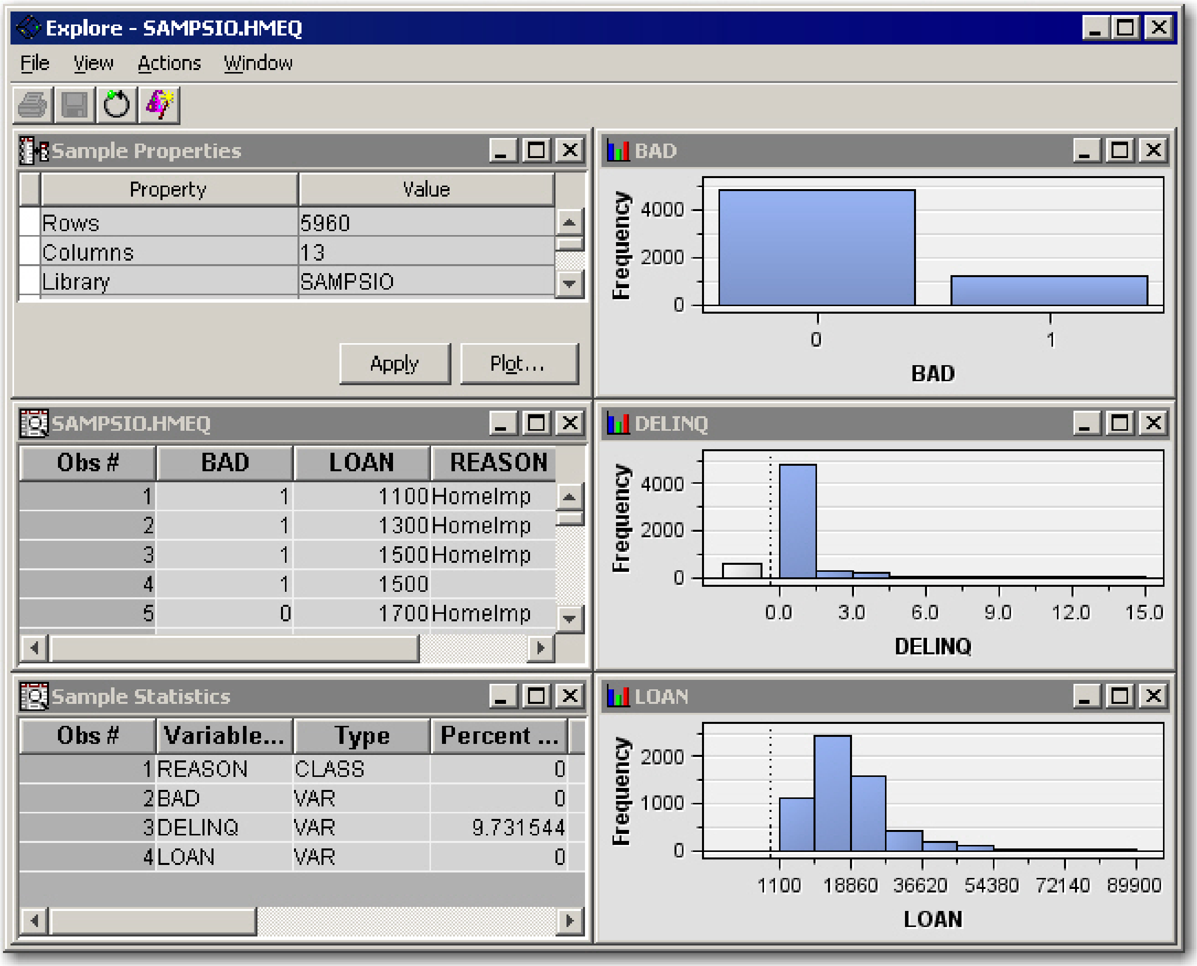
Exploring Variables

It can be useful to view the metadata or distribution plots of variables of the incoming data for a node. To view these metadata or distribution plots:

1. Select a node in your diagram.
2. Click the ... button of the Variables property.
3. In the Variables window, select the variable or variables that you want to explore. Then, click **Explore** to open the Explore window.

The following example of the Explore window shows four variables, BAD, LOAN, REASON, DELINQ, that were selected from a data set that is going to be partitioned.

Display 18.42 *Explore Window*



Property	Value
Rows	5960
Columns	13
Library	SAMPSIO

Obs #	BAD	LOAN	REASON
1	1	1100	HomeImp
2	1	1300	HomeImp
3	1	1500	HomeImp
4	1	1500	
5	0	1700	HomeImp

Obs #	Variable...	Type	Percent ...
1	REASON	CLASS	0
2	BAD	VAR	0
3	DELINQ	VAR	9.731544
4	LOAN	VAR	0

Note: The number of observations that are loaded on the client machine depends on the record length of the selected variables. The record length is the sum of the variable length for all the selected variables. More observations are loaded for smaller record lengths. The default and maximum sizes for different record lengths were selected to balance network traffic and interactive responsiveness. For example, if you want to use 10,000 observations for the Explore window, you can reset the maximum number of observations to display by inserting the following macro variable in your SAS Enterprise Miner project start code:

```
%let EM_EXPLOREOBS_MAX=10000;
```

If you make this change, performance might degrade. Scatter plots and table views might become sluggish since all 10,000 rows will be processed, even if they are not individually displayed in plots.

- The Sample Properties window displays details about the data. For more information, see “Using the Diagram Editor” on page 240.
- The Sample Statistics window displays the variable, variable type, percent of data missing, minimum value, maximum value, number of levels, percent of data on the mode, and the mode of the data. For smaller tables, the entire data set might be sampled. However, for larger data sets only a sample of the data is taken based on the Sample Properties.

- To view the raw data that was sampled, select **View** ⇨ **Table** from the main menu. In the picture above, this is the window titled SAMPSIO.HMEQ.
- By default, histograms are displayed for the variables that you select. You can create various types of graphs for a variable. For more information, see "Creating Graphs" on page 271.

Using the SAS Enterprise Miner Program Editor, Log, and Output Windows

You can use the following windows to submit SAS code and view the results.

- SAS Enterprise Miner Program Editor window on page 282
- SAS Enterprise Miner Log window on page 282
- SAS Enterprise Miner Output window on page 282

SAS Enterprise Miner Program Editor Window

Use the Program Editor window in the SAS Enterprise Miner user interface to submit SAS code. You can use the Program Editor to do the following tasks:

- create and submit macros
- submit any other code that you write

You can clear the code from the Program Editor window by selecting **Edit** ⇨ **Clear All** from the main menu.

Note: The Enterprise Miner Program Editornew window is not designed to submit SAS Enterprise Miner batch-processing code. SAS Enterprise Miner batch-processing code should be submitted in either a SAS batch job or through the native SAS Program Editor.

CAUTION:

Non-batch code that is submitted through the SAS Enterprise Miner Program Editor must not change the assignment of any SAS Enterprise Miner libraries, including the EMDS (data sources) and EMWS (workspaces) libraries. Changing the assignment of these libraries causes SAS Enterprise Miner to operate incorrectly, and SAS Enterprise Miner might not recognize data sources and projects.

SAS Enterprise Miner Log Window

The Enterprise Miner Log window displays the log that is generated when you do either of the following tasks:

- start a SAS Enterprise Miner session
- submit code in the Enterprise Miner Program Editor window.

Note: The SAS Enterprise Miner Log window does not display the SAS log from the submitted process flows. To view the SAS log, you view the Results window from the appropriate node

SAS Enterprise Miner Output Window

Use the Enterprise Miner Output window to view the results of code that you submit in the Enterprise Miner Program Editor window.

Locked Data Sources and Diagrams

When a client/server user creates or opens a SAS Enterprise Miner project or process flow diagram, internal processes that maintain the diagram and its data sources are owned by, or "locked," to that user. These locks prevent unauthorized users from opening another user's diagram, and they prevent authorized users from modifying more than one instance of the same diagram at a time. If a client/server connection for SAS Enterprise Miner is severed, or if the process flow diagram is abnormally interrupted or terminated, the project diagram and data source objects remain locked to the last user when SAS Enterprise Miner is restarted. When a user tries to resume a data mining flow that he or she owns, it will still be locked because the locked objects are locked to the user in a session that no longer exists.

The locks are released after a time-out period, which is specified as an option when the mid-tier Java Virtual Machine is started. The lock lease expiration time is specified by the Java Server, where it is set using the java.rmi.dgc.leaseValue. The leaseValue time is set in milliseconds. The default setting for the lock lease expiration property is 600,000 milliseconds, or ten minutes. To change the lock setting to a shorter interval, such as 30 seconds, modify the Java server start file using a statement such as:

```
java -Djava.rmi.dgc.leaseValue=30000 ServerMain
```

In the statement, 30000 is the number of milliseconds that are required for a 30-second time-out period before the locks are released.

SAS Server Unsupported Actions and Language Elements for SAS Enterprise Miner

Double Quotation Marks on the Command Line: The SAS Enterprise Miner SAS server command interpreter does not support the use of double quotation marks (" "). When you submit code to SAS through the SAS Enterprise Miner command line, use single quotation marks (' ') instead of double quotation marks.

Part 8

Data Sources

Chapter 19
Enterprise Miner Data Sources

SAS Enterprise Miner Data Sources

Overview

A data source in SAS Enterprise Miner defines all the information about a SAS data set that is needed for data mining. This information includes the name and location of the data set, the SAS code that is used to define a library path, the variable roles, the measurement levels, and other attributes that guide the data mining process.

Data sources are essential to a SAS Enterprise Miner project. Features of data sources include:

- A data source can be used in any diagram within a project.
- A data source can be copied from one project to another.
- Data sources that are defined in a project are stored in the same directory, the DataSources directory.

You create data sources either by using the **Data Source** wizard or by submitting SAS code.

Alternatively, administrators can create data sources for users.

Note: You cannot define a data source based on a data set in the WORK library.

SAS Libraries and Data Sources

SAS Enterprise Miner Data Sources Library: EMDS

SAS Enterprise Miner reads data sources from an EMDS (SAS Enterprise Miner Data Sources) library. Data sources are not the actual training data, but are the metadata that defines the source data. The source data must exist in some allocated LIBNAME (for example, SAMPSIO). The source data LIBNAME allocation should be defined in either the SAS Enterprise Miner server start code (preferred), or in the SAS Enterprise Miner project start code.

SAS Enterprise Miner Local Data Sources Library: EMLDS

SAS Enterprise Miner creates an EMLDS (SAS Enterprise Miner Local Data Sources) library in your project. Each project will have its own EMLDS library.

SAS Enterprise Miner Global Data Sources Library: EMGDS

You can define the EMGDS (SAS Enterprise Miner Global Data Sources) library in your SAS Enterprise Miner server start code. The LIBNAME statement in the server start code might resemble

```
LIBNAME EMGDS "/projects/global_datasources";
```

Use the EMGDS library to support

- data source definitions that are shared between multiple users and in multiple projects.
- the creation of data sources by external processes such as nightly data integration jobs or query applications. These external processes can create data source definitions that can be used by SAS Enterprise Miner GUI users.

You can overwrite the assignment of the EMGDS library by including a different EMGDS LIBNAME in your project start code. For example, you could include the following in the code:

```
LIBNAME  EMGDS "projects/userID/my_global_datasources";
```

Using the SAS Enterprise Miner Data Sources Library

SAS Enterprise Miner creates the EMDS library by concatenating the EMLDS and EMGDS libraries:

```
LIBNAME  EMDS (EMLDS EMGDS);
```

Data source information is read from the EMDS library, instead of from the EMLDS and EMGDS libraries.

If you run SAS Enterprise Miner with the GUI, then you cannot write to the EMGDS library. Because of the EMDS LIBNAME concatenation, any changes that you make to a global data source are written to the EMLDS library. Therefore, changes to these data sources are made within the project. This enables you to modify a global data source without affecting the work of other users who are using the same data source.

You can remove the changes that you make to a global data source by either deleting the local changes, or by creating a new project.

%EMDS Macro

You can use the %EMDS macro to create data source definitions. You can create data source definitions in any directory, but creating them in the EMGDS location is one way to make them accessible by SAS Enterprise Miner. You can use this macro when you run external processes such as nightly data integration jobs or query applications. For more information about the %EMDS Macro, see SAS Enterprise Miner %EMDS Macro on page 1439 .

You can also create target profile definitions for a data source by using the %EMTP macro. See SAS Enterprise Miner %EMTP Target Profiler Macro on page 1441 for more information.

Contents of a Data Source

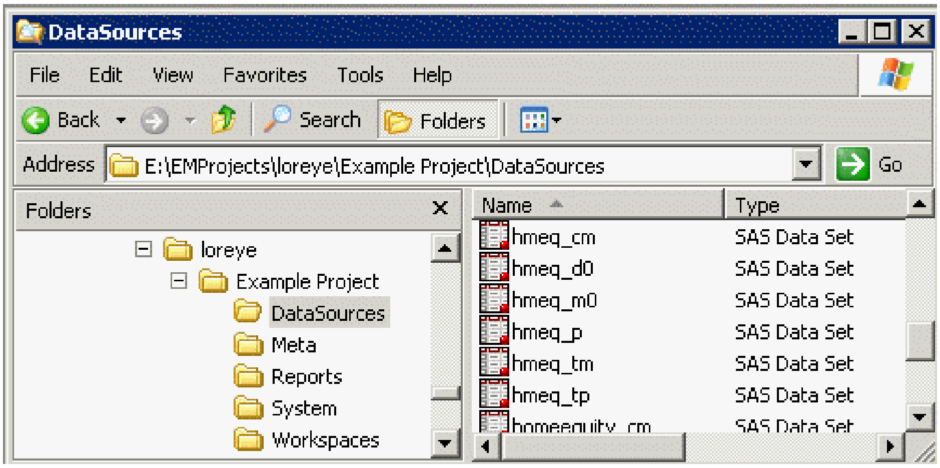
Overview

A data source consists of SAS data sets and text files. After you have created the necessary files and placed them in the correct location, the data source will appear in the Project Panel of the SAS Enterprise Miner window. You can cautiously edit these files to change the properties of a data source.

Each SAS Enterprise Miner project contains one data source subdirectory, which is named DataSources. All data sources that are defined in a project are stored in the DataSources subdirectory. By default, the DataSources subdirectory of a project is assigned the SAS library name EMDS.

Note: The DataSources subdirectory contains definitions of a data source (for example, the location of the actual data sources). The actual data sources do not reside in this subdirectory.

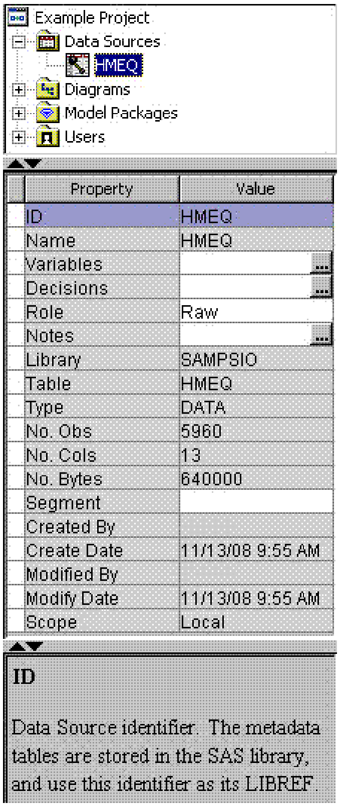
The following is an example of the contents of the DataSources subdirectory in your file system:



Address E:\EMProjects\loreye\Example Project\DataSources

Name	Type
hmeq_cm	SAS Data Set
hmeq_d0	SAS Data Set
hmeq_m0	SAS Data Set
hmeq_p	SAS Data Set
hmeq_tm	SAS Data Set
hmeq_tp	SAS Data Set
homeequity_cm	SAS Data Set

In the SAS Enterprise Miner project panel below, the data source HMEQ is defined for the project Example Project.



Property	Value
ID	HMEQ
Name	HMEQ
Variables	...
Decisions	...
Role	Raw
Notes	...
Library	SAMPSIO
Table	HMEQ
Type	DATA
No. Obs	5960
No. Cols	13
No. Bytes	640000
Segment	
Created By	
Create Date	11/13/08 9:55 AM
Modified By	
Modify Date	11/13/08 9:55 AM
Scope	Local

ID

Data Source identifier. The metadata tables are stored in the SAS library, and use this identifier as its LIBREF.

When you select a data source, the Properties Panel displays the data source properties. The property ID is the data source identifier. By default, it is generated from the data source name by ignoring numbers and special characters.

The DataSources subdirectory in your file system contains the following files for each defined data source. The prefix of these filenames is the data source ID.

- Tablemeta File on page 290 — is a SAS data set named <ID>_tm.
- Columnmeta File on page 291 — is a SAS data set named <ID>_cm.
- Target Profile File on page 292 — is a SAS data set named <ID>_tp.
- Decision Matrix Files on page 292 — are SAS data set named <ID>_d. Number can be 1, 2, 3..., depending on the number of target variables.
- Decision Variable Files on page 292 — are SAS data set named <ID>_m. Number can be 1, 2, 3..., depending on the number of target variables.
- Properties File on page 293 — is a text file named <ID>_properties.
- Notes File on page 293 — is a text file named <ID>_notes.

Tablemeta File

The tablemeta file is structured as a single row table of the form created by the CONTENTS procedure. It has information about the data source. For example, the tablemeta file of the HMEQ data source shown above contains data source name, library name, the time that the data source is created and modified, the number of observations, and so on. The tablemeta data set contains the following variables:

- **CRDATE** — the date on which the data source was created.
- **ENGINE** — the version of SAS system that the data source was created in.
- **LIBNAME** — the name of the SAS library.

- **MEMLABEL** — the label of the data source.
- **MEMNAME** — the name of the data source.
- **MODATE** — the date on which the data source was last modified.
- **NCOLS** — the number of variables of the data source.
- **NOBS** — the number of observations of the data source.
- **ROLE** — the data role of the data source.
- **TYPEMEM** — the data member type, either DATA or VIEW.

*

Columnmeta File

The columnmeta file contains a row for each variable of the form created by the CONTENTS procedure. It contains information about variables in the data source. For example, the columnmeta file of the HMEQ data contains variable names, roles, measurement levels, and other attributes for each variable in the SAMPSIO.HMEQ data set. The columnmeta data set contains the following variables:

- **COMMENT** — contains additional information. SAS Enterprise Miner nodes might set the value of this field.
- **CREATOR** — identifies the SAS Enterprise Miner node that created the variable, if any. For example, probability variables that are created by the **Regression** node have creator Reg.
- **DISTRIBUTION** — the expected univariate distribution.
- **FAMILY** — identifies the general source of a variable such as demographic, financial, historic for record keeping. Such values can be useful in constructing the data, for example.
- **FORMAT** — specifies the format of a variable
- **FORMATTYPE** — one of the following: number, date, time, choice.
- **INDEX** — is the Boolean index indicator.
- **INDEXTYPE** — is the index type if the variable is indexed.
- **INFORMAT** — is the SAS informat for the variable.
- **LABEL** — is the variable label.
- **LENGTH** — is the variable length.
- **LEVEL** — is the measurement level of the variable.
- **LOWERLIMIT** — is the lower limit of variable values.
- **NAME** — is the variable name.
- **ORDER** — is the formatted value sorting order for class variables.
- **PRICE** — is the associated cost value of a variable.
- **REPORT** — indicates whether a variable should be automatically included in reports such as the predictive model assessment decile and percentile tables.
- **ROLE** — is the variable role.
- **TYPE** — specifies the variable type.
- **UPPERLIMIT** — is the upper limit of variable values.

Target Profile File

The target profile file is SAS data set and the rows correspond to the target variables that are defined in the data source. The target profile file for a data source with two targets contains a row for each of the targets. The target profile file contains the following variables:

- **ProfileID** — is the identifier of a target profile.
- **ProfileName** — is the name of a target profile.
- **Target** — is the target variable name.
- **Level** — is the measurement level of the target.
- **Use** — indicates whether the target profile is used.

Decision Matrix Files

The decision matrix file is a SAS data set that stores decision matrix values and prior probabilities. Each row in the decision matrix data set corresponds to a decision. The decision matrix data set contains the following variables:

-
- **<target_name>** — The name of this variable is the same as the target variable. The values are the decisions for the target variable.
- **COUNT** — is the number of observations in the data set that have the decision value.
- **DATAPRIOR** — is the prior probability value that is proportional to the data.
- **TRAINPRIOR** — is the prior probability value that is proportional to the data.
- **DECPRIOR** — is the adjusted probability value that you specified.
- Other variables. One variable is generated for each decision. The variable value is the decision matrix value.

Decision Variables Files

The decision variables file stores information about the target and other new variables that can be found in the scored data set. The decision variables file contains the following variables:

- **_TYPE_** — is the variable type.
- **VARIABLE** — is the name of the variable in the scored data set.
- **LABEL** — is the variable label.
- **LEVEL** — is the type of the decision matrix, either profit or loss. It also displays the measurement level of the target.
- **EVENT** — is the event level of the target.
- **ORDER** — is the order of the target values, either descending or ascending.
- **FORMAT** — is the format of the variable.
- **TYPE** — is the type of the variable. C represents a character variable. N represents a numeric variable.
- **COST** — is the cost variable or the value of constant cost, if defined.
- **USE** — indicates whether to use the decision matrix in modeling.

Properties File

The properties file lists properties of the data source as they were edited in the wizard and displayed in the properties window. The form of the file is one line per property, and each line is written as a name:value pair.

```
#!EMDataSource%5
    CreateDate: 1336903904.6
   ModifyDate: 1336903904.7
   CreatedBy: user2
   ModifiedBy: user2
```

Notes File

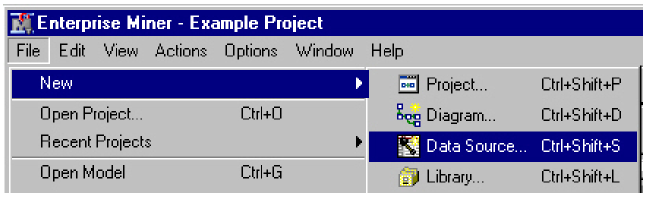
The notes file contains the notes that you enter for record keeping.

Creating a Data Source Using the Data Source Wizard

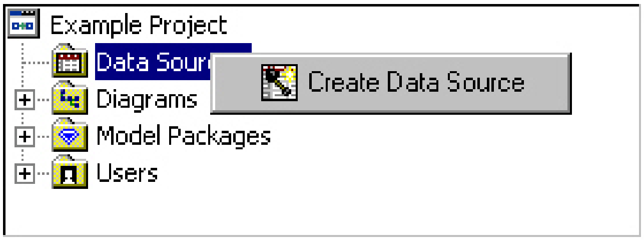
Follow these steps to use the **Data Source** wizard to create a data source.

1. Use one of the following methods to open the wizard:

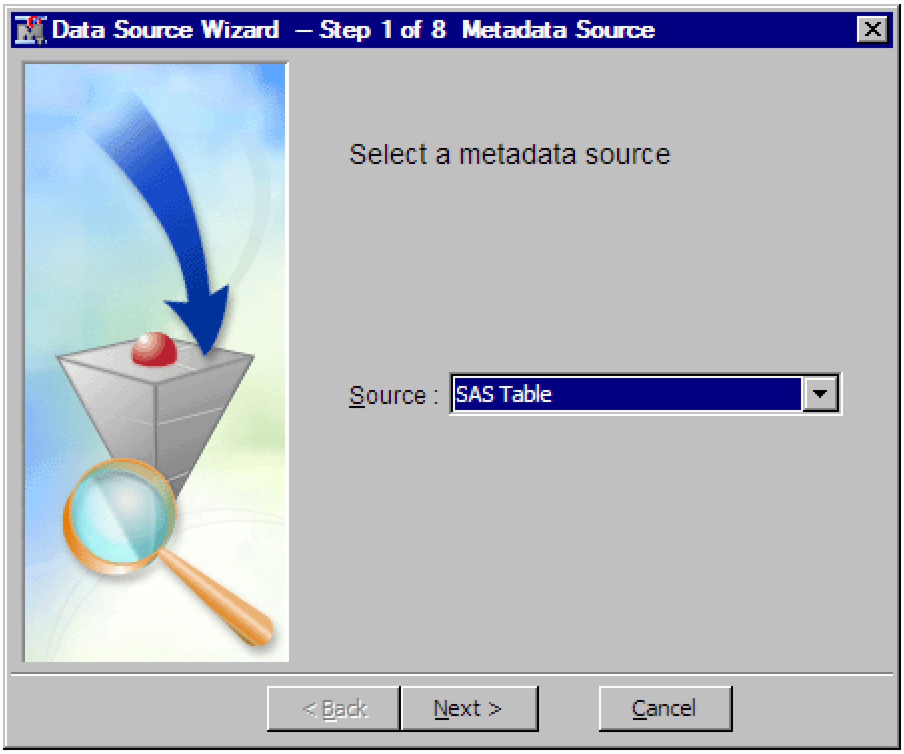
   - Select **File** ⇨ **New** ⇨ **Data Source** from the SAS Enterprise Miner main menu.



   - Right-click the Data Sources folder in the Project Panel and select **Create Data Source**.



2. In the Data Source Wizard — Metadata Source window, select the source of data that you want to access and click **Next**.



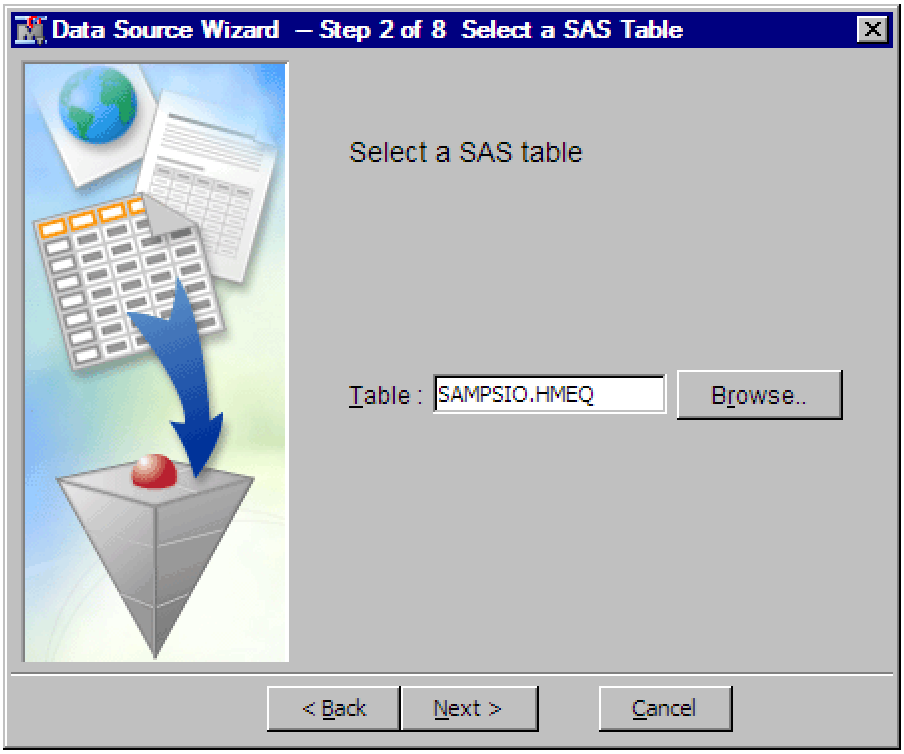
You can select from the following sources:

- **SAS Table** — This is an SAS library engine format table. The library must be created before the table is selected.

Note: If you have SAS tables only, you do not have to pre-assign libraries. The libraries are automatically available as a **Metadata Repository** from the Create Data Source wizard. If you have database or RDBM libraries (such as Oracle), you must pre-assign RDBMS libraries, and they are then available as a **SAS Table** in the Create Data Source wizard. If you have both SAS and RDBMS tables, you must choose the **Library is pre-assigned** option for both types of tables. With this option, all of the tables are made available via **SAS Table** in the Create Data Source wizard.

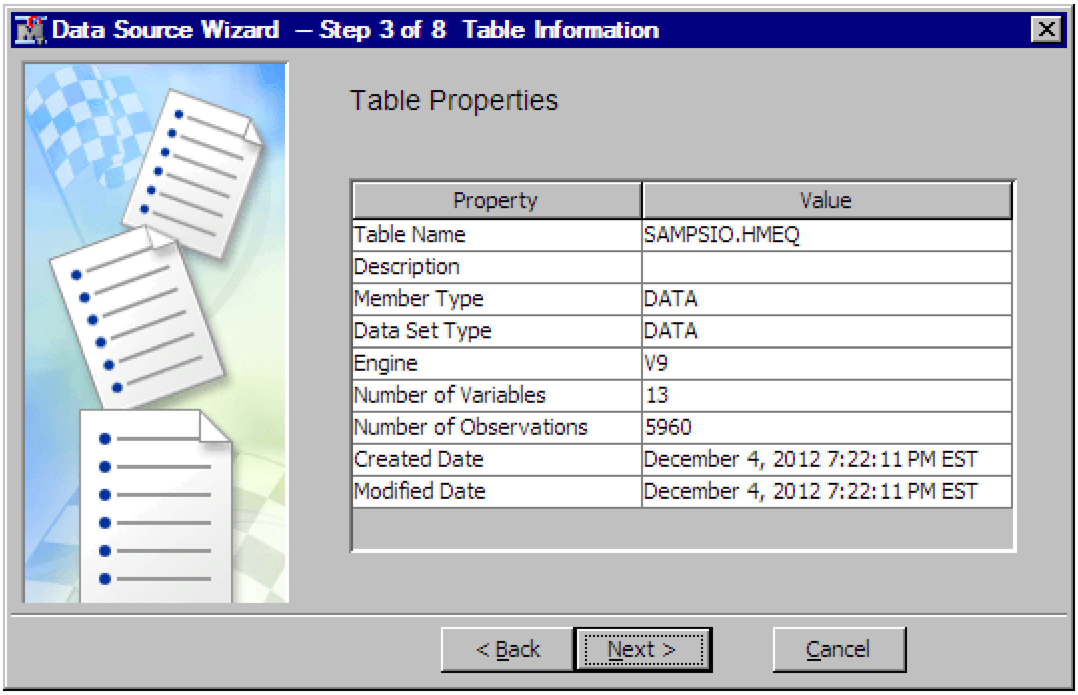
Password-protected tables are not valid for creating data sources.

3. If you select **SAS Table** as the source, the Data Source Wizard — Select a SAS Table window appears. Select a SAS data table by entering the data set name or clicking **Browse** to select from a list. Then, click **Next**.



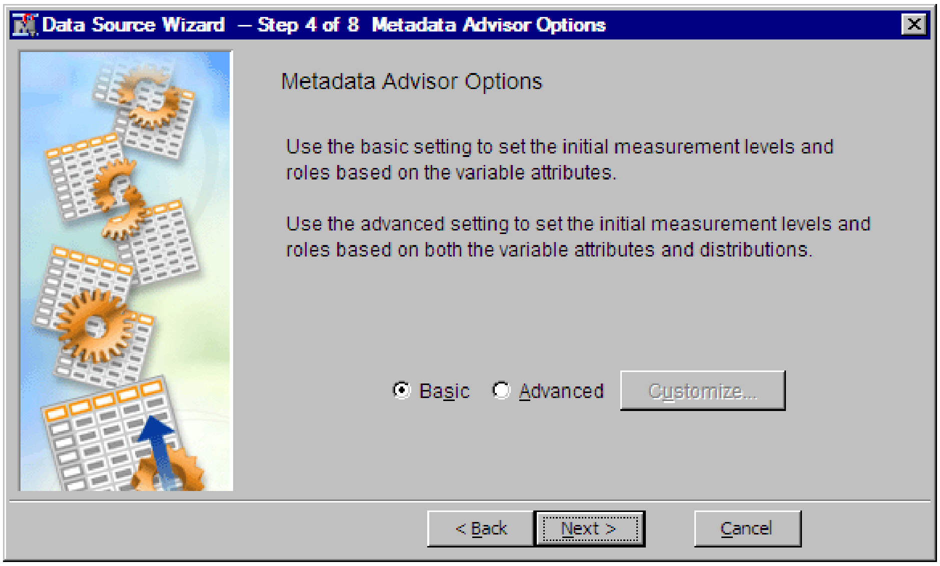
If you select **Metadata Repository** as the source, the Data Source Wizard — Import Metadata from Metadata Repository window appears. Click **Browse** to select an item from the list of registered tables.

4. In the Data Source Wizard – Table Information window, you can view the table attributes including the table name, the number of variables, and the number of observations. Then, click **Next**.



Property	Value
Table Name	SAMPSIO.HMEQ
Description	
Member Type	DATA
Data Set Type	DATA
Engine	V9
Number of Variables	13
Number of Observations	5960
Created Date	December 4, 2012 7:22:11 PM EST
Modified Date	December 4, 2012 7:22:11 PM EST

5. The Data Source Wizard – Metadata Advisor Options window enables you to select the type of advisor option used to create column attributes. Column attributes are also known as metadata.



Metadata Advisor Options

Use the basic setting to set the initial measurement levels and roles based on the variable attributes.

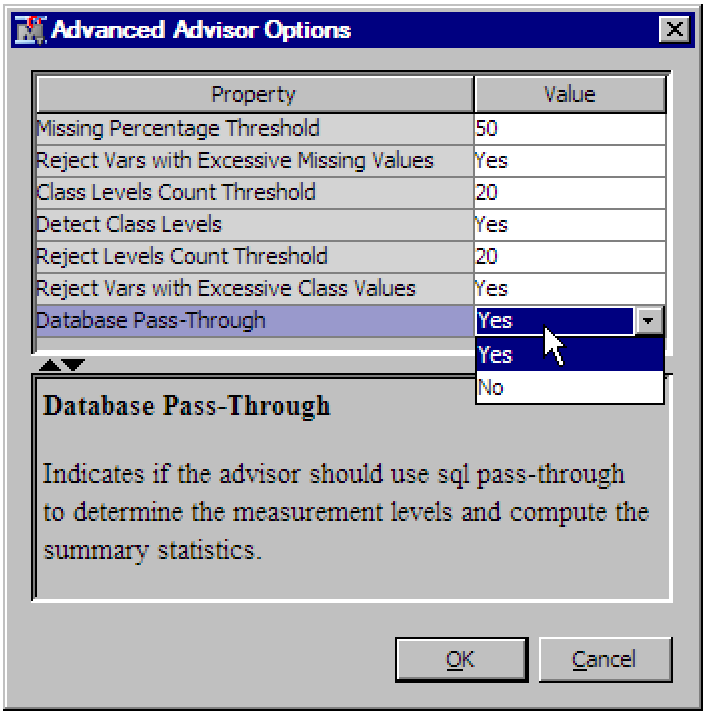
Use the advanced setting to set the initial measurement levels and roles based on both the variable attributes and distributions.

● Basic ○ Advanced

Two options are available.

- **Basic** — Use the **Basic** option when you already know the variable roles and measurement levels. The initial role and level are based on the variable type and format values.
- **Advanced** — Use the **Advanced** option when you want SAS Enterprise Miner to automatically set the variable roles and measurement levels. Automatic initial roles and level values are based on the variable type, the variable format, and the number of distinct values contained in the variable. If this is selected, you will be able to display statistics in the Data Source Wizard – Column Metadata step. The Database pass-through option enables you to compute summary statistics in the database rather than in SAS Enterprise Miner.

If you select **Basic**, click **Next** to proceed. If you select **Advanced**, click **Customize** to view details of the options:



Property	Value
Missing Percentage Threshold	50
Reject Vars with Excessive Missing Values	Yes
Class Levels Count Threshold	20
Detect Class Levels	Yes
Reject Levels Count Threshold	20
Reject Vars with Excessive Class Values	Yes
Database Pass-Through	Yes

Database Pass-Through

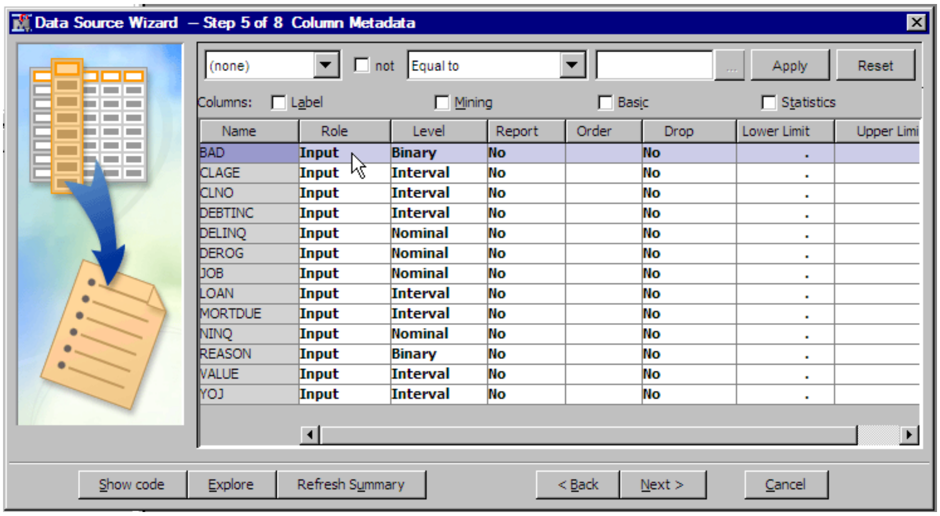
Indicates if the advisor should use sql pass-through to determine the measurement levels and compute the summary statistics.

To customize the advanced options, enter a value or use the drop-down list in the Value column to change it. Click **OK** in the Advanced Advisor Options window to save your changes. Then, select **Next**.

The following options are available in the Advanced Options window:

- **Detect Class Levels** — specifies whether the number of class levels is determined for each variable.
- **Class Levels Count Threshold** — specifies the maximum number of class levels for each variable. When the **Detect Class Levels** property is set to **Yes**, if there are more class levels than the value specified here, the variable is considered an interval variable. Valid values are positive integers greater than or equal to 2.
- **Reject Vars with Excessive Missing Values** — specifies whether variables with a large percentage of missing values should be rejected.
- **Missing Percentage Threshold** — specifies the maximum percentage of missing values allowed for each variable. When the **Reject Vars with Excessive Missing Values** property is set to **Yes**, a variable is rejected when the percentage of missing values is greater than the value specified here. Valid values are integers between 0 and 100, inclusive.
- **Reject Vars with Excessive Class Values** — specifies whether variables with a large number of class variable levels should be rejected.
- **Reject Levels Count Threshold** — specifies the maximum number of class variable levels allowed for each variable. When the **Reject Vars with Excessive Class Values** property is **Yes**, class variables with more levels than the value specified here are rejected.
- **Identify Empty Columns** — specifies whether empty columns are assigned the value of MISSING.
- **Database Pass-Through** — specifies if the data source advisor should use in-database processing to calculate summary statistics and determine measurement levels.

6. The Data Source Wizard – Column Metadata window displays the default attributes that are defined for each variable.


Data Source Wizard – Step 5 of 8 Column Metadata
(none)
□ not
Equal to
Apply
Reset
Columns: □ Label □ Mining □ Basic □ Statistics
Show code
Explore
Refresh Summary
< Back
Next >
Cancel

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limi
BAD	Input	Binary	No		No	.	
CLAGE	Input	Interval	No		No	.	
CLNO	Input	Interval	No		No	.	
DEBTINC	Input	Interval	No		No	.	
DELINQ	Input	Nominal	No		No	.	
DEROG	Input	Nominal	No		No	.	
JOB	Input	Nominal	No		No	.	
LOAN	Input	Interval	No		No	.	
MORTDUE	Input	Interval	No		No	.	
NINQ	Input	Nominal	No		No	.	
REASON	Input	Binary	No		No	.	
VALUE	Input	Interval	No		No	.	
YOJ	Input	Interval	No		No	.	

On top of the table is a configurable WHERE clause filter that is helpful when configuring a long list of variables. To view extra columns, select the **Label**, **Mining**, **Basic**, or **Statistics** check boxes.

The following columns are displayed if **Mining** is checked:

- Creator
- Comment
- Format Type

The following columns are displayed if **Basic** is checked:

- Type
- Format
- Informat
- Length

To enable the calculation of summary statistics, check **Statistics**.

The following new columns are displayed:

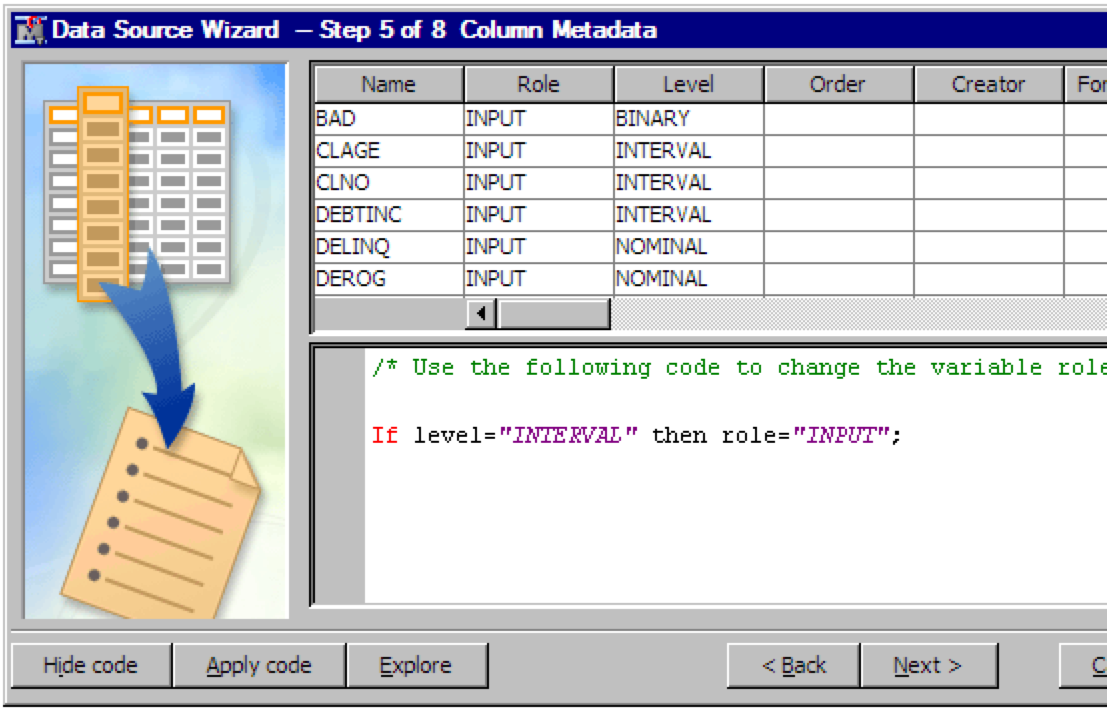
- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percentage of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for interval variables only. For class variables, . is displayed.
- **Maximum** — displays the maximum values for interval variables only. For class variables, . is displayed.
- **Mean** — displays the mean values for interval variables only. For class variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for interval variables only. For class variables, . is displayed.
- **Skewness** — displays the skewness for interval variables only. For class variables, . is displayed.
- **Kurtosis** — displays the kurtosis values for interval variables only. For class variables, . is displayed.

Select a variable and click **Explore** to view the variable distribution. The following is a list of the attributes that you can change. To change an attribute, select the corresponding field of a variable and choose an item from the drop-down list.

- **Name** — is the name of the variable.
- **Role** — is the model role of the variable.
- **Level** — is the measurement level of the variable.
- **Report** — indicates whether a variable should be automatically included in reports such as the predictive model assessment decile and percentile tables.
- **Order** — is the formatted value sorting order for class variables.
- **Drop** — drops the variable.
- **Lower Limit** — is the lower limit of the variable.
- **Upper Limit** — is the upper limit of the variable.
- **Creator** — identifies the SAS Enterprise Miner node that created the variable, if any. For example, probability variables that are created by the **Regression** node have creator Reg.

- **Comment** — contains additional information. SAS Enterprise Miner nodes might set the value of this field.
- **Format Type** — is the format type of the variable.

Alternatively, you can write SAS code to automate the assignment of column attributes, particularly when you have a large number of variables. Clicking **Show Code** enables the Program Editor.



Data Source Wizard — Step 5 of 8 Column Metadata

Name	Role	Level	Order	Creator	For
BAD	INPUT	BINARY			
CLAGE	INPUT	INTERVAL			
CLNO	INPUT	INTERVAL			
DEBTINC	INPUT	INTERVAL			
DELINQ	INPUT	NOMINAL			
DEROG	INPUT	NOMINAL			

```
/* Use the following code to change the variable role

If level="INTERVAL" then role="INPUT";
```

Note: Make sure the measurement levels and model roles in the SAS code are in UPPER CASE.

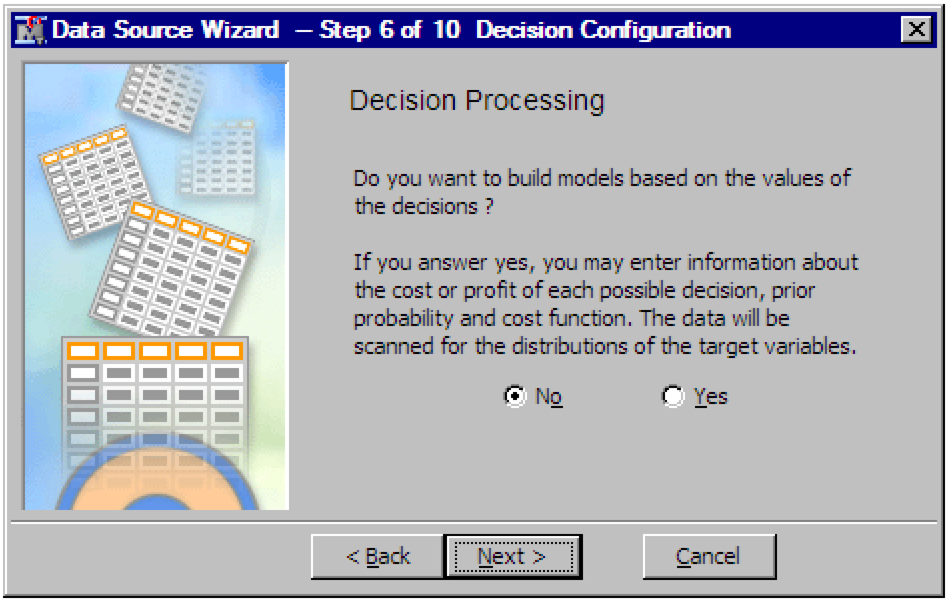
To submit the code, click **Apply Code**.

When you are using the Program Editor, variable names and attributes are displayed as raw values not translated for localization. This is because the program code runs and modifies the raw values. Once you have finished editing and submitting code and reviewing the results, select the Hide Code to view the translated display values.

Most supervised data mining models use a target variable. If you have not used code or other methods to specify a variable with the role of Target, do so now by selecting the variable in the table and choosing **Target** from the list. Use this method to specify other variable roles as well.

Click **Next**.

7. In the Data Source Wizard — Decision Configuration window, you choose whether to define a target profile that produces optimal decisions from a model. Notice that because a binary target variable was chosen, two more configuration windows are added to the Data Source Wizard. As a result, the numbering of the Data Source Wizard windows has changed. If you used Basic Metadata Advisor option, or if you did not configure a target variable role, decision processing is not enabled, and this window will not appear in your Data Source Wizard. If you used the Advanced Metadata Advisor role, and if you specified a binary target variable for your data source, the window depicted below appears, numbered Step 6 of 10.



Data Source Wizard — Step 6 of 10 Decision Configuration

Decision Processing

Do you want to build models based on the values of the decisions ?

If you answer yes, you may enter information about the cost or profit of each possible decision, prior probability and cost function. The data will be scanned for the distributions of the target variables.

● No ○ Yes

- If you select **Yes** and click **Next**, the Data Source Wizard – Decision Configuration window appears.
- If you select **No** and click **Next**, the Data Source Wizard – Data Source Attributes window appears.

8. The Data Source Wizard — Decision Configuration window enables you to set the decision values in the Target Profile. For detailed information about the Target Profiler, see "Enterprise Miner Target Profiler" on page 207 in the SAS Enterprise Miner Reference Help. After you finish configuration of the decision configuration, click **Next**.

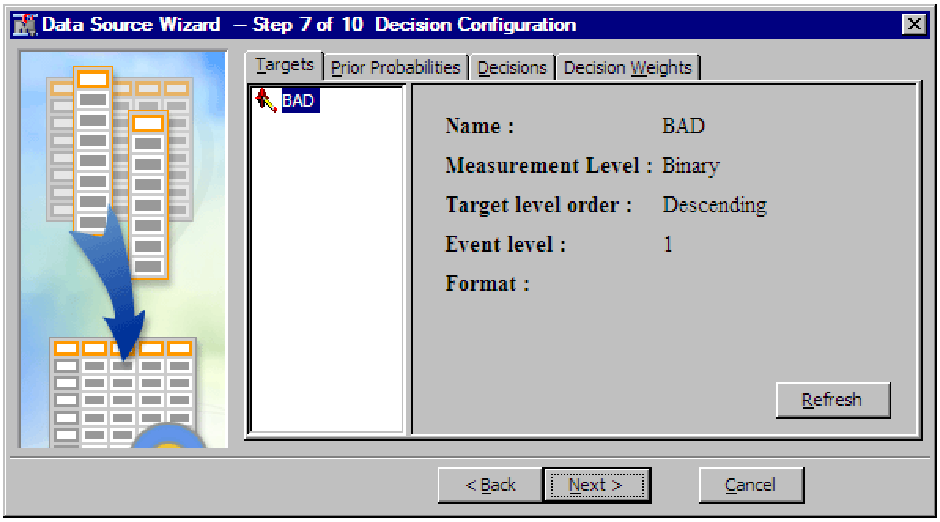
   Note: For interval targets, decision configuration is not supported for this release.

   The Data Source Wizard — Decision Configuration window has the following tabs:

   - **Targets Tab**

     The left panel of the **Targets** tab displays the target variables that are defined in the data source.

     For categorical targets, the right panel of the **Targets** tab lists the variable name, measurement level, order, and the event level.



Data Source Wizard — Step 7 of 10 Decision Configuration

Name : BAD

Measurement Level : Binary

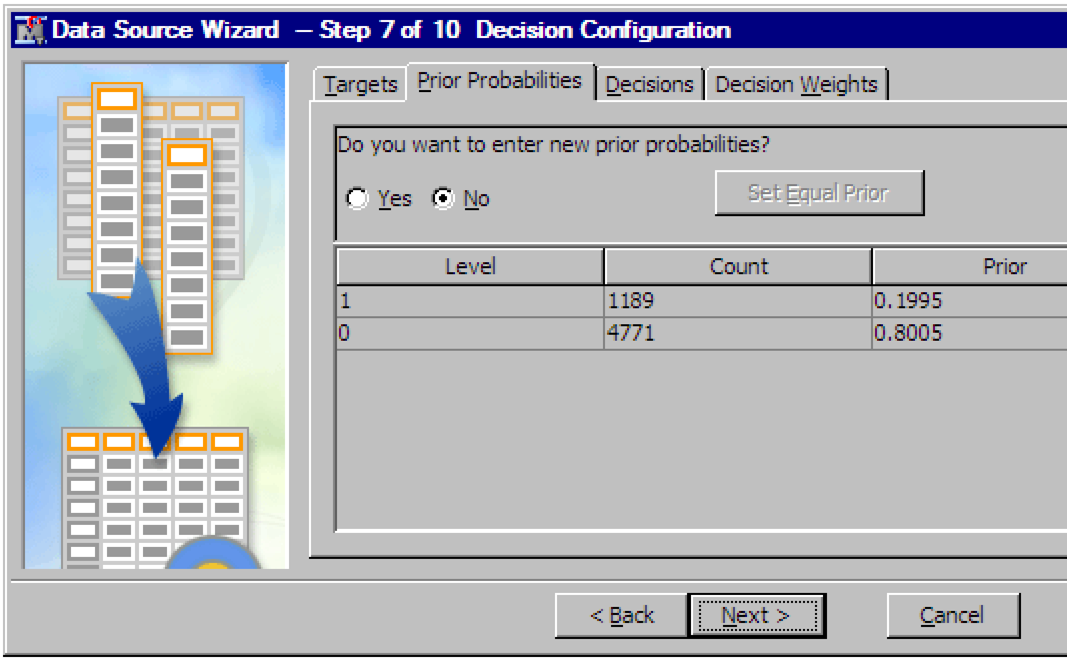
Target level order : Descending

Event level : 1

Format :

- **Prior Probabilities Tab**

  The **Prior Probabilities** tab enables you to specify prior probability values to implement prior class probabilities for categorical targets.



Do you want to enter new prior probabilities?

○ Yes ● No

Level	Count	Prior
1	1189	0.1995
0	4771	0.8005

  The **Prior Probabilities** tab has the following columns:

  Level — displays the levels for target variables.

  Count — displays the number of training observations for each target level.

  Prior — displays the percentage of training observations for each target level.

  Adjusted Prior — displays the prior probability values that you specify. The values that you specify in this column must sum to 1. This column is displayed only if you select **Yes**.

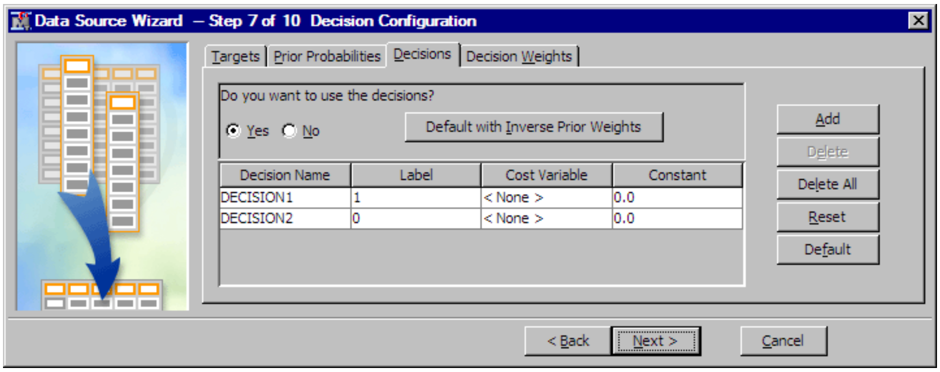
  To use your prior probabilities, select **Yes** and enter your adjusted probabilities.

  To set prior probabilities equal to each other, click **Set Equal Prior**.

- **Decisions Tab**

  The **Decisions** tab enables you to specify decisions, a numeric constant cost or a cost variable for each decision.

  The following display shows an example of the **Decisions** tab. To specify a cost variable, select the corresponding field for a decision and select the variable from the drop-down list. In order to use a variable as the cost variable, the variable role must be set to **Cost** in the Data Source Wizard — Columns Meta window. To specify a constant cost, select **_Constant_** from the drop-down list and enter a value in the corresponding Constant field.



Do you want to use the decisions?

● Yes ○ No

Decision Name	Label	Cost Variable	Constant
DECISION1	1	< None >	0.0
DECISION2	0	< None >	0.0

To add a decision, click **Add**.

To delete a decision, select a decision, and click **Delete**.

To delete all decisions, click **Delete All**.

To reset all the settings in the **Decisions** and **Decision Weights** tabs back to the values that you had when you opened the decision editor, click **Reset**.

To use the default values for all the settings in the **Decisions** and **Decision Weights** tabs, click **Default**. The default matrix contains a decision column for each corresponding target level. No cost variable or information is used by default. See "Decision Weights Tab" on page 212 in the Target Profile documentation for more information about default matrices.

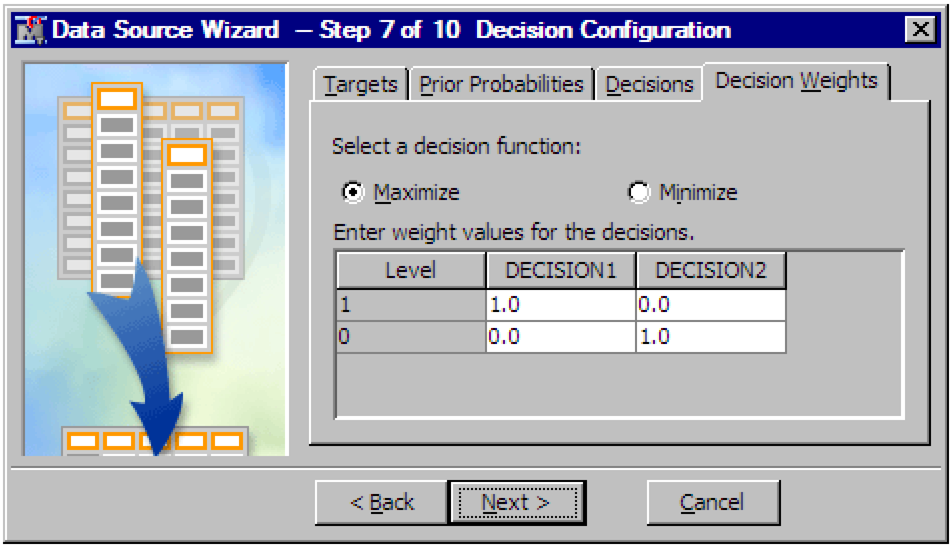
To set the inverse priors as decision weights, click **Default with Inverse Priors**. The **Decision Weights** tab reflects the decision weights that are calculated as the inverse of prior probabilities that you set in the **Prior Probabilities** tab.

- **Decision Weights Tab**

  The **Decision Weights** tab enables you to specify the values in the decision matrix. The decision matrix contains columns that correspond to each decision, and rows that correspond to target values. The values of the decision variables represent target-specific consequence, which can be PROFIT, LOSS, or REVENUE. Based on the values of the decision variables, select either **Maximize** or **Minimize** to specify the assessment objective for the matrix. Select **Maximize** if the values in the decision matrix represent profit or revenue. Select **Minimize** if the values in the decision matrix represent loss. By default, **Maximize** is selected.

  The target levels that are displayed in the default decision matrix depend on the order of target levels. By default, it is set to descending for categorical targets.

  In the following example, there are two decisions because the target BAD contains values of 1 and 0. To change the values in the decision matrix, select a field in the matrix and enter a number.



Select a decision function:

● Maximize ○ Minimize

Enter weight values for the decisions.

Level	DECISION1	DECISION2
1	1.0	0.0
0	0.0	1.0

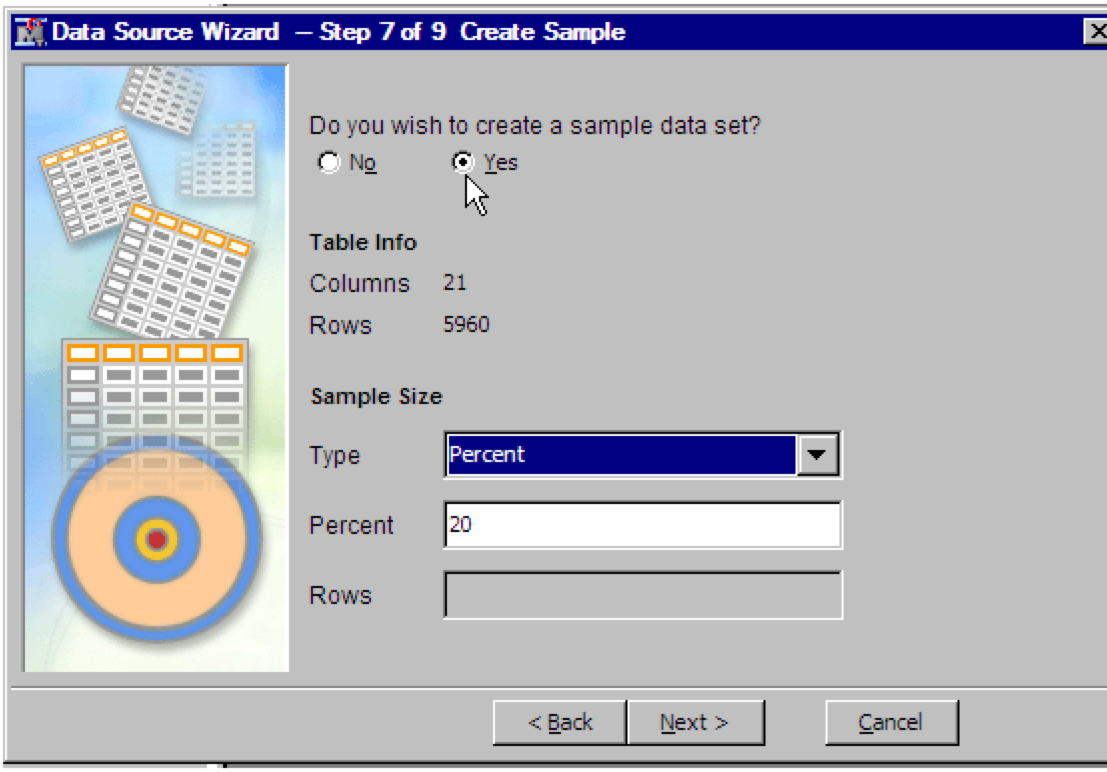
Select **Next**.

9. In the Data Source Wizard – Create Sample window, you specify information about the sample data set that Enterprise Miner automatically creates for data sources. If you did not create a target variable, this window will be labeled Data Source Wizard, Step 6 of 8. If you created a target variable, but did not create a decision matrix, this window will be labeled Data Source Wizard, Step 7 of 9. If you created a target variable and a decision matrix, then this window will be labeled Data Source Wizard, step 8 of 10. Regardless of the path that you took to this window, there are three remaining windows in the Data Source Wizard.

   If you want to control the size of the sample data set that Enterprise Miner automatically creates for your data source, select **Yes**, then specify either the percentage of the data to sample, or the number of rows to sample. If you leave **No** selected, the automatic data set is created with a default size of 100000 rows.

   If one or more class targets are defined in the data source, and/or one or more segment variables are defined in the data source, then the automatically created data source sample will be stratified by those variables, up to a limit of two stratification variables.

   Sampling is performed in the database where the data is stored, unless the Database pass-through option was set to **No** in the Advanced Advisor. Click **Next**.



Data Source Wizard – Step 7 of 9 Create Sample

Do you wish to create a sample data set?

○ No ● Yes

Table Info

Columns 21

Rows 5960

Sample Size

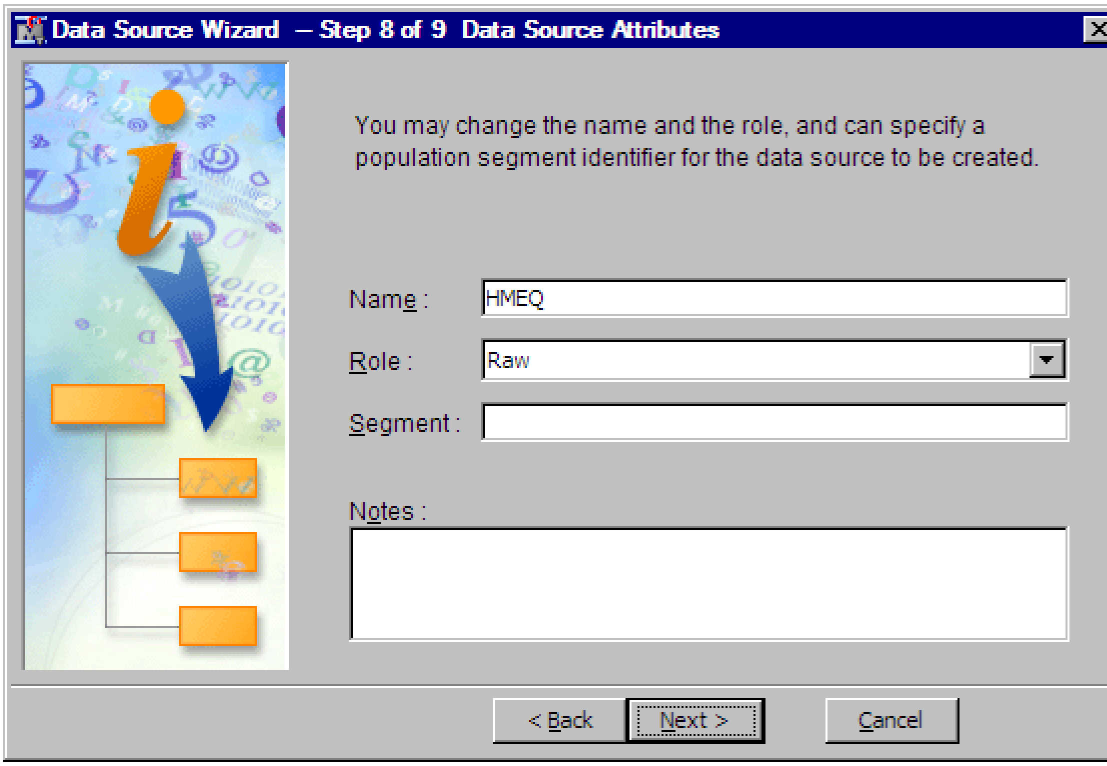
Type: Percent

Percent: 20

Rows:

Note: When High-Performance nodes are used, you can control the size of the automatic sample that SAS Enterprise Miner creates in Step 7 of the Data Source Wizard. However, SAS Enterprise Miner will always create the automatic sample, even if it is not specified. If the Data Source sample is specified in Step 7 of the Data Source Wizard, then the sample will be the size that you specify in the wizard. Otherwise, Enterprise Miner will create a 100,000 row sample of the data source. If one or more class targets are defined, and/or one or more segment variables are defined, then the sample will be stratified by those variables, up to a limit of two stratification variables.

10. In the Data Source Wizard – Data Source Attributes window, you can specify the name, the role, or a population segment identifier for the data source. Click **Next**.



Data Source Wizard – Step 8 of 9 Data Source Attributes

You may change the name and the role, and can specify a population segment identifier for the data source to be created.

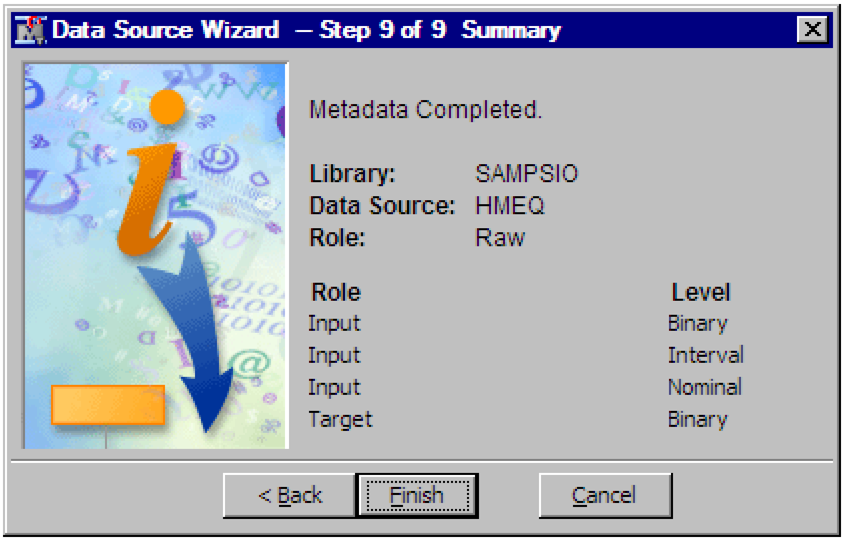
Name : HMEQ

Role : Raw

Segment :

Notes :

11. In the Data Source Wizard – Summary window, click **Finish**. The data source appears in the Project Panel of the SAS Enterprise Miner main window.



Data Source Wizard – Step 9 of 9 Summary

Metadata Completed.

Library: SAMPSIO
Data Source: HMEQ
Role: Raw

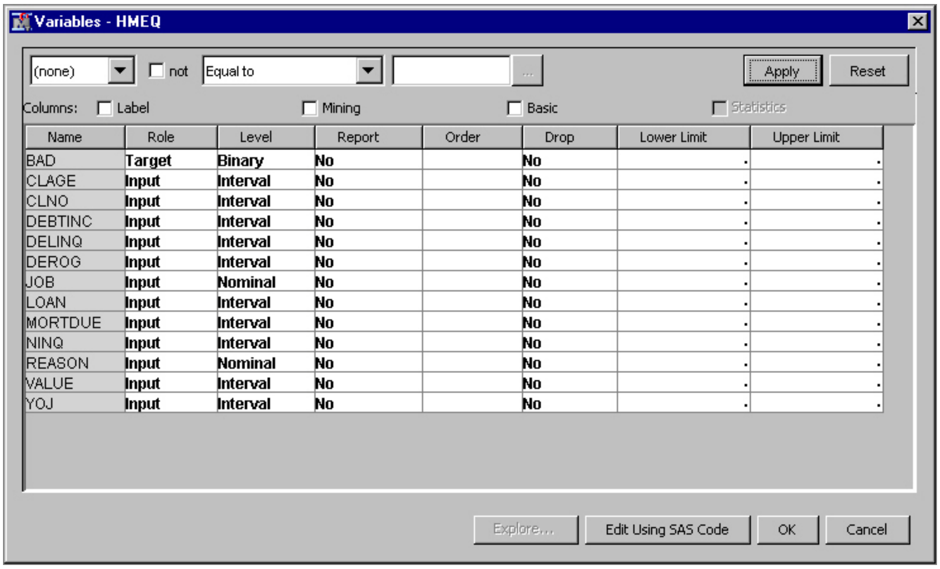
Role	Level
Input	Binary
Input	Interval
Input	Nominal
Target	Binary

Manipulating a Data Source

After a data source has been created, it is displayed in the Project Panel of the SAS Enterprise Miner main window. The following pop-up menu items are available when you right-click a data source:

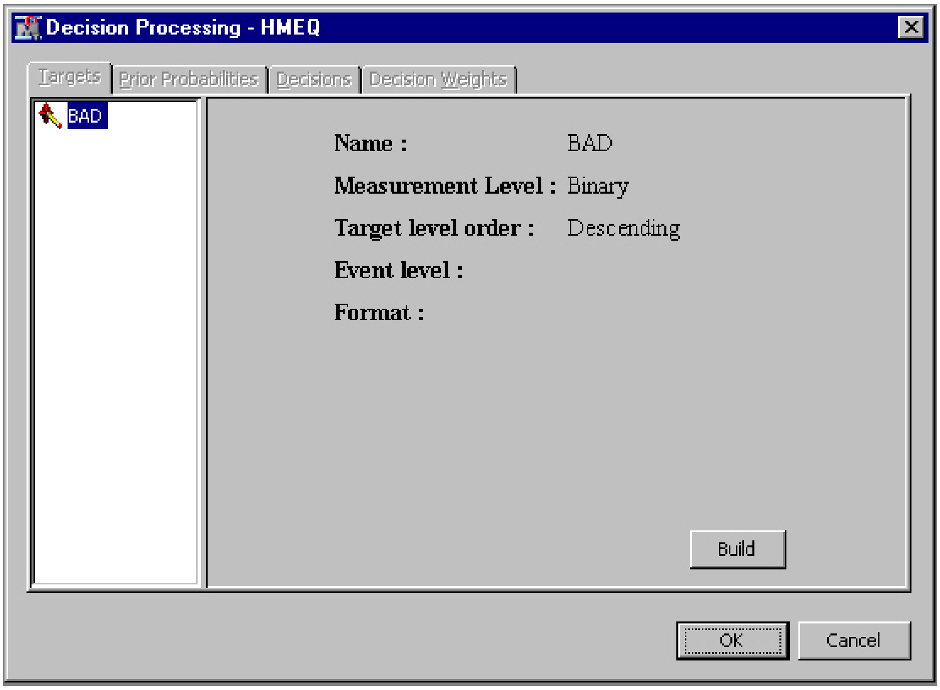
- **Rename** — opens the Input window that you can use to enter a new data source name.
- **Duplicate** — creates a copy of the selected data source.

- **Delete** — deletes the selected data source.
- **Edit** Variables — opens the Variables window that enables you to modify the attributes of variables.



Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
BAD	Target	Binary	No		No	.	.
CLAGE	Input	Interval	No		No	.	.
CLNO	Input	Interval	No		No	.	.
DEBTINC	Input	Interval	No		No	.	.
DELINQ	Input	Interval	No		No	.	.
DEROG	Input	Interval	No		No	.	.
JOB	Input	Nominal	No		No	.	.
LOAN	Input	Interval	No		No	.	.
MORTDUE	Input	Interval	No		No	.	.
NINQ	Input	Interval	No		No	.	.
REASON	Input	Nominal	No		No	.	.
VALUE	Input	Interval	No		No	.	.
YOJ	Input	Interval	No		No	.	.

- **Refresh Metadata** — refreshes the metadata.
- **Edit Decisions** — opens a Decision Processing window.



Name : BAD

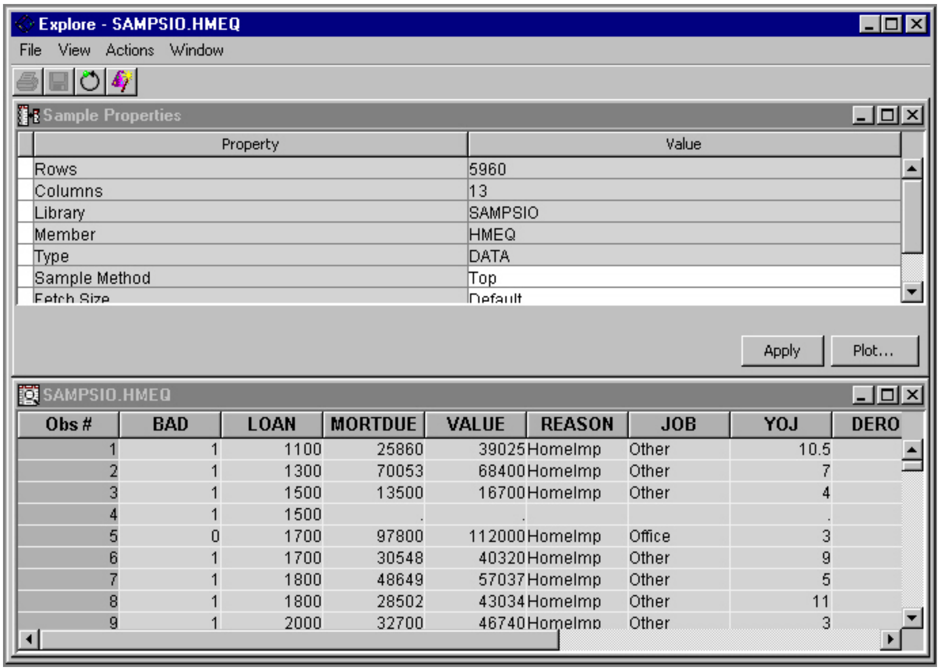
Measurement Level : Binary

Target level order : Descending

Event level :

Format :

- Explore — opens an Explore window.



Property	Value
Rows	5960
Columns	13
Library	SAMPSIO
Member	HMEQ
Type	DATA
Sample Method	Top
Fetch Size	Default

Obs #	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DERO
1	1	1100	25860	39025	HomeImp	Other	10.5	
2	1	1300	70053	68400	HomeImp	Other	7	
3	1	1500	13500	16700	HomeImp	Other	4	
4	1	1500	.	.			.	
5	0	1700	97800	112000	HomeImp	Office	3	
6	1	1700	30548	40320	HomeImp	Other	9	
7	1	1800	48649	57037	HomeImp	Other	5	
8	1	1800	28502	43034	HomeImp	Other	11	
9	1	2000	32700	46740	HomeImp	Other	3	

After a data source has been created, it can be used in any process flow diagram within the project.

You can use the data source that was defined in another project by copying and pasting the data source definition files from one project to the other. For example, suppose that the HMEQ data source is defined in Project_A, and you want to use the HMEQ data source in Project_B without re-creating it. First, locate the files for both Project_A and Project_B. Then you copy the data source definition files in the DataSources subdirectory of Project_A, and paste them to the DataSources subdirectory of Project_B.

For information about the data source definition files and how to identify these files for a data source, see Contents of a Data Source on page 289 .

Data Source Roles

A data source appears in the Project Panel of the SAS Enterprise Miner window. When a data source is selected, properties are displayed in the Properties Panel. See Data Source Properties for detailed information.

When you select a data source, use the Role property to change the role of a data source. The role of a data source determines how the data source is used throughout the process flow diagram. Select one of the following values:

- **Raw** (Default) — is used as raw input to a node.
- **Train** — is used to fit initial models.
- **Validate** — is used by default for model comparison, if available. The validate data source is also used for fine-tuning the model. The **Decision Tree** and **Neural Network** nodes have the capacity of overfitting the TRAIN data set. To prevent these nodes from overfitting the train data set, the validate data set is automatically used to retreat to a simpler fit than the fit based on the train data alone. The validate data set can also be used by the **Regression** node for fine-tuning stepwise regression models.
- **Test** — is used to obtain a final, unbiased estimate of the generalization error of the model.

- **Score** — is used to score a new data set that might not contain the target.
- **Transaction** — is used in the **Time Series** and **Path Analysis** nodes. Transaction data is time-stamped data that is collected over time at no particular interval.

  If you do not have training, validation, and test data sets, then you can create them with a successor **Data Partition** node.

Model Roles and Measurement Levels of Variables

To specify a model role or measurement level for a variable, select the ... button of the Variables property to open the variables table. Use the drop-down lists in the Role and Level columns to change the model role and measurement level of a variable.

Select one of the following values for model role:

- **Assessment** — is generated by a modeling node. An assessment variable is used for model comparisons.
- **Censor** — is a censor variable in survival analysis.
- **Classification** — is generated by a modeling node. A classification variable contains the prediction for a categorical target variable.
- **Cost** — is a variable that contains the decision costs for each observation in the input data set.
- **Cross ID** — is a variable that contains groups or levels for cross-sectional analysis. This is an optional variable used in the **Time Series** node.
- **Decision** — is a variable that contains information about the decision based on a predictive model.
- **Frequency** — is a variable that represents the frequency of occurrence for other values in each observation. Unlike some SAS procedures, the frequency variable can contain noninteger values and can be used for weighting observations.
- **ID**— an indicator variable for every observation in the data set. Observations with the same ID values form a transaction. The **Association** node requires an ID variable for association discovery. ID variables are normally excluded when analyzing the data with the other modeling nodes.
- **Input** — is a variable that is used to predict the target. It is the independent or explanatory variable.
- **Key** — is an indicator variable that is used instead of the **ID** role when building temporary tables on the grid. If no **Key** variable is found, you will experience performance degradation. As a result, High-Performance Data Mining diagrams created prior to SAS Enterprise Miner 12.3 that use an ID variable might run slower in SAS Enterprise Miner 12.3. It is highly recommended that you change the role of the ID variable to **Key**. Each observation of a **Key** variable must be unique.
- **Label** — is a variable that contains the text label.
- **Prediction** — is a variable that contains the predicted values of the target.
- **Referrer** — is used in the **Path Analysis** node. When you analyze web data that a user visits site B from site A, site A is the referrer in this case.
- **Rejected** — a variable that is excluded from the analysis in the process flow diagram. If you know a variable that is not important in the analysis, set the model role of the variable to Rejected.

- **Residual** — a variable that contains the residuals.
- **Segment** — is a variable that identifies segment. The **Clustering** node creates a segment variable. The **Text Miner** node also generates a segment variable is clustering analysis is performed.
- **Sequence** — is a variable that represents the time span from observation to observation. The **Association** node requires a sequence variable for sequence discovery. The sequence (time stamp) variable must be recorded on the same scale.
- **Target** — is a variable whose value is known in some currently available data, but will be unknown in some future or fresh data. It is the dependent or response variable.
- **Text** — is a variable that contains the text of documents or the path to the documents. The **Text Miner** node uses text variables.
- **Text Location** — is a variable that is used by the **Text Miner** node. This variable indicates the complete path to the HTML files that is created by %tmfilter. This variable is not created if the **destdir** option is not used in the %tmfilter statement.
- **Time ID** — is a variable that contains a time identifier, usually in SAS date, time, or datetime format. The **Time Series** node requires a timeID variable.
- **Treatment** — is a variable that the **NetLift** node uses to identify a treatment..
- **Web Address** — is a variable that is used by the **Text Miner** node. This variable indicates the complete path to the original file that %tmfilter processes.

Select one of the following measurement levels:

- **Binary** — contains two discrete values (for example, PURCHASE: Yes, No).
- **Interval** — contains values that vary across a continuous range (for example, TEMP: 0, 32, 34, 36, 50, 56, 80, ..., 102).
- **Nominal** — contains a discrete set of values that do not have a logical ordering (for example, PARTY: Democrat, Republican, other).
- **Ordinal** — contains a discrete set of values that do have a logical ordering (for example, GRADE: A, B, C, D, F).
- **Unary** — contains one discrete value.

Using a Data Source in the Graphic User Interface (GUI)

To use a data source in your process flow diagram, select the data source in the Project Panel and drag it to the Diagram Workspace.

Using a Data Source in the Batch Processing Interface

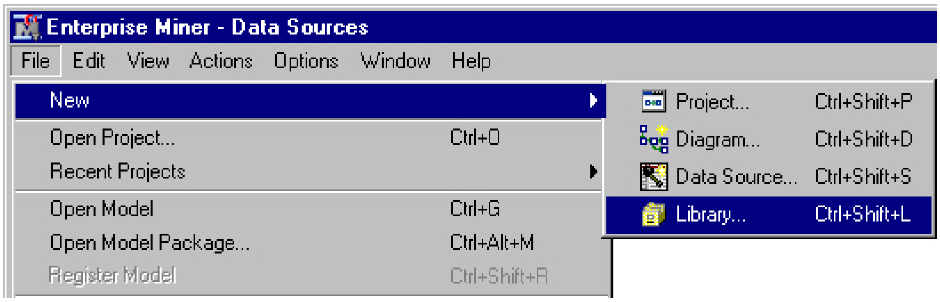
You can create and run SAS Enterprise Miner batch code using the SAS Enterprise Miner 12.3 batch processing code. When you create a SAS Enterprise Miner process using batch processing, you can enter the name of a specific data source within the project as a property of a data source node. See the SAS Enterprise Miner 12.3 Batch Processing documentation for detailed information about using the batch interface to submit SAS Enterprise Miner projects.

Creating a LIBNAME Using the Library Wizard

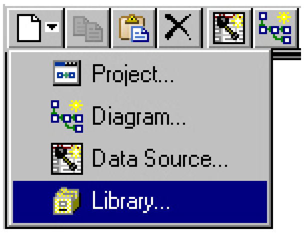
The SAS Enterprise Miner **Library** wizard enables you to select directories in the scope of the SAS session and create simple library definitions.

For more complex library definitions, use the SAS Enterprise Miner project start code or the SAS Metadata LIBNAME Engine that is accessed through SAS Management Console. The Library wizard has a Browse Directories action that browses directories on the SAS server, and not the SAS client.

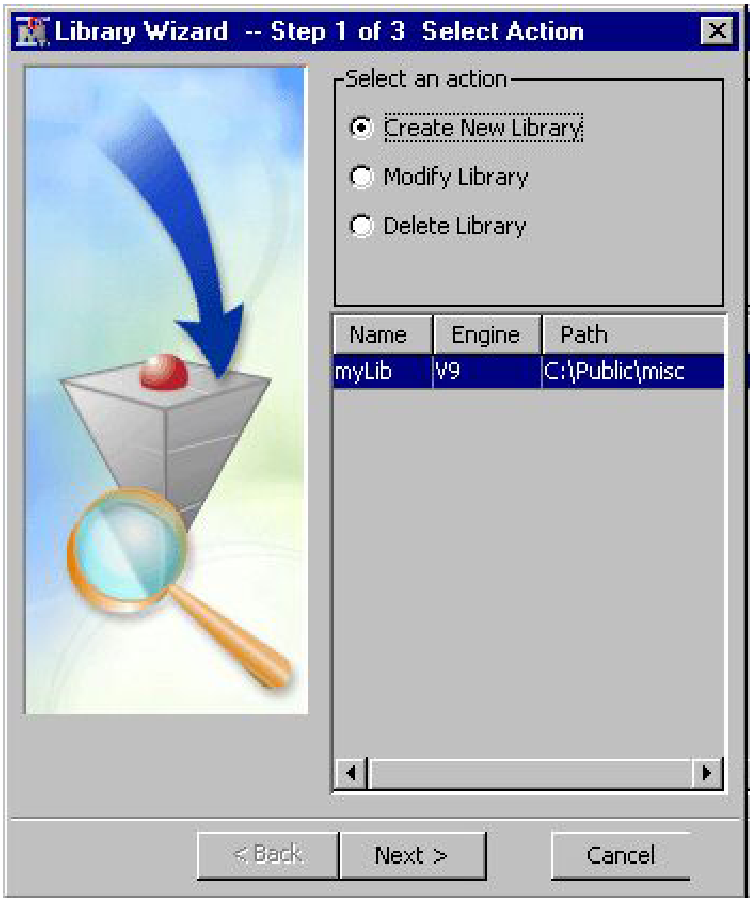
1. Use one of the following methods to open the wizard.
   - Select **File** ⇨ **New** ⇨ **Library** from the SAS Enterprise Miner window main menu.



   - Select from the SAS Enterprise Miner tools palette.



2. In the Library Wizard Select Action window, select **Create New Library** and click **Next**. You can modify or delete an existing saved LIBNAME definition by selecting this set from the table and selecting **Modify Library** or **Delete Library**.

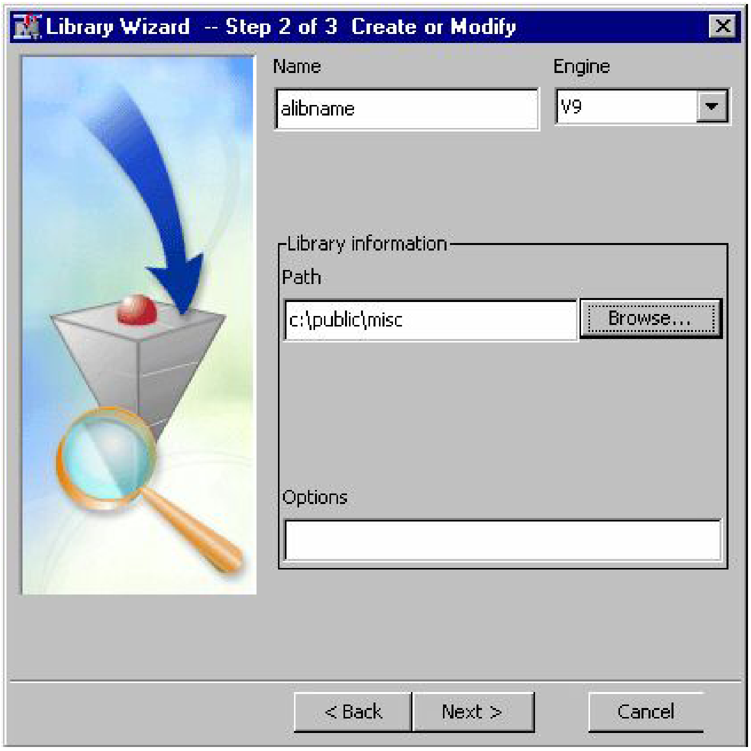

Library Wizard -- Step 1 of 3 Select Action

Select an action

● Create New Library
○ Modify Library
○ Delete Library

Name	Engine	Path
myLib	V9	C:\Public\misc

3. In the Library Wizard Create or Modify window, specify the library name, engine (V6, V8, V9, BASE, SPD Engine, or Excel), path, and options. Click **Next**.


Library Wizard -- Step 2 of 3 Create or Modify

Name: alibname

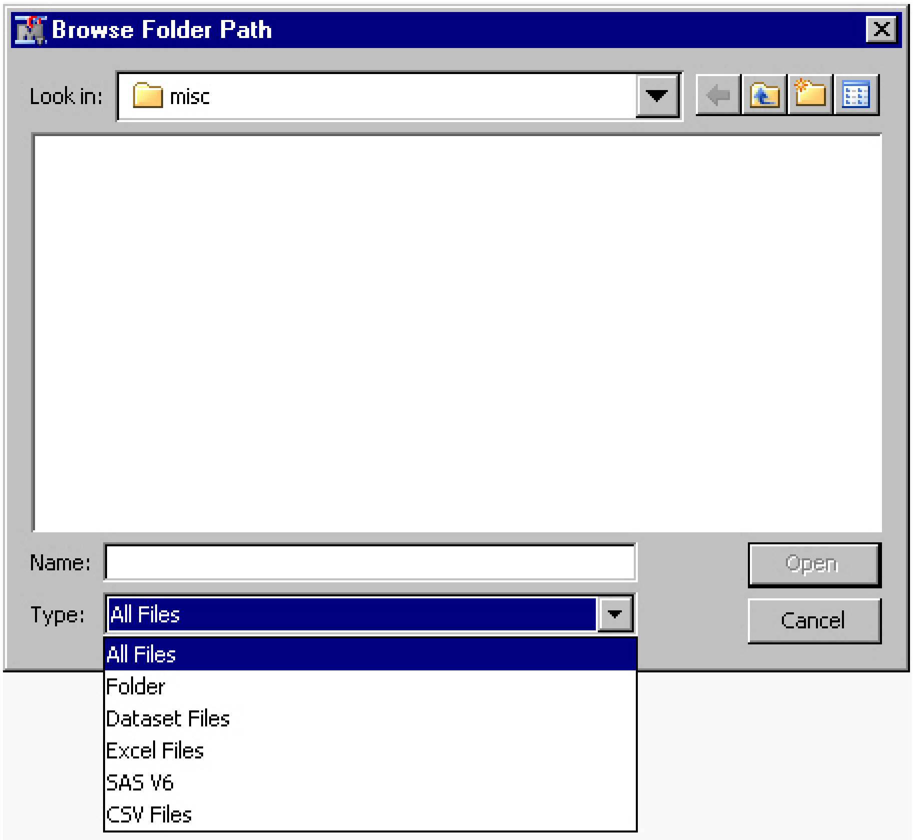
Engine: V9

Library information

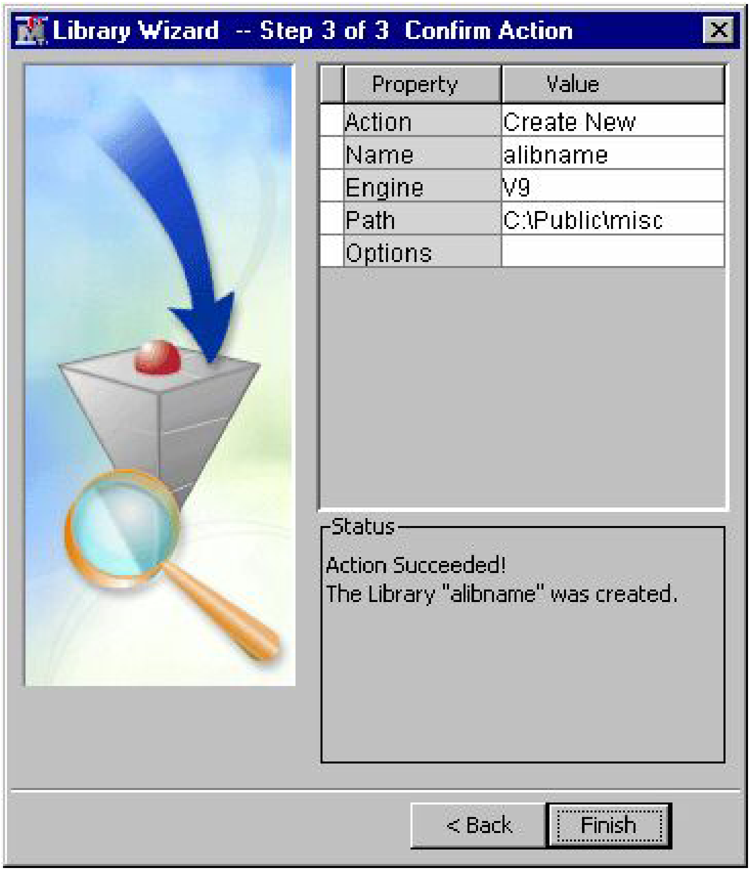
Path: c:\public\misc

Options:

Note that you can click **Browse** to open the Browse Folder Path window.



4. In the Confirm Action window, a summary of your LIBNAME definition properties is displayed. Click **Finish**.

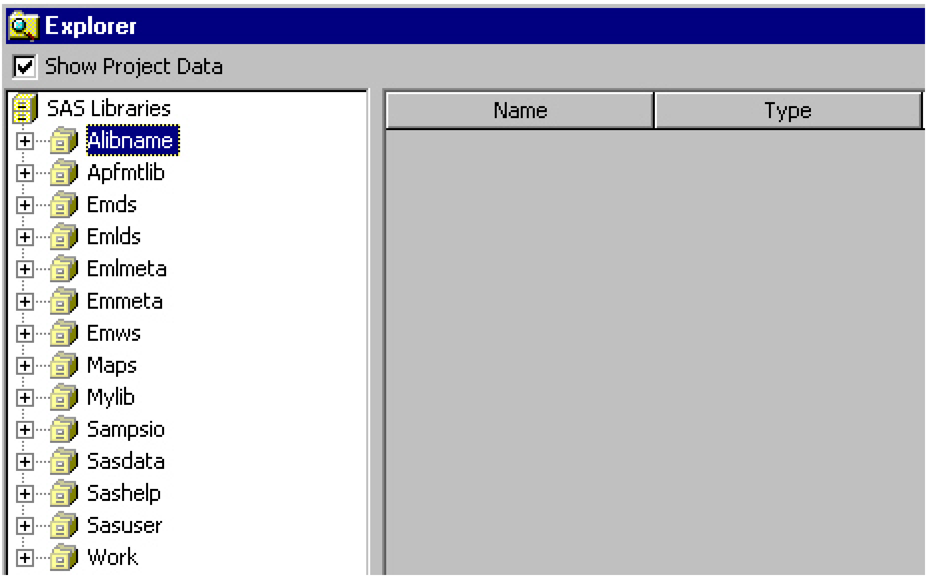


Property	Value
Action	Create New
Name	alibname
Engine	V9
Path	C:\Public\misc
Options	

Status

Action Succeeded!
The Library "alibname" was created.

5. Verify that your new library definition appears in the Library Explorer window. Select **View** ⇨ **Explorer**from the SAS Enterprise Miner window main menu.

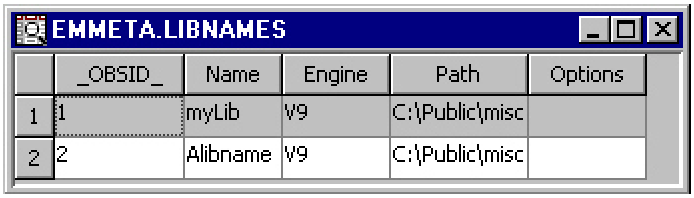


☑ Show Project Data

SAS Libraries
- Alibname
- Apfmtlib
- Emds
- Emlds
- Emlmeta
- Emmeta
- Emws
- Maps
- Mylib
- Sampsio
- Sasdata
- Sashelp
- Sasuser
- Work

Name	Type

Note: The library definitions are saved in a simple table named EMMETA.LIBNAMES.



	OBSID	Name	Engine	Path	Options
1	1	myLib	V9	C:\Public\misc	
2	2	Alibname	V9	C:\Public\misc	

Example: Importing a SAS Table from a SAS Metadata Repository

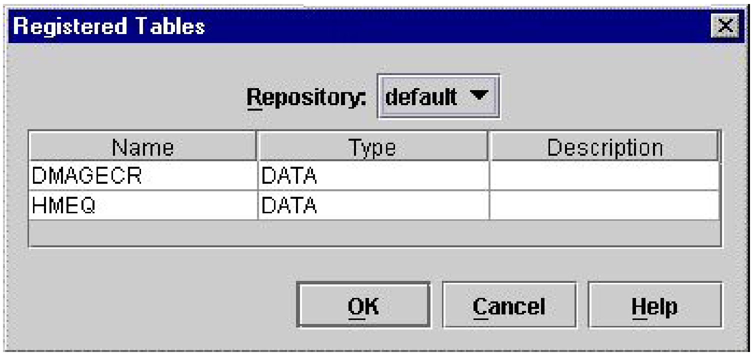
You can use SAS Data Integration Studio to create a SAS table definition and create a data source by importing the table in SAS Enterprise Miner. SAS Data Integration Studio is a SAS Open Metadata Architecture application that enables you to share metadata repository with other SAS Open Metadata Architecture applications such as SAS Enterprise Miner. See the SAS Data Integration Studio User's Guide online documentation at **`http://support.sas.com/documentation/onlinedoc/index.html`** for steps on how to create or register SAS library and table definitions.

Before you begin, you need to know the following information about your SAS Metadata Server that SAS Enterprise Miner runs on. You can find the information by selecting **Help** ⇨ **Configuration** from the SAS Enterprise Miner main menu.

- host name of your machine
- port
- user ID
- password
- name of the repository.

Importing the Table in SAS Enterprise Miner

1. In SAS Enterprise Miner, follow the steps in Creating a Data Source Using the Data Source Wizard on page 293 to create a data source. Select **Metadata Repository** as the source.
2. In the Data Source Wizard — Import Metadata from Metadata Repository window, click **Browse** to open the Registered Tables window.
3. The Registered Tables window display the metadata objects that you defined in SAS Data Integration Studio. Select a table (for example, DMAGECR) and click **Next**.
4. You can ignore the next window, because the library is pre-defined. If the library is not pre-defined, you will see the LIBNAME statement that is used to allocate the library.



Repository: default

Name	Type	Description
DMAGECR	DATA	
HMEQ	DATA	

Useful Tips

When working with Data Sources consider the following tips:

- SAS Enterprise Miner 6.2 and beyond support VALIDVARNAME=ANY column names through an option specified in the project start code. However, variable names are truncated to 32 characters and modified to contain only letters, numbers, or an underscore (_). A variable name must start with either a letter, or in certain special cases, an underscore (_).

  CAUTION:

  In general, you should not create variable names that begin with an underscore (_) because Enterprise Miner reserves the use of initial underscore (_) variable names during training. However, when using the Enterprise Miner Survival node, there are two conditions that require a variable with a specific name that begins with an underscore (_). The first condition occurs when you use the Survival node to score data (with role = SCORE), you must have a time interval variable named **_t_**. The second condition occurs when you use the Survival node to work with Expanded Format data. Expanded format data sets also require the time interval variable named **_t_**. If you are not scoring survival data, or working with Expanded Format survival data, creating Enterprise Miner variable names with an initial underscore (_) is generally not recommended, because the software reserves the use of variable names with an initial underscore (_) in all training data sets.

- Data sets in WORK should not be used to create data sources or as input data. When nodes run, they run in separate SAS sessions, and the WORK library maps to a different temporary directory.

- In a code node, you can create data sets in WORK. However, these data sets are temporary, and you cannot view them using the Enterprise Miner client. If you want to create results data sets in a code node or extension, you should use the EM_REGISTER macro. The data set will reside in the diagram library.

- You should avoid creating data sets in SASUSER. If you do create data sets in SASUSER, you might have permission problems because multiple users can share diagrams and every user has his or her own SASUSER.

Part 9

Node Reference

Chapter 20
Introduction to SEMMA

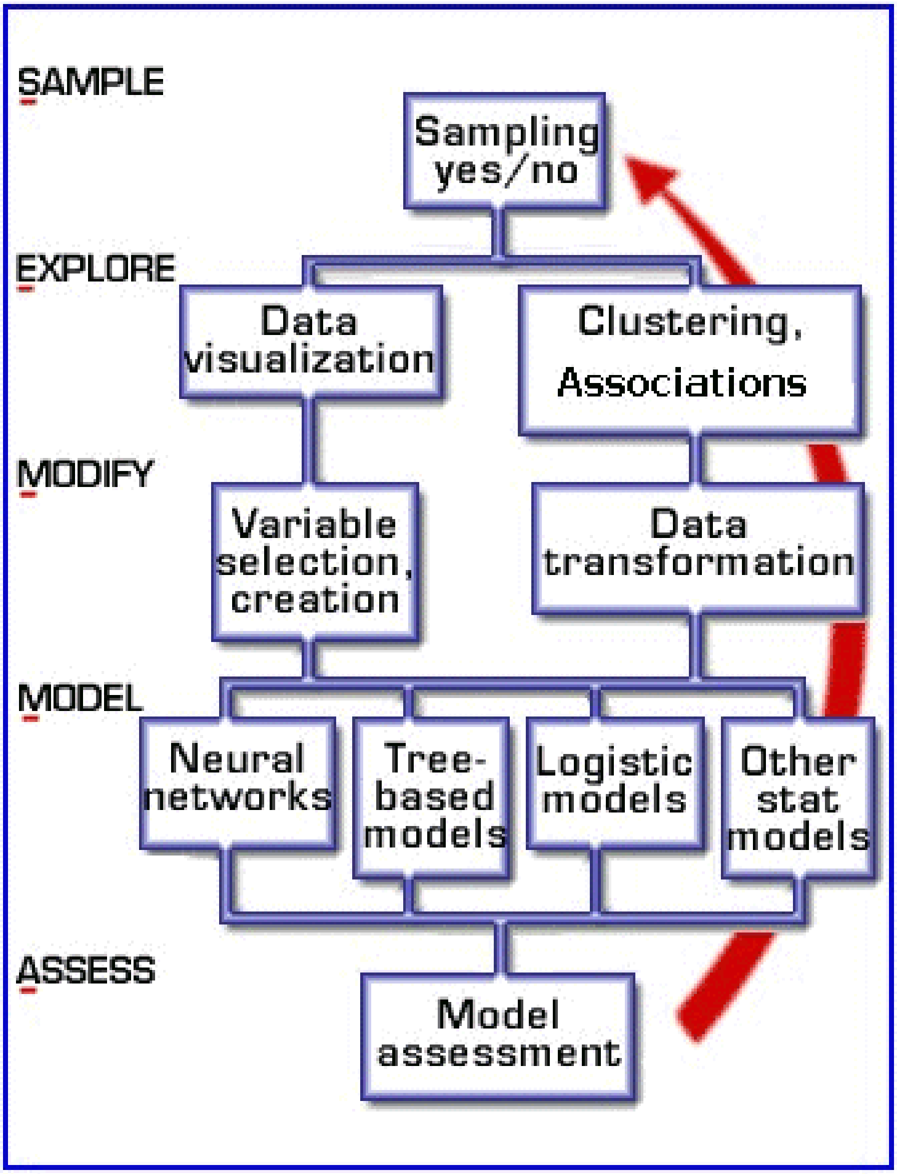
Introduction to SEMMA

SAS Institute defines data mining as the process of Sampling, Exploring, Modifying, Modeling, and Assessing (SEMMA) large amounts of data to uncover previously unknown patterns which can be utilized as a business advantage. The data mining process is applicable across a variety of industries and provides methodologies for such diverse business problems as fraud detection, householding, customer retention and attrition, database marketing, market segmentation, risk analysis, affinity analysis, customer satisfaction, bankruptcy prediction, and portfolio analysis.

Enterprise Miner software is an integrated product that provides an end-to-end business solution for data mining.

A graphical user interface (GUI) provides a user-friendly front end to the SEMMA data mining process:

- Sample the data by creating one or more data tables. The samples should be large enough to contain the significant information, yet small enough to process.
- Explore the data by searching for anticipated relationships, unanticipated trends, and anomalies in order to gain understanding and ideas.
- Modify the data by creating, selecting, and transforming the variables to focus the model selection process.
- Model the data by using the analytical tools to search for a combination of the data that reliably predicts a desired outcome.
- Assess the data by evaluating the usefulness and reliability of the findings from the data mining process.



You may or may not include all of the SEMMA steps in your analysis, and it may be necessary to repeat one or more of the steps several times before you are satisfied with the results. After you have completed the assess phase of the SEMMA process, you apply the scoring formula from one or more champion models to new data that may or may not contain the target. Scoring new data that is not available at the time of model training is the end result of most data mining problems.

The SEMMA data mining process is driven by a process flow diagram, which you can modify and save. The GUI is designed in such a way that the business analyst who has little statistical expertise can navigate through the data mining methodology, while the quantitative expert can go "behind the scenes" to fine-tune and tweak the analytical process.

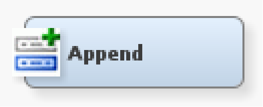
Enterprise Miner contains a collection of sophisticated analysis tools that have a common user-friendly interface that you can use to create and compare multiple models. Statistical tools include clustering, self-organizing maps (Kohonen maps), variable selection, trees, linear and logistic regression, and neural networking. Data preparation tools include outlier detection, variable transformations, data imputation, random sampling, and the partitioning of data sets (into train, test, and validate data sets). Advanced visualization tools enable you to quickly and easily examine large amounts of data in multidimensional histograms and to graphically compare modeling results.

Part 10

Node Reference: Sample Nodes

Chapter 21
Append Node

Append Node



Overview of the Append Node

The Append node tool is located on the Sample tab of the Enterprise Miner Toolbar. You use the Append node to append data sets that are exported by two different paths in a single process flow diagram. The Append node can also append train, validation, and test data sets into a new training data set.

Append Node and Data Sets with Missing Values

The Append node is unaffected by data sets that have observations that have missing values.

Using the Append Node

Data sources to be joined using the Append node must share a common variable. The common variable must have the same variable name and the same variable role and variable level in both data sources.

Append Node Properties

Append Node General Properties

The following general properties are associated with the Append node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Append node that is added to a diagram will have a Node ID of APPEND. The second Append node added to a diagram will have a Node ID of APPEND2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data - Append window. The Imported Data - Append window contains a list of the ports that provide data sources to the Append node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data - Append window. The Exported Data - Append window contains a list of the output data ports that the Append node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Append Node Train Properties

The following train properties are associated with the Append node:

- **Data Selector** — Use the Data Selector table to specify the status for individual data sources that can be imported into the Append node. Select the ... button to open a SAS Table Editor window that contains a table of the data sources that are available to the Append node. You set the value in the Use column to specify whether each data source in the table is appended to the data. The default Use setting for data sources connected to the Append node is Yes.

- **Output Type** — Use the Output Type property to specify whether the Append node should create data sets or data step views as output.
    - **Data** — the Append node produces data set output.
    - **View** — the Append node produces data step view output.
- **Action** — Use the Action property to specify what append action should be performed.
    - **By Role** — When set to By Role, the Append node combines data set partitions into similar data set partitions. Train data set partitions are appended to Train partitions, Validate data set partitions are appended to Validate partitions, and so on. If the data set being appended does not have data in the role being appended, it will be added. For example, if a data source that is partitioned into Train, Validate, and Test data roles is By Role-appended with a data source that is partitioned as Score data, the appended data set that is exported to successor nodes will contain partitions of Train, Validate, Test and Score data.
    - **Combine** — When set to Combine, the Append node will combine data sets that have the role of Train, Validate, and Test into the role of Train. For example, if a data source that is partitioned into Train, Validate, and Test data roles is Combine-appended with a data source that is partitioned as Score data, the appended data set that is exported to successor nodes will be partitioned only as Train data.

Append Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Append Node Results

You can open the Results window of the Append node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
    - **Settings** — displays a window with a read-only table of the configuration information in the Append Node Properties panel. The information was captured when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.

  - **Run Status** — indicates the status of the Append Node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Append node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Append node run.
  - **Output** — the SAS output of the Append node run.
  - **Flow Code** — the SAS code that was used to produce the output that the Append Node passes on to the next node in the process flow diagram.
  - **Append Code** — the SAS code used to append the new data set to the source data set.
- **Scoring**
  - **SAS Code** — the Append node does not generate SAS code.
  - **PMML Code** — the Append node does not generate PMML code.

Chapter 22

Data Partition Node

Data Partition Node

Overview of the Data Partition Node

Most data mining projects utilize large volumes of sampled data. After sampling, the data is usually partitioned before modeling.

Use the Data Partition node to partition your input data into one of the following data sets:

Train is used for preliminary model fitting. The analyst attempts to find the best model weights using this data set.

Validation is used to assess the adequacy of the model in the Model Comparison node.

The validation data set is also used for model fine-tuning in the following nodes:

- **Decision Tree node** — to create the best subtree.
- **Neural Network node** — to choose among network architectures or for the early-stopping of the training algorithm.
- **Regression node** — to choose a final subset of predictors from all the subsets computed during stepwise regression.

Test	is used to obtain a final, unbiased estimate of the generalization error of the model.

Partitioning provides mutually exclusive data sets. Two or more mutually exclusive data sets share no observations with each other. Partitioning the input data reduces the computation time of preliminary modeling runs. However, you should be aware that for small data sets, partitioning may be inefficient as the reduced sample size can degrade the fit of the model and its ability to generalize.

What are the steps to partitioning a data set? First, you specify a sampling method: simple random sampling, stratified random sampling, or cluster sampling. Then you specify the proportion of sampled observations to write to each output data set (Train, Validation, and Test). It is not uncommon to omit the test data set; that is, assign 0% of the observations to the test data set.

Data Partition Node Data Requirements

The Data Partition node must be preceded by a node that exports at least one raw, train, or document data table, which is usually an Input Data node or a Sample node.

Data Partition Node Properties

Data Partition Node General Properties

The following general properties are associated with the Data Partition node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Data Partition node that is added to a diagram will have a Node ID of Part. The second Data Partition node added to a diagram will have a Node ID of Part2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Data Partition window. The Imported Data — Data Partition window contains a list of the ports that provide data sources to the Data Partition node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data - Data Partition window. The Exported Data - Data Partition window contains a list of the output data ports that the Data Partition node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.

  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Data Partition Node Train Properties

The following train properties are associated with the Data Partition node:

- **Variables** — Use the Variables property to specify the status for individual variables that are imported into the Data Partitioning node. Select the ... button to open a window containing the variables table. You can specify the Partitioning Role for individual variables, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Output Type** — Use the Output Type property of the Data Partitioning node to specify the type of output the node should generate. The drop-down box offers choices of data set output (**Data**), or SAS data step views (**View**). The default setting for the Output Type property is **Data**.
- **Partitioning Method** — Use the Partitioning Method property to specify the sampling method that you want to use when you are partitioning your data.

  You can choose from:

  - **Default** — When Default is selected, if a class target variable or variables is specified, then the partitioning is stratified on the class target variables. Otherwise, simple random partitioning is performed. Note that if additional stratification variables are specified in the variables editor, they will be used in addition to the target variables.
  - **Simple Random** — When Simple Random is selected, every observation in the data set has the same probability of being written to one of the partitioned data sets.
  - **Cluster** — Using simple cluster partitioning, you allocate the distinct values of the cluster variable to the various partitions using simple random partitioning. Note that with this method, the partition percentages apply to the values of the cluster variable, and not to the number of observations that are in the partition data sets. If you select cluster partitioning as your method, you must use Variables property of the Data Partition node to set the Partition Role of the cluster variable (which may be the target variable) to Cluster.
  - **Stratified** — Using stratified partitioning, you specify variables to form strata (or subgroups) of the total population. Within each stratum, all observations have an equal probability of being written to one of the partitioned data sets. You perform stratified partitioning to preserve the strata proportions of the population within each partition data set. This might improve the classification precision of fitted models. If you select Stratified partitioning as your method and no class target variables are defined, then you must use the Variables window to set the partition role and specify a stratification variable.
- **Random Seed** — Use the Random Seed property to specify an integer value to use as a random seed for the pseudorandom number generator that chooses observations for sampling. The default value for the Random Seed property is 12345.

Data Partition Node Train Properties: Data Set Allocations

You can specify the percentages, or proportions of the source data that you wish to allocate as Train, Validation, and Test data sets.

- **Training** — Use the Training property to specify the percentage of observations that you want to allocate to the training data set. Permissible values are real numbers between 0 and 100. The SAS default training percentage is 40%.
- **Validation** — Use the Validation property to specify the percentage of observations that you want to allocate to the validation data set. Permissible values are real numbers between 0 and 100. The SAS default validation percentage is 30%.
- **Test** — Use the Test property to specify the percentage of observations that you want to allocate to the test data set. Permissible values are real numbers between 0 and 100. The SAS default test percentage is 30%.

Data Partition Node Report Properties

The following report properties are associated with the Data Partition node:

- **Interval Targets** — Use the Interval Targets property to specify whether you want to generate reporting statistics on interval targets in your data set. The Interval Targets summary displays a table of descriptive statistics based on the partitioned data sets and original data for the interval target variables. The default setting for the Interval Targets report property is Yes.
- **Class Targets** — Use the Class Targets property to specify whether you want to generate reporting statistics on the class targets in your data set. The Class Targets summary displays bar charts that compare the distribution of the class target variables in the incoming data set with the partitioned data sets for the same variables. The charts display the percent of the total frequency for each level of the target for each data set.

Note: The Data Partition node allows users to disable partitioned data summaries of interval and class target variables which will speed processing when working with very large data sets.

Data Partition Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Data Partition Node Variables Table

Use the table in the Variables — Data Partition window to identify cluster and stratification variables and to specify data partitioning methods for individual variables on a one-by-one basis. To open the Variables — Data Partition window, select the ... button to the right of the Variables property in the Properties Panel while the Data Partition node is selected in the Diagram Workspace.

Use this table to specify the partitioning role of specific variables. You can choose the following roles: Cluster, Stratification, None, and Default. If a partition role is dimmed and unavailable, then that partition role would not be applicable for that particular variable.

The following are read-only attributes: Role, Level, Type, Order, Label, Format, Informat, Length, Lower Limit, Upper Limit, Comment, Report, Creator, Family, and Distribution.

You can highlight a variable in the table and click the Explore button to get sampling, distribution, and metadata information in the Explore Variable window.

The table in the Variables — Data Partition window contains the following columns:

- **Name** — displays the name of the variable.
- **Partition Role** — click the cell to specify the partitioning method that you want to use on an individual variable in your imported data set.
  - **Default** — Uses the partitioning method that is specified for all variables in the Data Partition node Properties panel, unless the Cluster partitioning method is chosen. If the partitioning method specified for all variables in the Properties Panel is **Default**, then class variables will be stratified and interval variables will use simple random sampling. If the **Cluster** partitioning method is specified, you must change the Partition Role for the cluster variable from **Default** to **Cluster**.
  - **None** — Do not partition the specified class or interval variable.
  - **Cluster** — If the Partitioning Method property of the Data Partition node is set to **Cluster**, you must use the Partition Role column of the Variables window to specify the cluster variable. Cluster partitioning samples from a cluster of observations are similar in some way.
  - **Stratification** — you specify variables from the input data set to form strata (or subsets) of the total population. Within each stratum, all observations have an equal probability of being selected for the sample. Across all strata, however, the observations in the input data set generally do not have equal probabilities of being selected for the sample. You perform stratified sampling to preserve the strata proportions of the population within the sample. This may improve the classification precision of fitted models.

The following are read-only columns in the Variables — Data Partition table. You must use a Metadata node if you want to modify any of the following information about Data Source variables imported into the Data Partition node.

- **Role** — displays the variable role
- **Level** — displays the level for class variables
- **Type** — displays the variable type
- **Order** — displays the variable order
- **Label** — displays the text label to be used for variable in graphs and output
- **Format** — displays the variable format
- **Informat** — displays the SAS Informat for variable
- **Length** — displays the variable length

- **Lower Limit** — displays the minimum value for variable
- **Upper Limit** — displays the maximum value for variable
- **Distribution** — the expected univariate distribution
- **Family** — identifies the general source of a variable such as demographic, financial, historic for record keeping. Such values may be useful in constructing the data.
- **Report** — displays the Boolean setting for creating reports
- **Creator** — displays the user who created the process flow diagram
- **Comment** — displays the user-supplied comments about a variable

Data Partition Node Results

After a successful node run, you can open the Results window of the Data Partition node by selecting the **Results** button in the Run Status window, or by going to the Diagram Workspace, right-clicking the Data Partition node, and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window that shows how the Data Partition node properties were configured when the node last ran. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Data Partition node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the properties of the variables used in this node. Here, the nominal variable that is employed has been assigned as the clustering variable.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Data Partition node run.
  - **Output** — the SAS output of the Data Partition node run. The output displays a variable summary and summary statistics for class or interval targets or both in the original data and in the various partition data sets.
  - **Flow Code** — Flow Code is not available for the Data Partition node.
- **Scoring**
  - **SAS Code** — the Data Partition node does not generate SAS code.
  - **PMML Code** — the Data Partition node does not generate PMML code.
- **Summary Statistics**
  - **Interval Variables** — The Interval Variables summary displays a table of descriptive statistics based on the partition data sets and original data for the interval target variables. The statistics are as follows.
    - **Data** — the type of data
    - **Variable** — the interval target variable

        - **Maximum** — maximum value
        - **Mean** — arithmetic mean
        - **Minimum** — minimum value
        - **Number of Observations** — the number of non-missing observations
        - **Missing** — number of missing observations
        - **Standard Deviation** — the standard deviation.
    - **Class Variables** — displays bar charts that compare the distribution of the class target variables in the incoming data set with the partitioned data sets for the same variables. The charts display the percent of the total frequency for each level of the target for each data set.
- **Table** — displays a table that contains the underlying data used to produce the selected chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — displays the Graph Wizard that you can use to create ad-hoc plots that are based on the underlying data that produced the selected table.

Data Partition Node Output Data Sources

The Data Partition node exports up to three data sets, depending on the settings that you make in the Properties panel. By default, the node exports a Train, Validate, and Test data set.

Chapter 23
File Import Node

File Import Node

Overview of the File Import Node

You use the File Import node to convert selected external flat files, spreadsheets, and database tables into a format that Enterprise Miner recognizes as a data source and can use in data mining process flow diagrams. The File Import node is located on the Sample tab of the SAS Enterprise Miner tool bar.

The File Import node enables you to configure the data conversion process by selecting the file that you want to import, and specifying the metadata information (such as table and variable roles) that Enterprise Miner requires to perform data mining operations.

Requirements of the File Import Node

The File Import node can convert the following types of external data files into a data source format that Enterprise Miner recognizes:

Input Data Source	Extension
dBASE 5.0, IV, III+, and III files	.dbf
Stata	.dta

Input Data Source	Extension
Microsoft Excel	.xls
SAS JMP files	.jmp
Paradox .DB files	.db
SPSS	.sav
Lotus	.wk1, .wk3, .wk4
Tab-Delimited values	.txt
Comma-Separated values	.csv
Delimited values (user defined)	.dlm

The files you wish to convert for use with Enterprise Miner must be in a location that your Enterprise Miner server can reach through local or network access.

Tables Requiring Special Attention

If an observation in your imported data set contains an entry with a single quote, then you may experience undesirable behavior from the File Import node. Depending on the data that you are importing, you might want the delimiter to end the open quote or you might want the delimiter embedded inside a quoted string. Because the File Import node cannot make this decision, you must set a macro variable that ensures your data is handled properly.

The EFI_NOQUOTED_DELIMITER macro variable can be used to ensure the desired behavior. If you are having trouble importing .txt or .csv files, submit the following code in your project startup code:

```
%let EFI_NOQUOTED_DELIMITER=yes|no;
```

The value that you choose will depend on the behavior that you desire. If you notice undesired behavior, set a value (**yes** or **no**) for EFI_NOQUOTED_DELIMITER, rerun the File Import node, and determine if the desired output is generated. If the desired output is not generated, switch the value of EFI_NOQUOTED_DELIMITER and rerun the File Import node.

For example, consider the tab-delimited table of derivatives below. You can copy this table into a text editor and save it as a .txt file to confirm the results for yourself.

```
derivative    f1    f2    f3
f'    25    12    7
f''    -5    3    0
```

This data set contains four variables and two observations. You want the File Import node to create a data set that also contains four variables and two observations. If you specify **EFI_NOQUOTED_DELIMITER=no;**, then only the first column is imported by the File Import node. The single quotes in the two observations cause the File Import node to ignore any values that comes after these quotes. If you specify **EFI_NOQUOTED_DELIMITER=yes;**, then the File Import node imports all four columns as desired.

File Import Node Properties

File Import Node General Properties

The following general properties are associated with the File Import node:

- **Node ID** — displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first File Import node that is added to a diagram will have a Node ID of FIMPORT. The second File Import node added to a diagram will have a Node ID of FIMPORT2, and so on.
- **Imported Data** — provides access to the Imported Data — File Import window. The Imported Data — File Import window contains a list of the ports that provide data sources to the File Import node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — provides access to the Exported Data — File Import window. The Exported Data — File Import window contains a list of the output data ports that the File Import node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — stores notes of interest, such as data or configuration information. Select the ... button to the right of the Notes property to open the Notes window.

File Import Node Train Properties

The following train properties are associated with the File Import node:

- **Variables** — specifies the status for individual variables that are imported using the File Import node. Select the ... button to open a window containing the variables table. You can specify the Partitioning Role for individual variables, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. The Variables window enables you to verify that variable names have been imported properly, configure metadata, and view a variable's sampling information, observation values, or a plot of variable distribution.

  You can use the following buttons to accomplish these tasks:

  - **Apply** — changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — changes metadata back to its state before use of the Apply button.
  - **Label** — adds a column for a label for each variable.
  - **Mining** — adds columns for the Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — adds statistics metadata for each variable.
  - **Explore** — opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Import File** — enables you to select an external file to import. Select the ... button to the right of the Import File label to open the File Import window. The File Import window enables you to specify the location of the external file to import and check file format types that may be imported.

  You can use the following buttons to accomplish these tasks:

  - **My Computer** or **SAS Servers** — specifies whether the file to import is located locally or remotely. Select **My Computer** if the data file you want to import is located on your local machine. Select **SAS Servers** to import a data file located on your SAS Workspace Server.
  - **Browse** — finds the external file to import.
  - **View File Import Types** — opens the Valid Import File Types window, which contains information about currently valid file types by software vendor and file extension.

- **Maximum rows to import** — specifies a maximum number of rows to import.
- **Maximum columns to import** — specifies a maximum number of columns to import.
- **Delimiter** — specifies the delimiter that separates columns of data in the input file.
- **Name Row** — specifies whether to generate SAS variable names from the data values in the first record either in the input file or for the specified range. The default is Yes.
- **Number of rows to skip** — specifies a row number to begin reading data. By default, no rows are skipped.
- **Guessing Rows** — specifies the number of rows of the file to scan in order to determine the appropriate data type and length for the columns. The scan data process scans from row 1 to the number that is specified by this property, up to 32767. The default value is 500.
- **File Location** — specifies if the file is located locally or on a remote server.
- **Advanced Advisor** — specifies whether you used the Advanced Advisor window to configure additional metadata properties. The default setting is No.
- **Rerun** — specifies whether you want to force the File Import node to run each time the process flow diagram is rerun. Click to the right of the Rerun label and select Yes to force the File Import node to run with each execution of the process flow diagram. The default setting is No.

File Import Node Score Properties

The following score property is associated with the File Import node:

- **Role** — specifies the score role. Available options include Train, Validate, Test, Score, and Transaction. The default setting is Train.

File Import Node Report Properties

The following report property is associated with the File Import node:

- **Summarize** — specifies whether you want to create a summary report. The default setting is No.

File Import Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Using the File Import Node

The File Import node is used in place of a data source node in a process flow diagram. Before a File Import node can be used, you must select an external file, and configure the metadata for the data that is being imported.

You can select an external file to import by specifying whether the file is located on your computer or on a SAS Server, and the path to the file. After the file is located, you can specify the metadata for each variable that is imported.

Successfully configuring and running the File Import node allows subsequent nodes to use the data from the external file together with the metadata you configured with the File Import node for subsequent processing and analysis.

File Import Node Results

After a successful node run, you can open the Results window of the File Import node by selecting the Results button in the Run Status window, or by going to the Diagram Workspace, right-clicking the File Import node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the Results window:

- **Properties**
  - **Settings** — displays how the File Import node properties were configured when the node last ran.

  - **Run Status** — indicates the status of the File Import node run. Information about whether the run completed successfully, the Run Start Time, Run Duration, and the Run ID are displayed in this window.
  - **Variables** — displays a table of the properties associated with a variable that is imported, including the Role and Level of a variable, and whether the variable was dropped.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the File Import node run.
  - **Output** — the SAS output of the File Import node run. The output displays a variable summary and summary statistics for class or interval targets or both in the original data and in the various partition data sets.
  - **Flow Code** — Flow Code is not available for the File Import node.
- **Scoring**
  - **SAS Code** — the File Import node does not generate SAS code.
  - **PMML Code** — the File Import node does not generate PMML code.
- **Report**
  - **Statistics Table** — displays a Statistics Table containing statistics for import variables. Note that to generate a Statistics Table in the results, the Summarize property in the Report property pane must be set to Yes before the File Import node is run.

    The information presented in the Statistics Table is as follows:

    - **Variable Name** — name of the import variable.
    - **Type** — numeric or character.
    - **Number of Levels** — number of levels for class variables only.
    - **Percent Missing** — percent of missing values. For variables with no missing values, 0 is displayed.
    - **Minimum** — minimum values for numeric variables only. For character variables, "." (missing notation) is displayed.
    - **Maximum** — maximum values for numeric variables only. For character variables, "." (missing notation) is displayed.
    - **Mean** — arithmetic mean values for numeric variables only. For character variables, "." (missing notation) is displayed.
    - **Standard Deviation** — standard deviation values for numeric variables only. For character variables, "." (missing notation) is displayed.
    - **Skewness** — skewness for numeric variables only. For character variables, . is displayed.
    - **Kurtosis** — kurtosis values for numeric variables only. For character variables, . is displayed.
  - **Interval Targets** — displays the Interval Targets window. Note that to generate an Interval Targets window in the results, the Summarize property in the Report property pane must be set to Yes before the File Import node is run, and the target variable must be an interval variable.

  - **Class Targets** — displays the percent of each value as a bar chart. Note that to generate a Class Target window in the results, the Summarize property in the Report property pane must be set to Yes before the File Import node is run, and the target variable must be a class variable, such as binary.

- **Table** — displays a table that contains the underlying data used to produce the selected chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.

- **Plot** — displays the Select a Chart Type windows that you can use to create ad hoc plots that are based on the underlying data that produced the selected table. The Plot option is only available if the Statistics table is selected.

File Import Node Example

Introduction

This section explains how to import data in an external data source by using the File Import node.

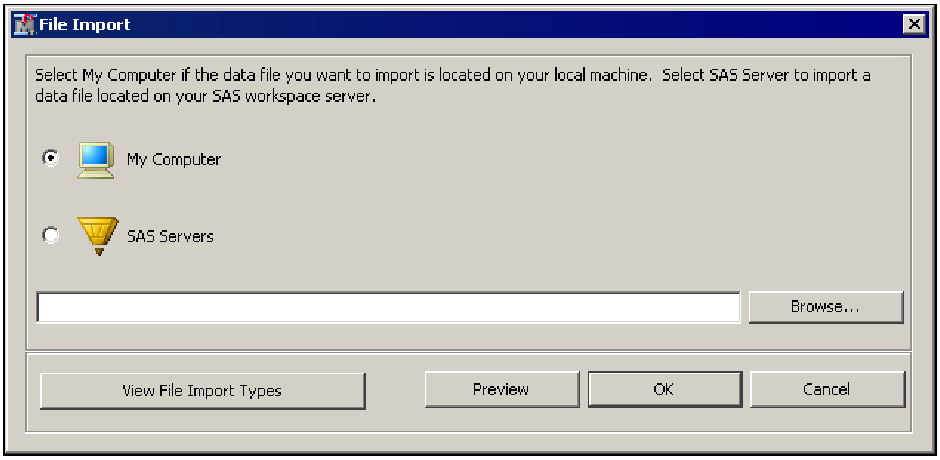
1. "Access the File Import Node" on page 343
2. "Select the External File to Import" on page 343
3. "Set Variable Roles" on page 345
4. "Set Additional Options" on page 347
5. "Run the File Import Node" on page 347

Access the File Import Node

To access the File Import node, put a File Import node into your Diagram Workspace. The File Import node is located on the Sample tab of the SAS Enterprise Miner toolbar. Alternatively, right-click on an empty space in an open Diagram Workspace and select **Add Node** ⇨ **Sample** ⇨ **File Import**.

Select the External File to Import

After the File Import node has been placed into the Diagram Workspace, select the File Import node. To specify the path to the external file you want to import, click the ... button for the Import File property. The File Import window opens.

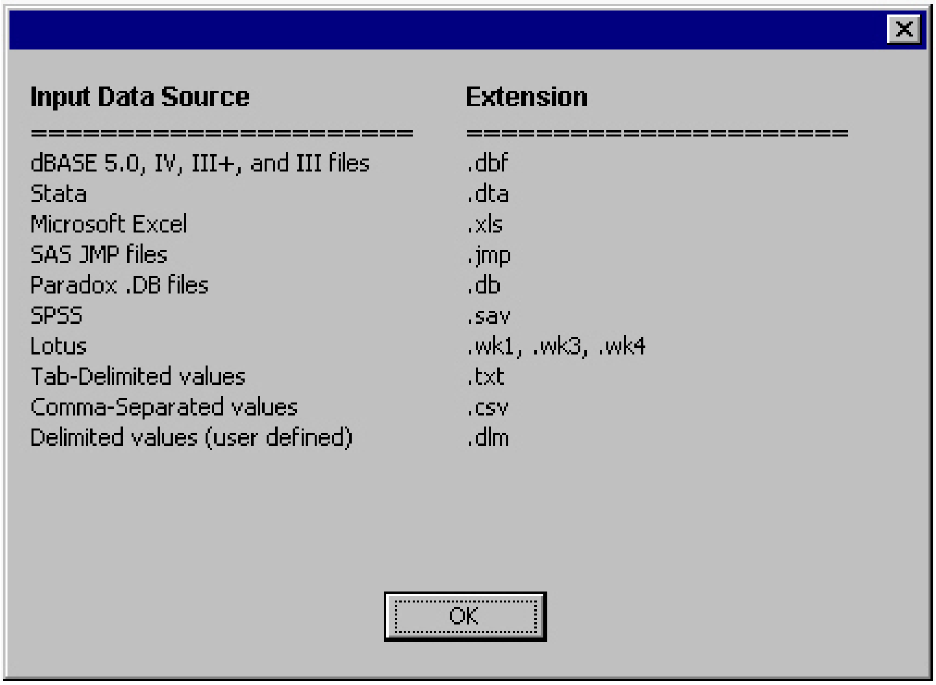


Select My Computer if the data file you want to import is located on your local machine. Select SAS Server to import a data file located on your SAS workspace server.

● My Computer

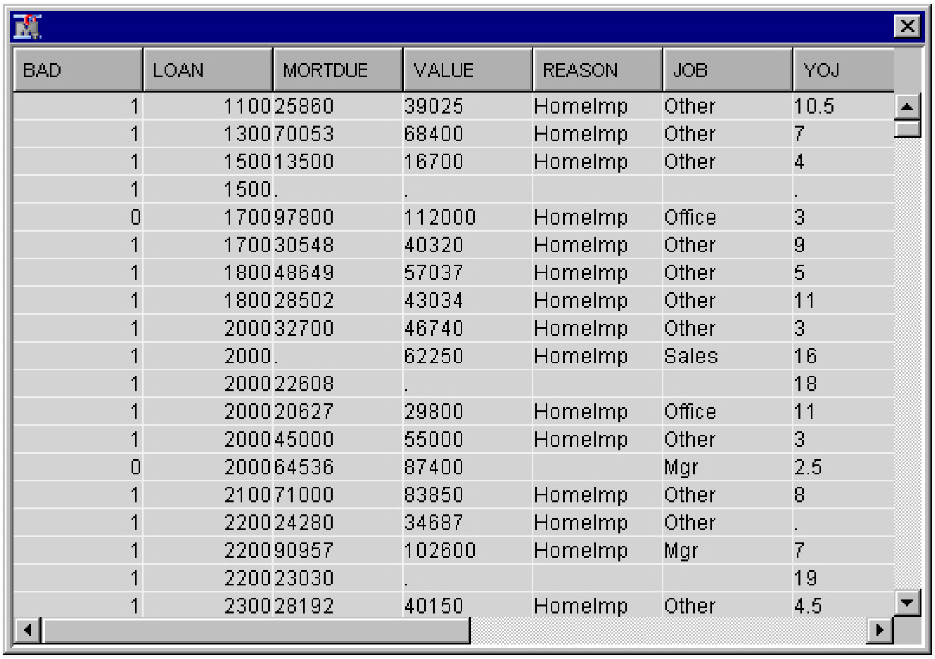
○ SAS Servers

Select My Computer if the data file you want to import is on your local machine, or select SAS Servers if the data file you want to import is on your SAS Workspace Server. Click View File Import Types to open the Valid Import File Types window, which contains information about currently valid file types. Use this resource to verify that the file you want to import is supported by the File Import node. Assume for this example that you want to import a Microsoft Excel file called *LoanResults.xls*.



Input Data Source	Extension
dBASE 5.0, IV, III+, and III files	.dbf
Stata	.dta
Microsoft Excel	.xls
SAS JMP files	.jmp
Paradox .DB files	.db
SPSS	.sav
Lotus	.wk1, .wk3, .wk4
Tab-Delimited values	.txt
Comma-Separated values	.csv
Delimited values (user defined)	.dlm

After reviewing the File Import Types window, you can verify that an .xls file is supported by the File Import node. Click **OK** to close the File Import Types window. Click **Browse** on the File Import window to navigate to the location of your external file and specify the path of the external file you want to import. After verifying that the correct path appears in the Browse text box, click **Preview** to preview the data set you are importing. Here is a sample preview for LoanResults.xls:

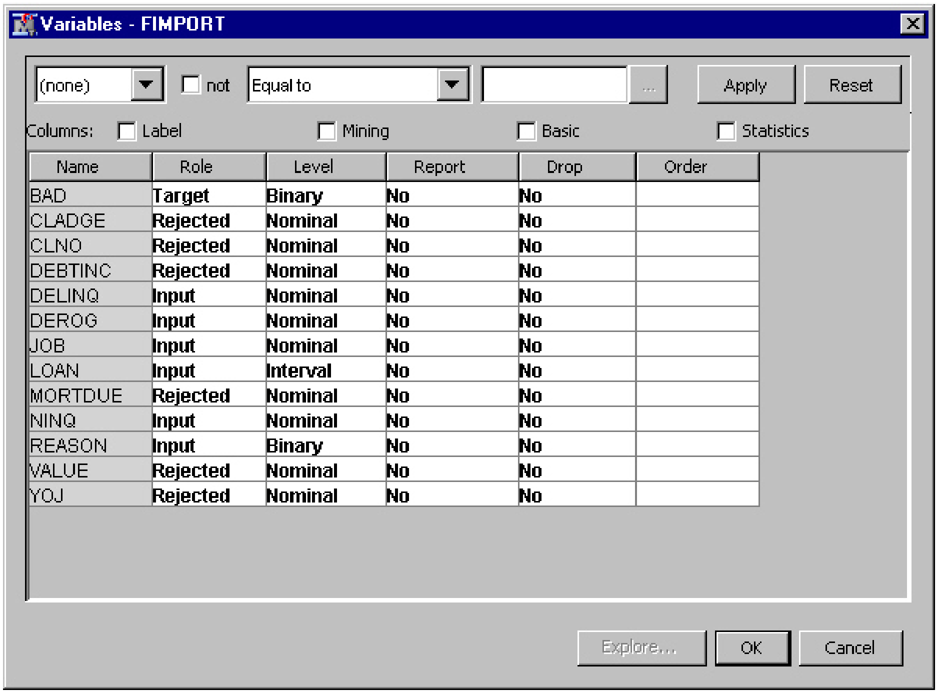


BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ
1	1100	25860	39025	HomeImp	Other	10.5
1	1300	70053	68400	HomeImp	Other	7
1	1500	13500	16700	HomeImp	Other	4
1	1500	.	.			.
0	1700	97800	112000	HomeImp	Office	3
1	1700	30548	40320	HomeImp	Other	9
1	1800	48649	57037	HomeImp	Other	5
1	1800	28502	43034	HomeImp	Other	11
1	2000	32700	46740	HomeImp	Other	3
1	2000	.	62250	HomeImp	Sales	16
1	2000	22608	.			18
1	2000	20627	29800	HomeImp	Office	11
1	2000	45000	55000	HomeImp	Other	3
0	2000	64536	87400		Mgr	2.5
1	2100	71000	83850	HomeImp	Other	8
1	2200	24280	34687	HomeImp	Other	.
1	2200	90957	102600	HomeImp	Mgr	7
1	2200	23030	.			19
1	2300	28192	40150	HomeImp	Other	4.5

Close the Preview window after you have finished checking your data set. Click **OK** on the File Import window to finish specifying the external file to import. After you have finished selecting the external file to import, you need to configure its metadata.

Set Variable Roles

After you have specified the external file you want to import, you can provide additional information about each variable that is being imported, such as the variable's role. With the File Import node still selected, click the ... button for the Variables property to open the Variables window for the File Import node. Alternatively, right-click the File Import node and select **Edit Variables** to open the Variables window.



Name	Role	Level	Report	Drop	Order
BAD	Target	Binary	No	No	
CLADGE	Rejected	Nominal	No	No	
CLNO	Rejected	Nominal	No	No	
DEBTINC	Rejected	Nominal	No	No	
DELINQ	Input	Nominal	No	No	
DEROG	Input	Nominal	No	No	
JOB	Input	Nominal	No	No	
LOAN	Input	Interval	No	No	
MORTDUE	Rejected	Nominal	No	No	
NINQ	Input	Nominal	No	No	
REASON	Input	Binary	No	No	
VALUE	Rejected	Nominal	No	No	
YOJ	Rejected	Nominal	No	No	

Check whether all of the variables from your external data set have been imported properly by examining the Name column. For example, if you are importing a Microsoft Excel file, you would expect column values in the first row in an .xls file to be recognized as variables.

Note that the variables appear in alphabetical order, which may or may not be the order the variables occur in the external data source you are importing. Next, click a variable in the Variables window. Then click **Explore** to view a variable's sampling information, observation values, or a plot of variable distribution. Note that the variables in the Explore window will appear in the order they are in your external data source.

After checking your data, close the Explore window. Then use the Variables window to specify values for the following default columns using the drop-down menu that can be accessed by clicking on the value of a variable in the appropriate column.

- **Role** — variable role.
- **Level** — level for class variables.
- **Report** — Boolean setting for creating reports.
- **Drop** — Boolean setting for dropping variables.
- **Order** — variable order.

You can add additional columns relating to a variable's label, or concerning basic, mining, or statistics information about a variable by marking the appropriate Columns check box. Note that to enable the Statistics check box you must set the Advanced Advisor Train property to **Yes** before the File Import node is run.

Select the Label check box to add the following columns:

- **Label** — a column for a variable label.

Select the Mining check box to add the following columns:

- **Lower Limit** — minimum value for a variable.
- **Upper Limit** — maximum value for a variable.
- **Creator** — user who created the process flow diagram.
- **Comment** — user-supplied comments about a variable.
- **Format Type** — variable format.

Select the Basic check box to add the following columns:

- **Type** — variable type.
- **Format** — SAS Format for variable.
- **Informat** — SAS Informat for variable
- **Length** — length of the variable.

Select the Statistics check box to add the following columns:

- **Number of Levels** — number of levels for class variables only.
- **Percent Missing** — percent of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — minimum values for numeric variables only. For character variables, "." (missing notation) is displayed.
- **Maximum** — maximum values for numeric variables only. For character variables, "." (missing notation) is displayed.

- **Mean** — arithmetic mean values for numeric variables only. For character variables, "." (missing notation) is displayed.
- **Standard Deviation** — standard deviation values for numeric variables only. For character variables, "." (missing notation) is displayed.
- **Skewness** — skewness for numeric variables only. For character variables, . is displayed.
- **Kurtosis** — kurtosis values for numeric variables only. For character variables, . is displayed.

At the top of the Variables Window, you can further configure variable metadata. Use the drop-down menus, check box, and selector button to specify how you want to change your metadata. Click **Apply** to implement the specified change, or Reset to reverse the effect of clicking Apply.

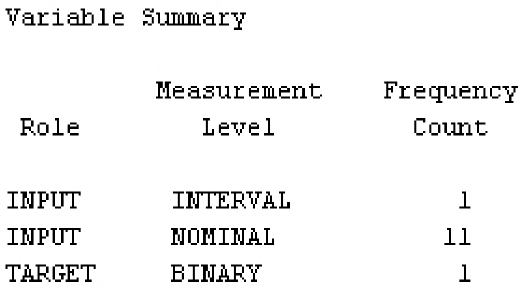
Set Additional Options

After you have finished specifying the metadata for data you are importing, you can specify additional import options. For example, you can specify the row to begin importing, the maximum number of rows and columns to import, or whether a delimiter should be used. See "File Import Node Properties" on page 339 for more information on import options.

Run the File Import Node

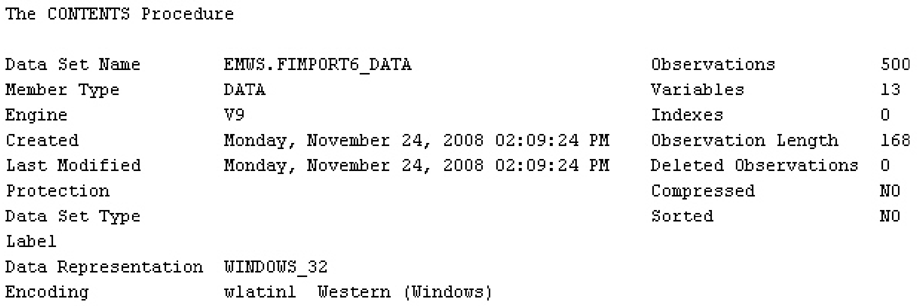
After you finish configuring the File Import node, you should run the File Import node and verify that the external data source was imported properly before you use the File Import node as a data source in a process flow diagram. Right-click the File Import node, and select **Run**.

After the File Import node has finished running, click **Results** on the Run Status window and maximize the Output window. In the Variable Summary section of the output, check the variable role information. The following is sample Variable Summary output.



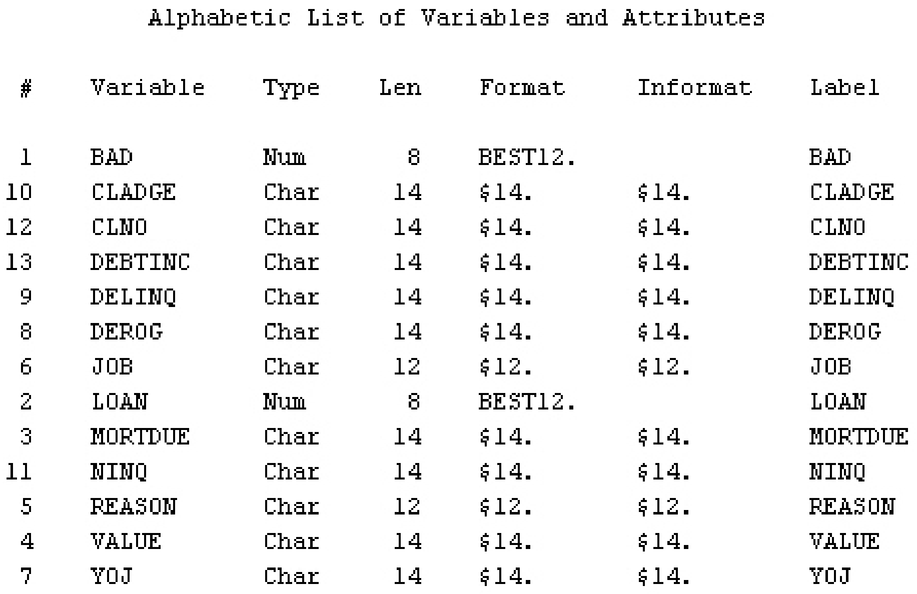
```
Variable Summary

           Measurement    Frequency
 Role         Level         Count

INPUT      INTERVAL           1
INPUT      NOMINAL           11
TARGET     BINARY             1
```

In the CONTENTS Procedure section of the output, check the number of observations that were processed. The following is sample output for the CONTENTS procedure:



```
The CONTENTS Procedure

Data Set Name        EMWS.FIMPORT6_DATA                    Observations         500
Member Type          DATA                                  Variables            13
Engine               V9                                    Indexes              0
Created              Monday, November 24, 2008 02:09:24 PM Observation Length   168
Last Modified        Monday, November 24, 2008 02:09:24 PM Deleted Observations 0
Protection                                                 Compressed           NO
Data Set Type                                              Sorted               NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Windows)
```

In the Alphabetical List of Variables and Attributes section of the output, check the variables that were imported and information about them. The following is sample output for the Alphabetical List of Variables and Attributes section.

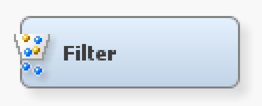


```
           Alphabetic List of Variables and Attributes

 #    Variable    Type    Len    Format     Informat    Label

 1    BAD         Num       8    BEST12.                BAD
10    CLADGE      Char     14    $14.       $14.        CLADGE
12    CLNO        Char     14    $14.       $14.        CLNO
13    DEBTINC     Char     14    $14.       $14.        DEBTINC
 9    DELINQ      Char     14    $14.       $14.        DELINQ
 8    DEROG       Char     14    $14.       $14.        DEROG
 6    JOB         Char     12    $12.       $12.        JOB
 2    LOAN        Num       8    BEST12.                LOAN
 3    MORTDUE     Char     14    $14.       $14.        MORTDUE
11    NINQ        Char     14    $14.       $14.        NINQ
 5    REASON      Char     12    $12.       $12.        REASON
 4    VALUE       Char     14    $14.       $14.        VALUE
 7    YOJ         Char     14    $14.       $14.        YOJ
```

After you have checked that conversion was successful, you can use the File Import node as a data source in a process flow diagram for further data mining analysis or predictive model creation.

Chapter 24
Filter Node

Filter Node



Overview of the Filter Node

The Filter node tool is located on the **Sample** tab of the Enterprise Miner tools bar. Use the Filter node to create and apply filters to your training data set and, optionally, to the validation and test data sets. You can use filters to exclude certain observations, such as extreme outliers and errant data that you do not want to include in your mining analysis. Filtering extreme values from the training data tends to produce better models because the parameter estimates are more stable. By default, the Filter node ignores target and rejected variables.

Filter Node Properties

Filter Node General Properties

The following general properties are associated with the Filter Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Filter node that is added to a diagram will have a Node ID of Filter. The second Filter node added to a diagram will have a Node ID of Filter2, and so on.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Filter window. The Imported Data — Filter window contains a list of the ports that provide data sources to the Filter node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Filter window. The Exported Data — Filter window contains a list of the output data ports that the Filter node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Filter Node Train Properties

The following train properties are associated with the Filter Node:

- **Export Table** — Use the Export Table property of the Filter node to indicate whether you want to export a data set containing filtered or excluded observations.
  - **Excluded** — Export the excluded observations.
  - **Filtered** — (default setting) Export the included observations.
  - **All** — Export all observations. A dummy indicator variable, M_FILTER, is created to identify the excluded observations. That is, M_FILTER is set to 1 for excluded observations and 0 otherwise.
- **Tables to Filter** — Use the Tables to Filter property of the Filter node to indicate whether you want to filter the training data set or all imported data sets.
  - **Training Data** — (default setting) Filter only the training data set.
  - **All Data Sets** — Filter all imported data sets.
- **Distribution Data Sets** — Use to create summary data sets to display histograms and bar charts when you set manual filters.
  - **Yes** — (default setting) The VARDIST and CLASSDIST data sets will be created if there are interval and class variables.
  - **No** — The data sets will not be created at training time.

  If you run the node with this property set to **Yes** and then change it to **No**, the tables will not be deleted. Regardless of the value of this property, you should be able to create or refresh the VARDIST and CLASSDIST data sets using their respective Variables editor.

Filter Node Train Properties: Class Variables

Use the Filter Node Class Variables properties to specify how you want to filter class variables. You can also use the Interactive Filter Class window to specify filter methods for individual variables.

- **Class Variables** — Click the ... button to the right of the Class Variables property to open the “Interactive Class Filter Window” on page 354 . You can use the columns in the table to specify Filtering Method, Keep Missing Values, Minimum Frequency Cutoff, Number of Levels Cutoff, and Report settings on a variable-by-variable basis, instead of applying a default filter to all class variables. The Role and Level values for a variable are displayed. You can see additional metadata information by selecting the Label, Mining, Basic, or Statistics check boxes.

- **Default Filtering Method** — Use the Default Filtering Method property of the Filter node to specify the method that you want to use to filter class variables.

    - **Rare Values (Count)** — Drop rare levels that have a count less than the level that you specify in the Minimum Frequency Cutoff property.

    - **Rare Values (Percentage)** — (default setting) Drop rare levels that occur in proportions lower than the percentage that you specify in the Minimum Cutoff for Percentage property.

    - **None** — No class variable filtering is performed.

- **Keep Missing Values** — Set the Keep Missing Values property of the Filter node to No if you want to filter out observations that contain missing values for class variables. The default setting for the Keep Missing Values property is Yes.

- **Normalized Values** — Set the Normalized Values property of the Filter node to No to suppress the normalization of class variable values before filtering. The default setting for the Normalize Values property is Yes.

    Categorical values (class variable level values) in Enterprise Miner are normalized as follows:

    - Left justified

    - Uppercased

    - Truncated to a length of 32 (after left-justification)

    This implies that values such as "YES", "yes", "Yes", and "yES" are all considered the same. Likewise, "purchasing a new big bright red ball" and "purchasing a new big bright red cap" are also considered identical, because they are identical under those rules for the first 32 characters.

- **Minimum Frequency Cutoff** — When you set the Default Filter Method for class variables to Rare Values (Count), you must use the Minimum Frequency Cutoff property of the Filter node to quantify the minimum count threshold. Class variable values that occur fewer times than the specified count are filtered. Permissible values are positive integers. The default value for the Minimum Frequency Cutoff property is 1.

- **Minimum Cutoff for Percentage** — When you set the Default Filter Method for class variables to Rare Values (Percentage), you must use the Minimum Percentage Cutoff property of the Filter node to specify the minimum percentage threshold. Observations with rare levels for a class variable are counted to calculate rare level proportions. Rare levels that do not meet the minimum percentage threshold are filtered. Permissible values are real numbers from 0 to 1. The default value is 1%, or 0.01.

- **Maximum Number of Levels Cutoff** — Use the Maximum Number of Levels Cutoff property to determine if a class variable in a data set will be considered for filtering. Only class variables that have fewer levels than the cutoff value are considered for filtering.

Filter Node Train Properties: Interval Variables

Use the Filter Node Interval Variables properties to specify how you want to filter interval variables.

You can configure and modify the following properties for the Filter Node:

- **Interval Variables** — Click the ... button to the right of the Interval Variables property to open the "Interactive Interval Filter Window" on page 355 for the data source that you wish to filter. The interactive window is especially useful if you wish to specify different filter methods for individual variables instead of applying the default filtering method to all interval variables. You can use the columns in the table to specify Filtering Method, Keep Missing Values, Filter Upper and Lower Limit, and Report settings on a variable-by-variable basis. The Role and Level values for a variable are displayed. You can see additional metadata information by selecting the Label, Mining, Basic, or Statistics check boxes. You can use the Interactive Interval Filter table to filter an interval variable for a single value by setting both the Filter Upper Limit and Filter Lower Limit columns for that variable to the desired variable value.
- **Default Filtering Method** — Use the Default Filtering Method property of the Filter node to specify the method that you want to use to filter interval variables. The Default Filtering Method applies only to input variables.
  - **Mean Absolute Deviation (MAD)** — The Median Absolute Deviation method eliminates values that are more than n deviations from the median. You specify the threshold value for the number of deviations, n, in the Cutoff for MAD property.
  - **User-Specified Limits** — The User-Specified method specifies a filter for observations that is based on the interval values that are displayed in the Filter Lower Limit and Filter Upper Limit columns of your data table. You specify these limits in the Variables table.
  - **Metadata Limits** — Metadata Limits are the lower and upper limit attributes that you can specify when you create a data source or when you are modifying the Variables table of an Input Data node on the diagram workspace.
  - **Extreme Percentiles** — The extreme percentiles method filters values that are in the top and bottom p^{th} percentiles of an interval variable's distribution. You specify the upper and lower threshold value for p in the Cutoff Percentiles for Extreme Percentiles property.
  - **Modal Center** — The Modal Center method eliminates values that are more than n spacings from the modal center. You specify the threshold value for the number of spacings, n, in the Cutoff Percentiles for Modal Center property.
  - **Standard Deviations from the Mean** — (default setting) The Standard Deviations from Mean method filters values that are greater than or equal to n standard deviations from the mean. You must use the Cutoff Percentiles for Standard Deviations property to specify the threshold value that you want to use for n.
  - **None** — Do not filter interval variables.

- **Keep Missing Values** — Set the Keep Missing Values property of the Filter node to No if you want to filter out observations that contain missing values for interval variables. The default setting for the Keep Missing Values property is Yes.
- **Tuning Parameters** — Click the ... button to the right of the Tuning Parameters property to open the Tuning Parameters window. You use the Tuning Parameters window to specify the tuning parameters, or cutoff values that correspond to the robust filter methods that are available in the Default Filtering Method property.
  - **Cutoff for MAD** — When you specify Mean Absolute Deviation as your Default Filtering Method, you must use the MAD Cutoff property of the Filter node to quantify n, the threshold value for the number of deviations from the median value. Observations with interval variable values that exceed the value of the Cutoff for MAD property are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
  - **Cutoff Percentiles for Extreme Percentiles** — When you specify Extreme Percentiles as your Default Filtering Method, you must use the Cutoff Percentiles for Extreme Percentiles property to specify p, the threshold value used to quantify the top and bottom pth percentiles. Observations with interval variable values in the extreme pth percentiles are filtered. Permissible values are percentages greater than or equal to 0 and less than 50. (P specifies upper and lower thresholds, 50% + 50% = 100%.) The default value is 0.5, or 0.5%.
  - **Cutoff for Modal Center** — When you specify Modal Center as your Default Filtering Method, you must use the Cutoff for Modal Center property to specify the threshold number of spaces n. Observations with interval values more than n spacings from the modal center are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
  - **Cutoff for Standard Deviation** — Use the Cutoff for Standard Deviation property to quantify n, the threshold for number of standard deviations from the mean. Observations with values more than n standard deviations from the mean are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 3.0.

Filter Node Score Properties

The following score properties are associated with the Filter Node:

- **Create Score Code** — Set the Create Score Code property to **No** if you want to suppress creation of score code by the node. The score code contains the filters that were created during training. The default setting for the Create Score Code property is **Yes**.
- **Update Measurement Level** — specifies whether the measurement level of class variables is updated.

Filter Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Interactive Class Filter Window

Introduction

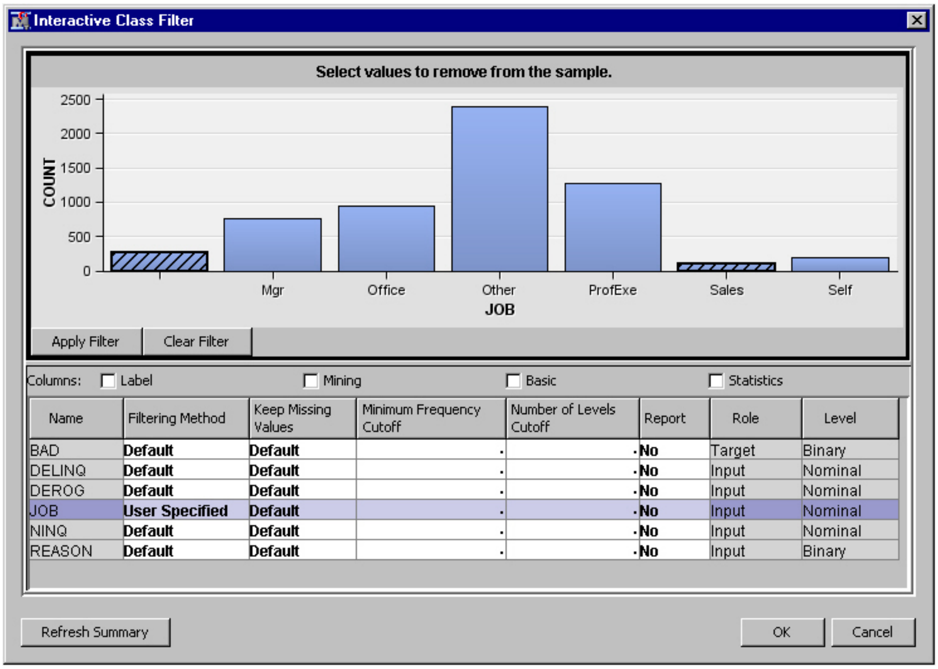
Use the Filter Variables editor to specify filtering properties for class variables on a variable-by-variable basis. You can configure and modify the Filtering Method, Keep Missing Values, Minimum Frequency Cutoff, Number of Levels Cutoff, and Report attributes. For details on these filtering properties, see the Filter Node "Filter Node Train Properties: Class Variables" on page 351 Properties.

Read-only attributes for the variable's Role and Level are also displayed. Additional read-only attributes can be displayed by selecting the Label, Mining, Basic, or Statistics check box. The Label option displays a Label read-only attribute. The Mining option displays the following read-only attributes: Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type. The Basic option displays the following read-only attributes: Type, Format, Informat, and Length. The Statistics option displays the following read-only attributes: Number of Levels, Percent Missing, Minimum, Maximum, Mean, Standard Deviation, Skewness, and Kurtosis.

Interactive Filtering

In addition to the filtering previously discussed, the Interactive Class Filter window provides the additional filtering method of **User Specified**. To interactively select values to exclude, select a variable in the table.

- The variable distribution displayed is based on a summary data set. To generate this data set, click **Refresh Summary**. Note that when the node runs, this data set gets refreshed or created.
- To select the values to exclude, click on the various bars. The selected bars should be outlined and shaded. Click **Apply Filter** to confirm your choices, and the Filtering Method field should change to User Specified.
- Click **Clear Filter** to clear the specified filtered values.



Name	Filtering Method	Keep Missing Values	Minimum Frequency Cutoff	Number of Levels Cutoff	Report	Role	Level
BAD	Default	Default	.	.	No	Target	Binary
DELINQ	Default	Default	.	.	No	Input	Nominal
DEROG	Default	Default	.	.	No	Input	Nominal
JOB	User Specified	Default	.	.	No	Input	Nominal
NINQ	Default	Default	.	.	No	Input	Nominal
REASON	Default	Default	.	.	No	Input	Binary

Interactive Interval Filter Window

Introduction

Use the editor to specify filtering properties for interval variables on a variable-by-variable basis. You can configure and modify the Filtering Method, Keep Missing Values, Filter Lower Limit, Filter Upper Limit, and Report attributes.

- **Filter Lower Limit** — a numeric field that can be used to define the lower limit of a variable filter.
- **Filter Upper Limit** — a numeric field that can be used to define the upper limit of a variable filter.

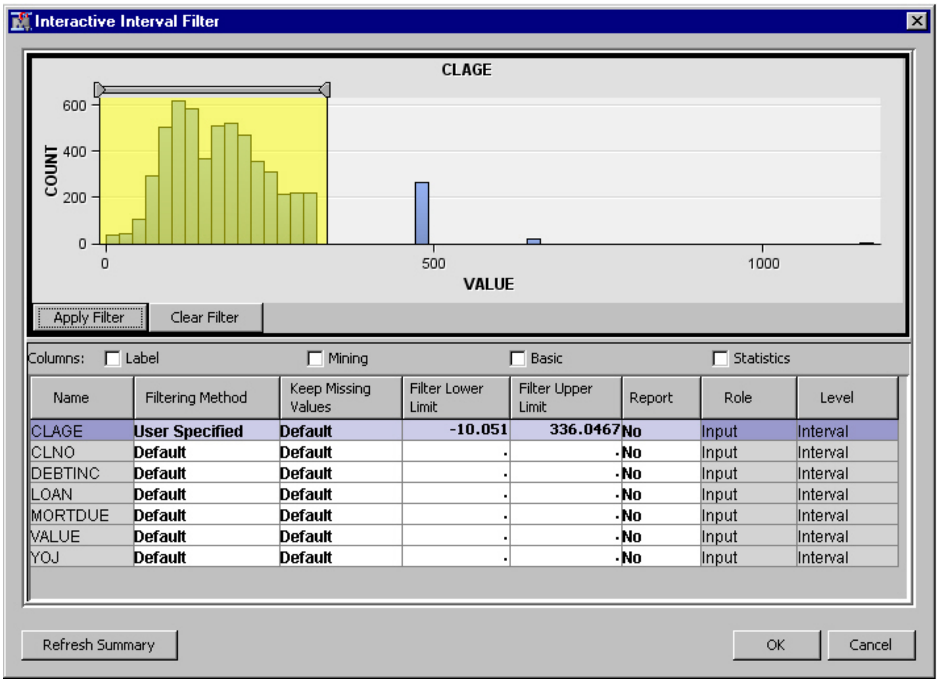
You can use the Interactive Interval Filter table to keep only records for only a specified single value of an interval variable. To keep only records for a specified value of an interval variable, set both the Filter Upper Limit and Filter Lower Limit columns for that variable to the desired variable value. For more details on these filtering properties, see Filter Node “Filter Node Train Properties: Interval Variables” on page 352 Properties.

Read-only attributes for the variable's Role and Level are also displayed. Additional read-only attributes can be displayed by selecting the Label, Mining, Basic, or Statistics check box. The Label option displays a Label read-only attribute. The Mining option displays the following read-only attributes: Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type. The Basic option displays the following read-only attributes: Type, Format, Informat, and Length. The Statistics option displays the following read-only attributes: Number of Levels, Percent Missing, Minimum, Maximum, Mean, Standard Deviation, Skewness, and Kurtosis.

Interactive Filtering

In addition to the standard filtering methods, the Interactive Interval Filter window provides the additional filtering method of **User Specified**. To interactively select a filtering range for a variable, select a variable in the table.

- The variable distribution displayed is based on a summary data set. To generate this data set, click **Refresh Summary**. Note that when the node runs, this data set gets refreshed or created.
- The shaded area identifies the range of values to be kept. To modify the filter limits, you can move the sliders at the top of the graph.
- You can also enter values directly in the Filter Lower Limit and/or Filter Upper Limit fields. After entering a value, the shaded area in the plot is updated.
- Click **Apply Filter** to confirm your choices, the Filtering Method field should change to User Specified.
- Click **Clear Filter** to clear the filter limits.



Columns: □ Label □ Mining □ Basic □ Statistics

Name	Filtering Method	Keep Missing Values	Filter Lower Limit	Filter Upper Limit	Report	Role	Level
CLAGE	User Specified	Default	-10.051	336.0467	No	Input	Interval
CLNO	Default	Default	.	.	No	Input	Interval
DEBTINC	Default	Default	.	.	No	Input	Interval
LOAN	Default	Default	.	.	No	Input	Interval
MORTDUE	Default	Default	.	.	No	Input	Interval
VALUE	Default	Default	.	.	No	Input	Interval
YOJ	Default	Default	.	.	No	Input	Interval

Filter Node Results

You can open the Results window of the Filter node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window that shows how the Filter node properties were configured when the node last ran. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Filter node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables used in this node.
  - **Train Code** — the code that Enterprise Miner used to train the node. This code is not available when the Table to Filter property is set to All Data Sets.

  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Filter node run.
  - **Output** — the SAS output of the Filter node run.
  - **Flow Code** — the SAS code used to produce the output that the Filter node passes on to the next node in the process flow diagram. This code is not available when the Table to Filter property is set to Training Data.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Filter node does not generate PMML code.
- **Filters**
  - **Excluded Class Variables** — displays a table of all of the class variables that were filtered from the training data set. The table provides data for class variable role, level, train count, train percent, label, filter method, and keep missing values.
  - **Limits for Interval Variables** — displays a table that shows the upper and lower limits that were established by the filter method for interval variables. The table provides data for interval variable role, minimum, maximum, filter method, keep missing values, and label.
- **Graphs** — when filters are applied, the node generates distribution plots for filtered variables that are specified as report variables in the Class Variables or Interval Variables property tables.
  - **Class Variables** — display bar charts that compare the distribution of class variables in the training data set with the filtered levels of the same variables. Use the drop down menu to select the variable that you want to view.
  - **Interval Variables** — display histograms that compare the distribution of interval variables in the training data set with the filtered variables. Use the drop down menu to select the variable that you want to view.
- **Table** — open the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart Type window so that you can plot the data in the table that you have in focus.

Chapter 25

Input Data Node

Input Data Node

Overview of the Input Data Node

The Input Data node represents the data source that you choose for your mining analysis and provides details (metadata) about the variables in the data source that you want to use.

The Input Data node is typically the first node that you use when creating a process flow diagram. Alternately, you can create a data set using the "SAS Code Node" on page 1109 and modify attributes using SAS programs. The Input Data node cannot be preceded by any other nodes. You can define multiple Input Data nodes in a single process flow diagram.

When you drag a data source on the diagram from the project, the Input Data node that you created keeps the same variable metadata as defined in the data source. You can modify variable attributes using the Input Data Variables Table Editor. Note that modifying the original data source in the project tree will not affect the metadata defined in associated input data nodes in the diagram.

By default, the data set keeps the role that was assigned when the data source was defined. The data set role determines how the data set is used throughout the process flow. More information on data set roles can be found in the "Input Data Node General Properties" on page 360 .

Input Data Node Properties

Input Data Node General Properties

The following general properties are associated with the Input Data node:

- **Node ID** — the Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Input Data node added to a diagram will have a Node ID of Ids. The second Input Data node added to the diagram will have a Node ID of Ids2.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Input Data window. The Imported Data — Input Data window contains a list of the ports that provide data sources to the Input Data node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Input Data window. The Exported Data — Input Data window contains a list of the output data ports that the Input Data node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Input Data Node Train Properties

The following train properties are associated with the Input Data node:

- **Output Type** — Use the Output Type property of the Input Data node to specify the type of output you want to generate.
  - **Data** — If Output Type is set to Data, then Enterprise Miner generates a data set, which is a static copy of the training table at the time that the node was run. A copy of the table is placed in the Enterprise Miner project's workspace folder. If your table is very large, you might want to generate a view instead; alternatively, you can create a sample of your data.

  - **View** — If Output Type is set to View, then Enterprise Miner generates a data step view of the training table.

  Note: When the Output Type property is set to **View**, you will always get the most up-to-date data. If your data is constantly updated, but you prefer a static view of the data instead of viewing the latest changes, set the Output Type property to **Data**.

- **Role** — Use the Role property of the Input Data node to specify the data set role that indicates how you want to use the data source in your process flow diagram.

  The role may be any of the following values:

  - **Raw** — data is used as raw input to the node.
  - **Train** (default) — data is used to fit initial models.
  - **Validate** — used by default for model assessment. The validation data set is also used for fine-tuning the model. The Tree and Neural Network nodes have the capacity of over-fitting the training data set. To prevent these nodes from over-fitting the training data set, the validation data set is automatically used to retreat to a simpler fit than the fit based on the training data alone. The validation data set can also be used by the Regression node for fine-tuning stepwise regression models.
  - **Test** — data is used to obtain a final, unbiased estimate of the generalization error of the model.
  - **Score** — used to score a new data set that may not contain the target.
  - **Transaction** — the Association, Path Analysis, and Time Series nodes expect a training data set of type Transaction.

- **Rerun** — The Rerun property of the Input Data node specifies whether you want to force the node to run each time the process flow diagram is rerun. If the data source that the node points to is refreshed, it will not be reread if the Input Data node has already run successfully and the Rerun property is in its default setting of No. Set the Rerun property to Yes to force the Input Data node to run with each execution of the process flow diagram. The Yes setting causes all node results to be refreshed upon node rerun.

- **Summarize** — Use the Summarize property of the Input Data node to specify whether to compute statistics for the active metadata. If the statistics already exist, then the statistics will be refreshed.

- **Drop Map Variables** — Use the Drop Map Variables property to indicate whether you want to drop the variables created to handle a validvarname=ANY column name when score code is generated.

Input Data Node Train Properties: Columns

- **Variables** — Use the Variables property to modify the properties that were defined in the data source. Select the ... button to open a variables table. You can change variable properties such as role, order, and level in the Variables table. You can also use the Variables property of the Input Data node to specify Lower Limit and Upper Limit values for numeric variables. The limits can then be used by successor nodes (for example, to filter the data).

  Note: For variables that are assigned the role **Key**, each observation must be unique. If your data set contains a **Key** variable and the observations are not unique, SAS Enterprise Miner will generate an error.

- **Decisions** — Use the Decisions property of the Input Data node to create or modify decision data for building models based on the value of the decisions. Click the ... button to open the Decision Processing window.
- **Refresh Metadata** — Use the Refresh Metadata property to refresh the metadata that is associated with the data source table, or a user-specified table. Refresh Metadata is useful when the data that is associated with an Enterprise Miner Data Source has changed in structure, such as the addition of one or more previously undefined columns. The metadata that is associated with new columns is generated using the Enterprise Miner Advisor, according to the Advisor property option setting. You can use the Variables Editor to modify variable attributes (such as Role and Level) for use in new columns. The table could have new columns, deleted columns, or columns with new attributes. The Refresh Metadata functionality lets users update metadata specifications in a manner that resembles the steps in the Enterprise Miner Data Source Wizard.
- **Advisor** — Use the **Basic** advisor setting to specify the initial measurement levels and roles for data mining variables that are determined using the variable attributes. Use the **Advanced** advisor setting to specify the initial measurement levels and roles for data mining variables that are determined by using both variable attributes and variable distributions.
- **Advanced Options** — When the Advisor property is set to Advanced, use the ... button to the right of the Advanced Options property to open a window that you can use to specify variable metadata settings for your data source.

  The following options are available in the Advanced Options window:

  - **Detect Class Levels** — specifies whether the number of class levels is determined for each variable.
  - **Class Levels Count Threshold** — specifies the maximum number of class levels for each variable. When the **Detect Class Levels** property is set to **Yes**, if there are more class levels than the value specified here, the variable is considered an interval variable. Valid values are positive integers greater than or equal to 2.
  - **Reject Vars with Excessive Missing Values** — specifies whether variables with a large percentage of missing values should be rejected.
  - **Missing Percentage Threshold** — specifies the maximum percentage of missing values allowed for each variable. When the **Reject Vars with Excessive Missing Values** property is set to **Yes**, a variable is rejected when the percentage of missing values is greater than the value specified here. Valid values are integers between 0 and 100, inclusive.
  - **Reject Vars with Excessive Class Values** — specifies whether variables with a large number of class variable levels should be rejected.
  - **Reject Levels Count Threshold** — specifies the maximum number of class variable levels allowed for each variable. When the **Reject Vars with Excessive Class Values** property is **Yes**, class variables with more levels than the value specified here are rejected.
  - **Identify Empty Columns** — specifies whether empty columns are assigned the value of MISSING.
  - **Database Pass-Through** — specifies if the data source advisor should use in-database processing to calculate summary statistics and determine measurement levels.

Input Data Node Train Properties: Data

- **Data Selection** — The Data Selection property of the Input Data node enables users to specify a working table instead of a static Enterprise Miner data source. It can be particularly useful to use a working table instead of a data source for your input data selection because working tables allow the data to change without needing to redefine the data source for each change.

    - **Data Source** — Use a previously defined SAS Enterprise Miner 12.3 data source for data training. (Default Setting)

    - **New Table** — Use a working table that is associated with the data source for data training. The user table must be compatible with specified data source metadata. When you select User Table as the value for your Data Selection property, you must use the New Table property to specify the name of the table that you want to use. For example, assume an analysis based on quarterly sales data. The sales data tables have differing names but share the same data structure. In this case, users can choose New Table in the Data Selection property and then simply specify the two-level table name for each quarterly sales report using the New Table property in the New Table Options section below.

- **Sample** — The Sample property indicates if a sample of the active data should be performed at training time. Set this property to **Yes** to create a sample and **No** to prevent the creation of a sample. When this property is set to **Default** the sample associated with the data source will be used if it exists.

- **Sample Options** — Use the ... button to the right of the Sample Options property to open a window that you can use to specify sampling preferences. You can sample either a percentage of the data or a specific number of observations.

Input Data Node Train Properties: Data Source

- **Data Source** — When the Data Selection property is set to Data Source, select the ... button to the right of the Data Source property to open a table that you use to specify the desired data source.

- **Data Source Properties** — Use the ... button to the right of the Data Source Properties to open a windows that you can use to view various properties of the data source. The properties are ID, Sample Data Set, Size Type, Sample Size, Created By, Create Date, Modified By, Modify Date, and Scope.

Input Data Node Train Properties: New Table

If you set the Data Selection property to New Table, you can use the New Table options to specify information about the replacement data table. The New Table Options properties are dimmed and unavailable if the Data Selection property is configured for any setting other than New Table.

- **New Table** — Use the New Table property of the Input Data node to specify the name of the SAS table that you want to use instead of an SAS Enterprise Miner 12.3 data source. Type in the two level SAS name for the table. If the Data Selection Property is not set to User Table, any information entered in the New Table property is ignored. The default setting for the New Table property is blank.

    Note: You must be sure that any table specified by the New Table property has a library assignment. You can assign the library via the project start code or the server start code.

- **Variable Validation** — Use the Variable Validation property of the Input Data node to specify the level of table data validation you want to perform when you use a new data table with the currently existing Data Source metadata. The type of variable validation performed controls how Enterprise Miner handles data type mismatches between the new and old tables.

  - **None** — Variable validation is not performed.
  - **Input Target** — All of the input variables and all of the target variables that are defined in the metadata must be in the new table and share the same data type attributes.
  - **Input** — All of the target variables that are defined in the metadata must be in the new table and share the same data type attributes.
  - **Strict** — All of the variables that are defined in the metadata must be in the new table and share the same data type attributes.

- **New Variable Role** — When you replace the table behind an existing Enterprise Miner Data Source with a new table, use the New Variable Role property to specify the role that Enterprise Miner should assign to new variables in the replacement table that are not defined in the existing metadata. The new variables can be assigned a variable type of Input or Rejected. The default setting for the New Variable Role property is **Rejected**.

Input Data Node Train Properties: Metadata

- **Table** — is the name of the data source table.
- **Library** — is the SAS Library where the table resides.
- **Description** — is a description of data source.
- **Role** — specifies the role of the data in the data source.
- **No. Obs** — indicates the number of rows in the data source.
- **No. Cols** — indicates the number of columns in the data source.
- **No. Bytes** — indicates the approximate files size of the data source in bytes.
- **Segment** — indicates the segment of the data source.
- **Scope** — indicates whether the table exists locally or globally.

Input Data Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Input Data Node Results

After a successful node run, you can open the Results window of the Input Data node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window that shows how the Input Data node properties were configured when the node last ran.
  - **Run Status** — indicates the status of the Input Data node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Input Data node run.
  - **Output** — the SAS output of the Input Data node run. The SAS output displays a variable summary, output table attributes, and input table attributes.
  - **Flow Code** — the SAS code used to produce the output that the Input Data node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — is not available in the Input Data node.
  - **PMML Code** — the Input Data node does not generate PMML code.

Chapter 26
Merge Node

Merge Node

Overview of the Merge Node

Use the **Merge** node to merge observations from two or more data sets into a single observation in a new data set. The **Merge** node is found on the **Sample** tab of the SAS Enterprise Miner node tools bar.

Data Set Requirements of the Merge Node

Data sets of types Train, Validate, Test, and Score can be merged. The **Merge** node supports both one-to-one and match merging and also provides users with the ability to not overwrite certain variables (such as predicted values and posterior probabilities) depending on how the node is configured.

Merge Node Properties

Merge Node General Properties

The following general properties are associated with the **Merge** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Merge node that is added to a diagram will have a Node ID of Merge. The second Merge node that is added to a diagram will have a Node ID of Merge2, and so on.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Merge window. The Imported Data — Merge window contains a list of the ports that provide data sources to the **Merge** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Merge window. The Exported Data — Merge window contains a list of the output data ports that the **Merge** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Merge Node Train Properties

- **Variables** — Select the ... button to the right of the Variables property to open the "Merge Node Variables Window" on page 370 .
- **Merging** — Use the Merging property to specify the type of merge that you want to perform.
  - **Match** — combines observations from two or more SAS data sets into a single observation in a new data set according to the values of common variables. All data sets are sorted by the BY variables for match merging. You specify BY variables in the "Merge Node Variables Window" on page 370 . The number of observations in the new data set is the sum of the largest number of observations in each BY group in all data sets.
  - **One-to-One** — (default setting) combines observations from two or more SAS data sets into a single observation in a new data set. In a one-to-one merge, the number of observations in the new data set is equal to the number of observations in the largest data set.
- **By Ordering** — Launches an editor that enables you to specify the order of the BY variables. You can enable this option by setting the value of the Merging property to **Match**.

- **Overwrite Variables** — Use the Overwrite Variables property to specify permission to overwrite certain variables that are common to data sets. The default setting for the Overwrite Variables property is **No**. Setting the Overwrite Variables property to **Yes** disables the Variables Group properties. You use the properties in the Variables Group to specify which variables you want to allow to be overwritten.

  Note that this property applies only to the variable roles listed in the **Variables Group** property group. That is, your variables might be overwritten if they are not set to one of the following roles: Assessment, Classification, Posterior, Prediction, Residual, Segment. To guarantee that a variable will not be overwritten, right-click the **Merge** node and select **Edit Variables**. In the Variables window, set the **Overwrite Variable** property to **No** for that variable.

Merge Node Train Properties: Variables Group

The Variables Group properties are unavailable when the Merge node **Overwrite Variables** property is set to **Yes**. When the **Overwrite Variables** property is set to **No**, use the Variables Group properties to specify which common variables you want to permit the **Merge** node to overwrite.

- **Segment** — overwrite segment variables in merging data sets. SAS Enterprise Miner ignores this setting unless the Overwrite Variables property is set to **Yes**. The default setting for the Segment property is **No**.
- **Assess** — overwrite assessment variables. SAS Enterprise Miner ignores this setting unless the Overwrite Variables property is set to **Yes**. The default setting for the Assess property is **No**.
- **Classification** — overwrite classification variables. SAS Enterprise Miner ignores this setting unless the Overwrite Variables property is set to **Yes**. The default setting for the Classification property is **No**.
- **Predicted or Posterior** — overwrite posterior probability variables (class targets) or overwrite predicted variables (interval targets). SAS Enterprise Miner ignores this setting unless the Overwrite Variables property is set to **Yes**. The default setting for the Overwrite Predicted or Posterior Variables property is **No**.
- **Residual** — overwrite residual variables. Residual variables are generated when you model class targets or interval targets and can be identified by the prefix R_. SAS Enterprise Miner ignores this setting unless the Overwrite Variables property is set to **Yes**. The default setting for the Residual property is **No**.

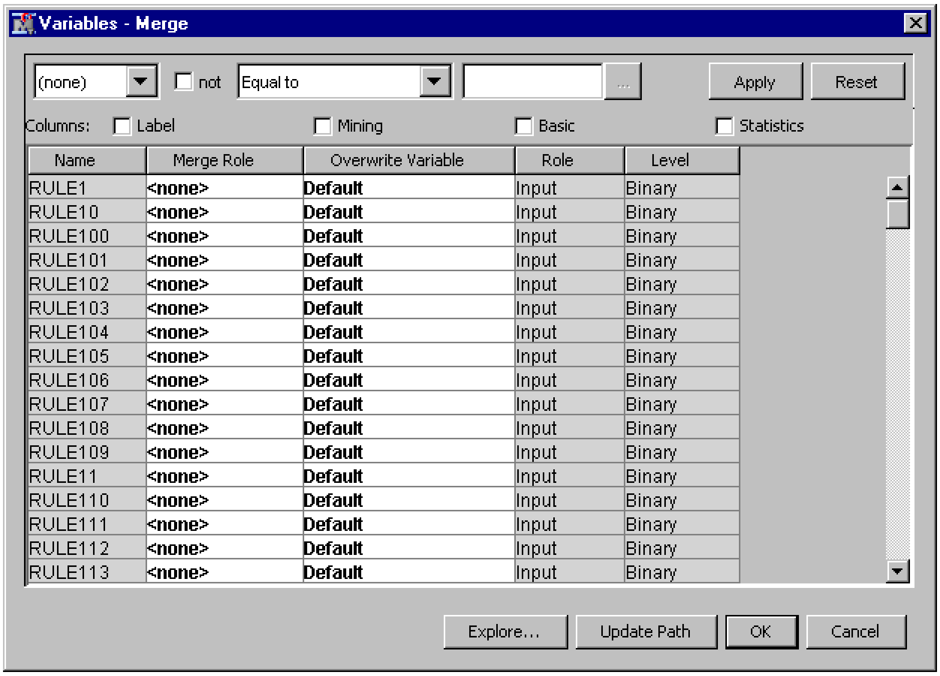
Merge Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Merge Node Variables Window

Use the table in the Variables — Merge window to identify cluster and stratification variables and to specify data partitioning methods for individual variables on a one-by-one basis. To open the Variables — Merge window, select the button to the right of the Variables property in the Properties Panel while the **Merge** node is selected in the diagram workspace. The example below uses the SAMPSIO.CUSTOMERS data set that is used in the “Merge Node Example” on page 373 .



Name	Merge Role	Overwrite Variable	Role	Level
RULE1	<none>	Default	Input	Binary
RULE10	<none>	Default	Input	Binary
RULE100	<none>	Default	Input	Binary
RULE101	<none>	Default	Input	Binary
RULE102	<none>	Default	Input	Binary
RULE103	<none>	Default	Input	Binary
RULE104	<none>	Default	Input	Binary
RULE105	<none>	Default	Input	Binary
RULE106	<none>	Default	Input	Binary
RULE107	<none>	Default	Input	Binary
RULE108	<none>	Default	Input	Binary
RULE109	<none>	Default	Input	Binary
RULE11	<none>	Default	Input	Binary
RULE110	<none>	Default	Input	Binary
RULE111	<none>	Default	Input	Binary
RULE112	<none>	Default	Input	Binary
RULE113	<none>	Default	Input	Binary

You can resize the columns in the Variables — Merge window to enhance readability. Click on a column heading to toggle between ascending and descending column and table sorts. Information in cells with a white background can be configured. Cells with gray backgrounds are read-only.

You can highlight a variable in the table and click the **Explore** button to get Sampling, distribution, and metadata information in the Explore Variable window.

The table in the Variables — Merge window contains two columns that you can modify:

- **Merge Role** — click the cell to specify BY variables when you are performing Match Merging.
- **Overwrite Variable** — click the cell to specify whether you want to overwrite this variable during the merge.
  - **Default** — use the overwrite behavior that is specified for this variable type in the **Merge** node Properties panel.
  - **Yes** — overwrite this variable during merge operations, even if the specifications for this variable type in the Merge node Properties panel indicate otherwise.
  - **No** — do not overwrite this variable during merge operations, even if the specifications for this variable type in the Merge node Properties panel indicate otherwise. Variables that are not overwritten are renamed and listed in the node output.

The Name, Role, and Level (for class variables) values for a variable are displayed as read-only properties.

The following buttons and check boxes provide additional options to view and modify variable metadata:

- **Apply** — changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
- **Reset** — changes metadata back to its state before use of the **Apply** button.
- **Label** — displays the text label to be used for the variable in graphs and output.
- **Mining** — adds columns for the Order, Report, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Order** — displays the variable order.
  - **Report** — displays the Boolean setting for creating reports.
  - **Lower Limit** — displays the minimum value for the variable.
  - **Upper Limit** — displays the maximum value for the variable.
  - **Creator** — displays the user who created the process flow diagram.
  - **Comment** — displays the user-supplied comments about a variable.
  - **Format Type** — displays the variable format type.
- **Basic** — adds columns for the Type, Format, Informat, and Length of each variable.
  - **Type** — displays the variable type.
  - **Format** — displays the variable format.
  - **Informat** — displays the SAS Informat for the variable.
  - **Length** — displays the variable length.
- **Statistics** — adds statistics metadata for each variable.
- **Explore** — opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

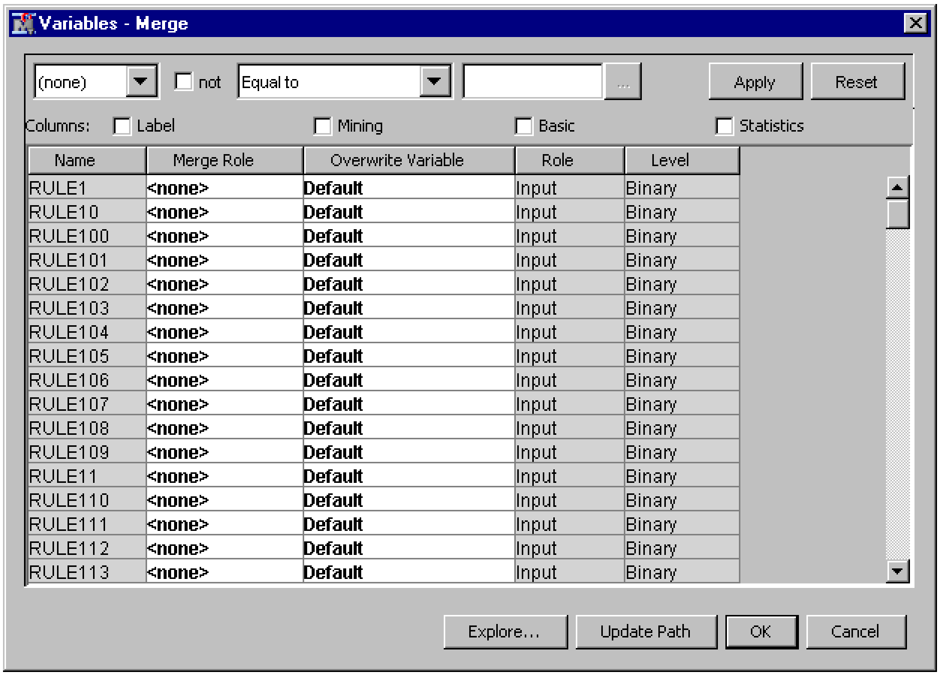
Note: You can use a Metadata node if you need to modify any of the preceding information about predecessor node data tables that are imported into the **Merge** node.

Setting the Merge Method

You use the Properties Panel of the **Merge** node to specify the merge method and overwrite variables settings to be applied to the set of imported variables. The merge method that you specify in the general section of the Merge Properties Panel are applied to the set of all non-rejected variables passed to the node. Right-click the Value cell next to Merging to choose the merging method for your data. You can choose between One-to-One and Match merging.

If you choose to overwrite certain variable types (Segment, Assess, Classification, Predicted or Posterior, or Residual) or not to overwrite certain variable types, you make those settings in the Variables Group section of the Properties Panel after setting the Overwrite Variables property to **Yes**.

It is possible to override the Merge Role and Overwrite Variables setting for one or more individual variables. You must open the Variables window where you can specify a new Merge Role or Overwrite Variable setting.



Columns: □ Label □ Mining □ Basic □ Statistics

Name	Merge Role	Overwrite Variable	Role	Level
RULE1	<none>	Default	Input	Binary
RULE10	<none>	Default	Input	Binary
RULE100	<none>	Default	Input	Binary
RULE101	<none>	Default	Input	Binary
RULE102	<none>	Default	Input	Binary
RULE103	<none>	Default	Input	Binary
RULE104	<none>	Default	Input	Binary
RULE105	<none>	Default	Input	Binary
RULE106	<none>	Default	Input	Binary
RULE107	<none>	Default	Input	Binary
RULE108	<none>	Default	Input	Binary
RULE109	<none>	Default	Input	Binary
RULE11	<none>	Default	Input	Binary
RULE110	<none>	Default	Input	Binary
RULE111	<none>	Default	Input	Binary
RULE112	<none>	Default	Input	Binary
RULE113	<none>	Default	Input	Binary

Merge Node Results

After a successful node run, you can open the Results window of the **Merge** node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Merge node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Merge node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays any user-supplied notes of interest, such as data or configuration information.
- **SAS Results**
  - **Log** — the SAS log of the Merge node run.
  - **Output** — the SAS output of the Merge node run. The SAS output displays a variable summary, a chart of merged table attributes, and a list of variables that were renamed.
  - **Flow Code** — the SAS code used to produce the output that the **Merge** node passes on to the next node in the process flow diagram.
- **Scoring**

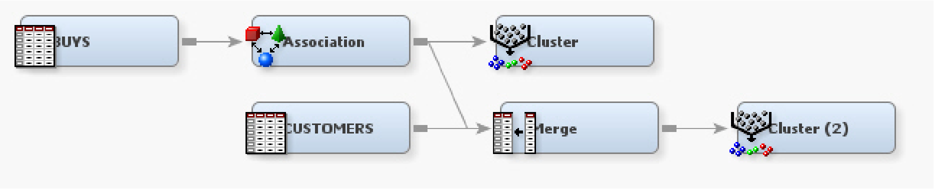
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **Merge** node does not generate PMML code.
- **Table** — opens the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart plotting wizard so that you can plot the data in the table that you have in focus.

Merge Node Example

Introduction

The following example merges customer transaction data with customer demographic data. In the example scenario, you examine clusters of customer transactions that are based on association sequence rules and compare them to clusters that are based on customer demographics. More analysis can be performed on the data, but it is beyond the scope of demonstrating the functionality of the **Merge** node. This example merges customer-based transaction and demographic data sets together so that you can form clusters of the purchasers' demographic data.

The process flow diagram that you build in the **Merge** node example will resemble the following:



The example assumes that you have a blank diagram open in an existing SAS Enterprise Miner project.

The following are the main steps to the **Merge** node example:

- "Generate the Example Data" on page 373
- "Create SAS Enterprise Miner Data Sources" on page 375
- "Find Association Rules for the Transaction Data" on page 379
- "Examine the Association Output" on page 381
- "Cluster Individuals Based on Transactions" on page 381
- "Merge the Demographic Data with the Transaction Data" on page 383
- "Analyze Customers Based on Merged Data" on page 384

Generate the Example Data

The data used in this example is generated from data sets in the SAMPSIO samples library that is shipped with SAS. The transaction data set that is created for this example is called SAMPSIO.BUYS. The customer demographic data set that is created for this example is called SAMPSIO.CUSTOMERS. You must submit the following SAS code to SAS Enterprise Miner to create the two example data sets before you can create the SAS Enterprise Miner data sources to use in your process flow diagram.

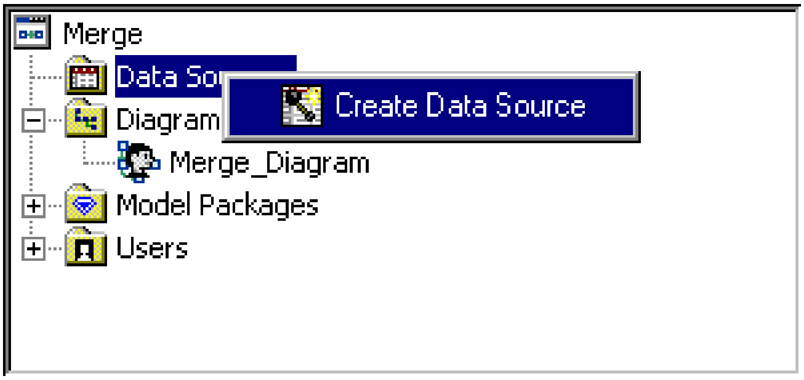
Open the SAS Enterprise Miner Program Editor from the main menu by selecting **View** ⇨ **Program Editor**. Cut and paste the SAS code that follows into the SAS Enterprise Miner Program Editor, and submit it by clicking the submit tool icon. Afterward, use the main menu to view the SAS Log by selecting the **Log** tab in the Program Editor window, and ensure that the code runs with no errors. After you verify that the code ran successfully and created the example data sets SAMPSIO.BUYS and SAMPSIO.CUSTOMERS, move on to the step to Create Data Sources in SAS Enterprise Miner.

```
%LET newds=SAMPSIO.BUYS ;

     PROC FREQ data=sampsio.assocs noprint ;
       table product / out=items; run ;
     DATA items ;
        length product $32 ;
     SET items ;
     product= "when( '" !! trim(left(product)) !! "' ) product= ' '" ;
     run ;

     PROC PRINT data=items ;
       run ;

     DATA &newds;
       length product $32 ;
     SET sampsio.assocs ;


       select(product) ;
       when( 'apples' ) product= 'dresses' ;
       when( 'artichok' ) product= 'pants' ;
       when( 'avocado' ) product= 'belts';
       when( 'baguette' ) product= 'sweaters';
       when( 'bordeaux' ) product= 'socks';
       when( 'bourbon' ) product= 'ties';
       when( 'chicken' ) product= 'slippers';
       when( 'coke' ) product= 'gloves';
       when( 'corned_b' ) product= 'pajamas';
       when( 'cracker' ) product= 'pants';
       when( 'ham' ) product= 'coats';
       when( 'heineken' ) product= 'shirts';
       when( 'hering' ) product= 'blouses';
       when( 'ice_crea' ) product= 'hats';
       when( 'olives' ) product= 'parkas';
       when( 'peppers' ) product= 'jackets';
       when( 'sardines' ) product= 'umbrellas';
       when( 'soda' ) product= 'monograms';
       when( 'steak' ) product= 'accessories';
       when( 'turkey' ) product= 'miscellaneous';
      end ;

     customer_id= 'cxx' !! trim(left(put(customer,best.)));
     run ;

     PROC SORT data=&newds ;
       by customer_id ; run ;
```

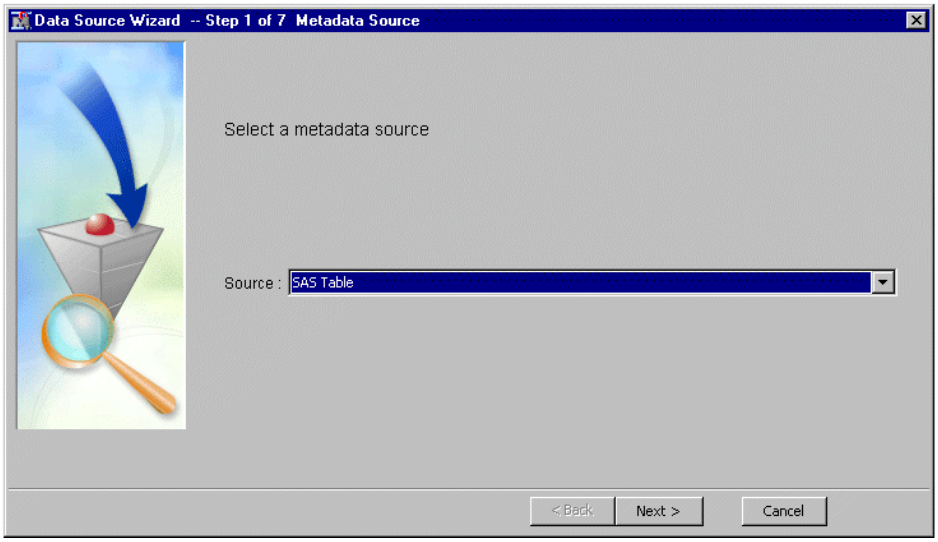
```
PROC FREQ data=&newds noprint ;
  table customer_id / out=clist ;
run ;

DATA sampsio.customers ;
merge sampsio.dmagecr
(in=a
 drop=good_bad purpose history
      amount savings housing depends
)
clist
(keep=customer_id) ;
if a then output ;
run ;
```

Create SAS Enterprise Miner Data Sources

You must associate the transaction and demographic data sets with your project by making them SAS Enterprise Miner Data Sources. In the Project Panel, right-click the Data Sources folder and select **Create Data Source**.



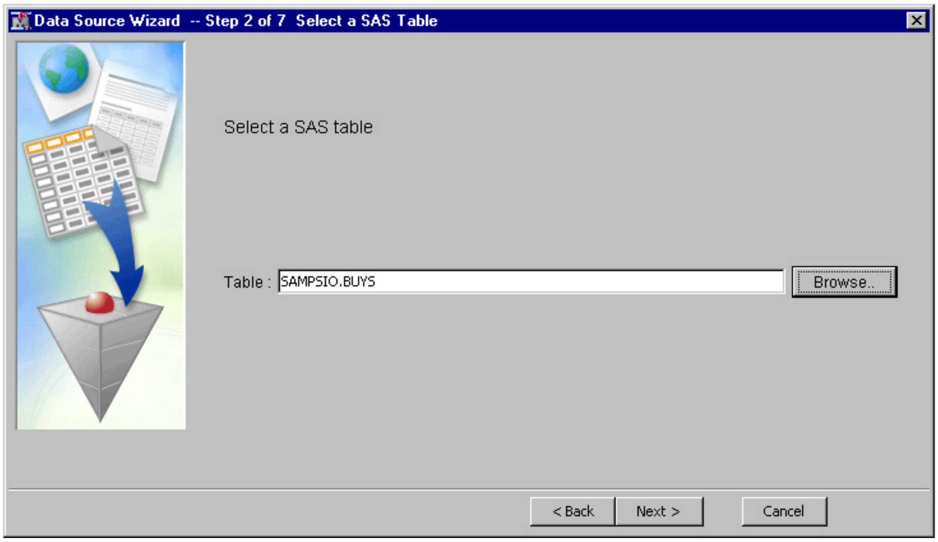
The Data Source Wizard opens to the Metadata Source page.



Select a metadata source

Source : SAS Table

The data sources are SAS tables. Click **Next** to continue to the Select a SAS Table window of the Data Source Wizard.

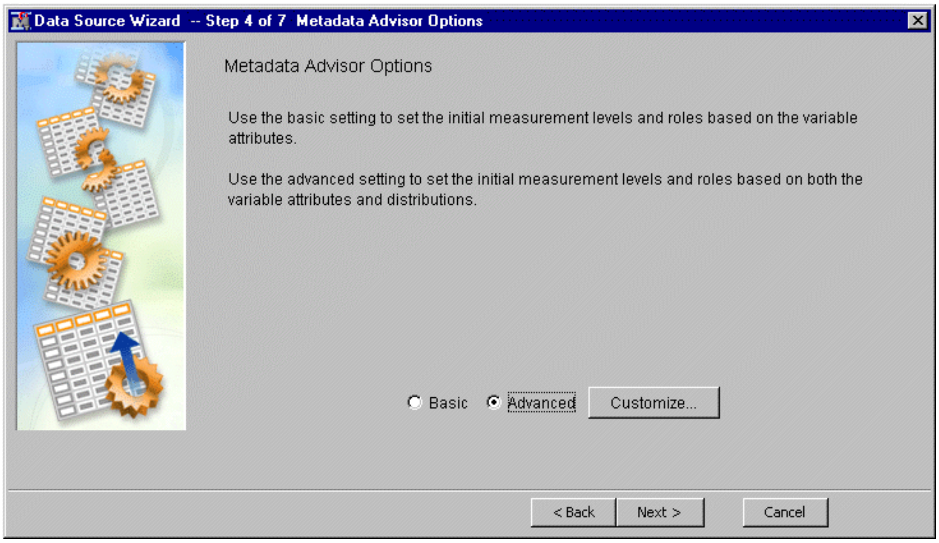

Data Source Wizard -- Step 2 of 7 Select a SAS Table

Select a SAS table

Table : SAMPSIO.BUYS

Type SAMPSIO.BUYS, the name of the transaction data set, in the **Table** box, or use **Browse** to open a file utility that you use to select the SAMPSIO library and the BUYS data set. After you populate the **Table** box, click **Next**. The Table Information window of the Data Source Wizard displays a data table of properties. Click **Next**.

The Metadata Advisor Options window of the Data Source Wizard opens.


Data Source Wizard -- Step 4 of 7 Metadata Advisor Options

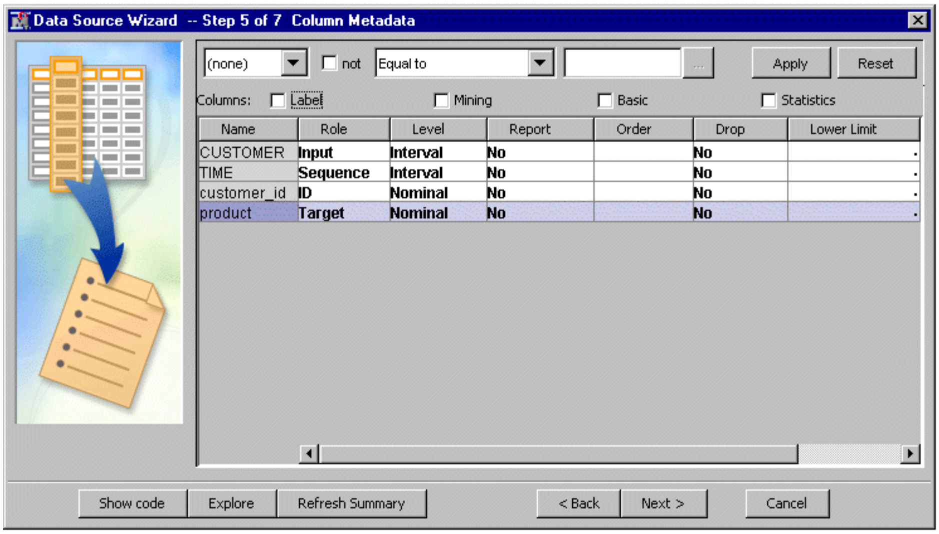
Metadata Advisor Options

Use the basic setting to set the initial measurement levels and roles based on the variable attributes.

Use the advanced setting to set the initial measurement levels and roles based on both the variable attributes and distributions.

○ Basic ● Advanced

Choose the **Advanced** advisor and click **Next**. The Columns Metadata page of the Data Source Wizard opens.



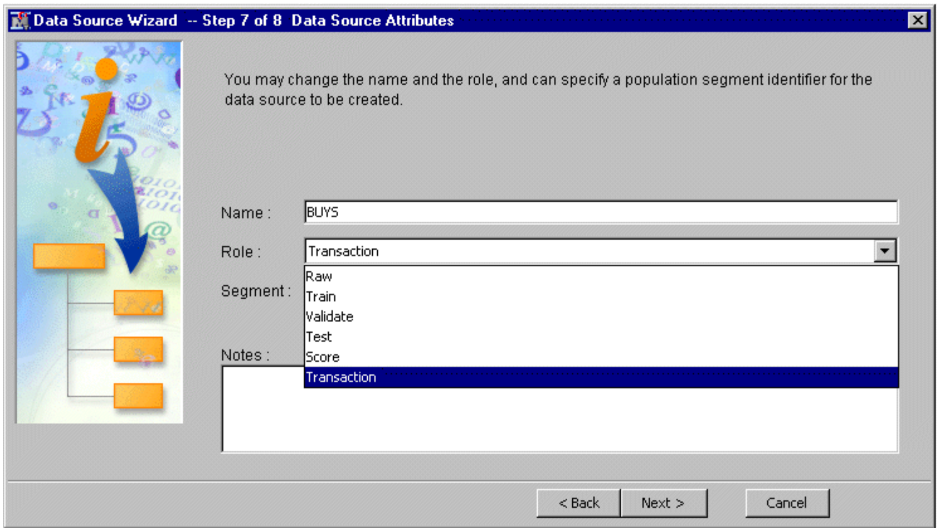
Data Source Wizard -- Step 5 of 7 Column Metadata

(none) □ not Equal to

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Role	Level	Report	Order	Drop	Lower Limit
CUSTOMER	Input	Interval	No		No	.
TIME	Sequence	Interval	No		No	.
customer_id	ID	Nominal	No		No	.
product	Target	Nominal	No		No	.

Set the level for the variable TIME to **Interval** and the role to **Sequence**. Set the role for the variable product to Target. Click **Next**. The Decision Configuration window of the Data Source Wizard opens. Leave **No** selected and click **Next**. The Data Source Attributes window of the Data Source Wizard opens.



Data Source Wizard -- Step 7 of 8 Data Source Attributes

You may change the name and the role, and can specify a population segment identifier for the data source to be created.

Name : BUYS

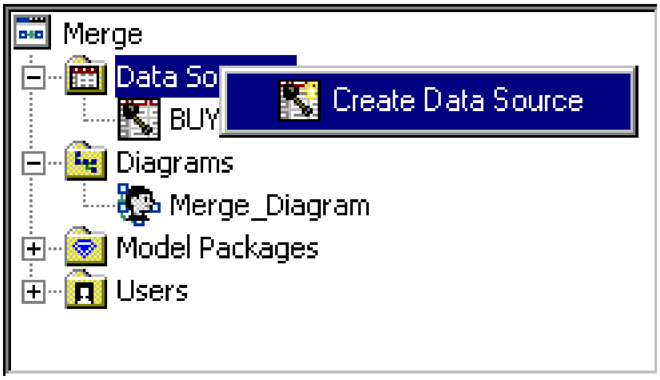
Role : Transaction

Segment :

Notes :

Select **Transaction** from the **Role** drop-down list, and then click **Next**. To complete the creation of the BUYS data source for your project, click **Finish**. Now the CUSTOMERS data source must be created.

Return to the Project Panel, right-click the Data Sources folder and select **Create Data Source**.



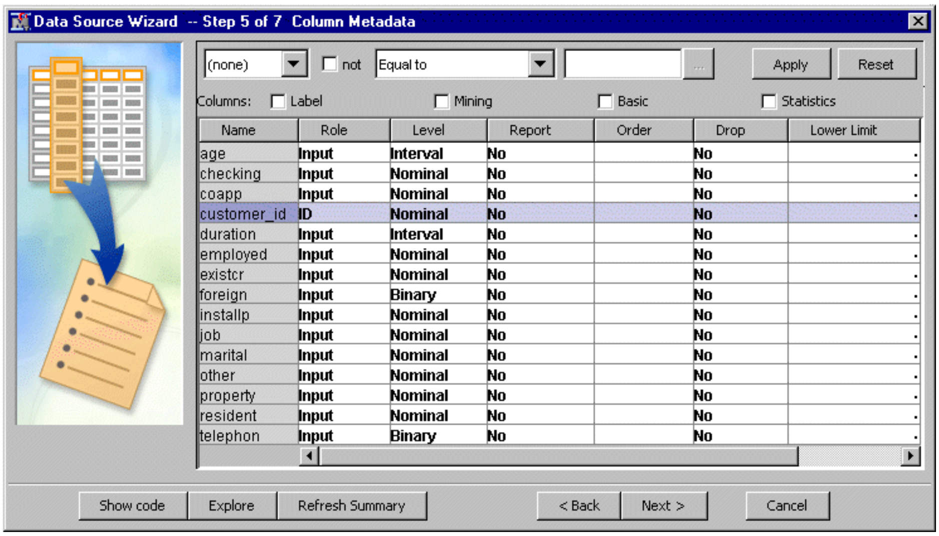
The Data Source Wizard opens to the Metadata Source page. Ensure that the **Source** is set to **SAS Table** and click **Next**.

The Select a SAS Table window of the Data Source Wizard opens. Type SAMPSIO.CUSTOMERS, the name of the demographic data set, in the **Table** box, or use **Browse** to open a file utility that you use to select the SAMPSIO library and the CUSTOMERS data set. After you populate the **Table** box, click **Next**.

The Table Information window of the Data Source Wizard displays a data table of properties. Click **Next**.

The Metadata Advisor Options window of the Data Source Wizard opens. Choose the **Advanced** advisor and click **Next**.

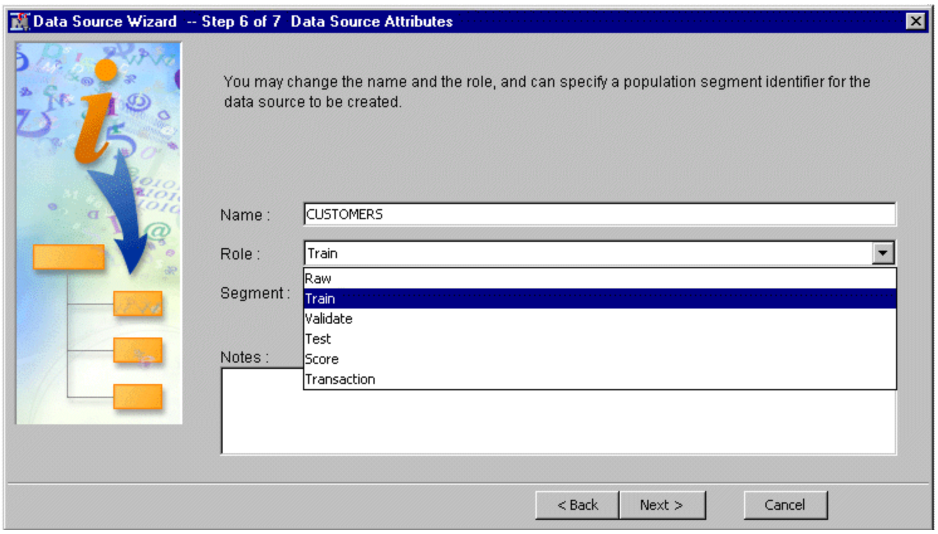
The Column Metadata page of the Data Source Wizard opens.


Data Source Wizard -- Step 5 of 7 Column Metadata
(none)
□ not
Equal to
Apply
Reset
Columns: □ Label □ Mining □ Basic □ Statistics
Show code
Explore
Refresh Summary
< Back
Next >
Cancel

Name	Role	Level	Report	Order	Drop	Lower Limit
age	Input	Interval	No		No	.
checking	Input	Nominal	No		No	.
coapp	Input	Nominal	No		No	.
customer_id	ID	Nominal	No		No	.
duration	Input	Interval	No		No	.
employed	Input	Nominal	No		No	.
existcr	Input	Nominal	No		No	.
foreign	Input	Binary	No		No	.
installp	Input	Nominal	No		No	.
job	Input	Nominal	No		No	.
marital	Input	Nominal	No		No	.
other	Input	Nominal	No		No	.
property	Input	Nominal	No		No	.
resident	Input	Nominal	No		No	.
telephon	Input	Binary	No		No	.

Confirm that the cCustomer_id variable from the DATA.CUSTOMERS data set is set to the **ID** role and that the Level is set to **Nominal**, and click **Next**.

The Data Source Attributes page of the Data Source Wizard opens.



You may change the name and the role, and can specify a population segment identifier for the data source to be created.

Name : CUSTOMERS

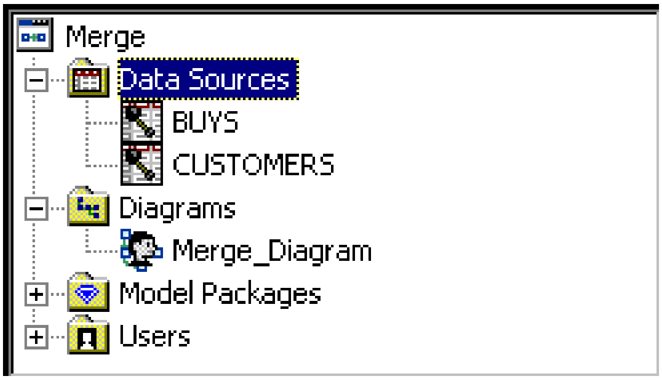
Role : Train

Segment :

Notes :

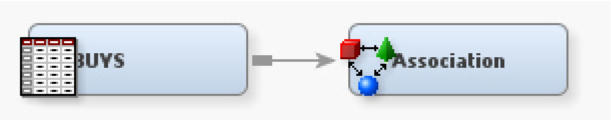
Set the Role for the CUSTOMERS demographic data set to **Train** and click **Next**. Click **Finish** to complete the creation of the customers data source.

You now should have entries BUYS and CUSTOMERS in Data Sources folder in your Project Panel.



Find Association Rules for the Transaction Data

Now we examine associations in the transaction data set. In your SAS Enterprise Miner Diagram Workspace, create the following process flow diagram:

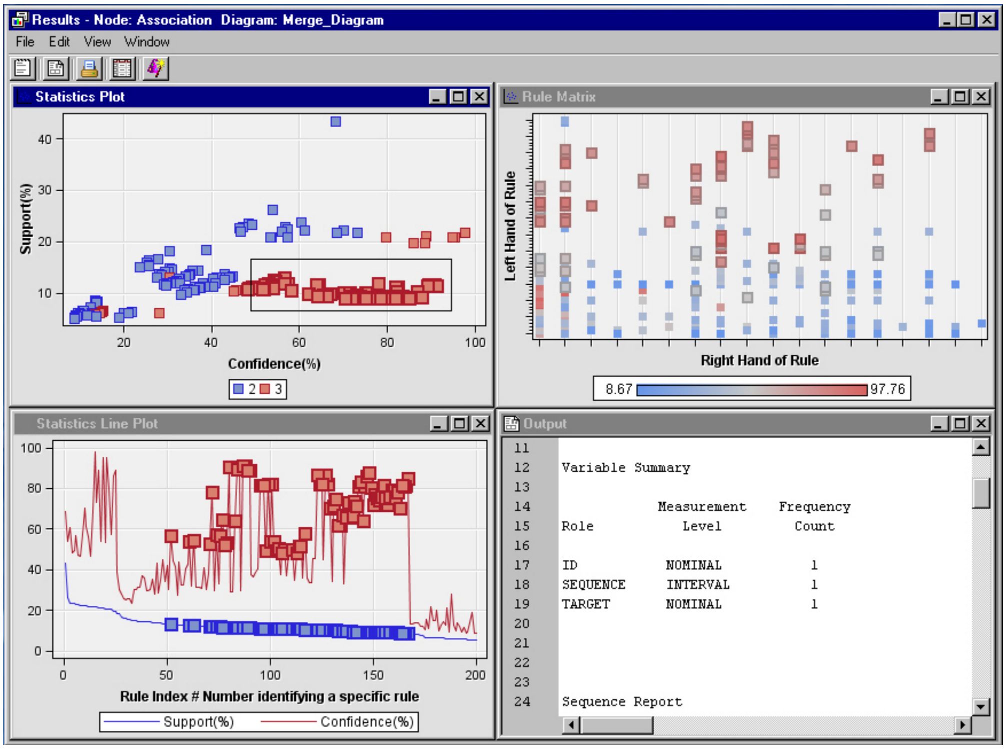


Select the **Association** node in the Diagram Workspace. In the properties panel for the **Association** node, in the Train properties, find the Rules property group. In the Rules property group, find and set the Export Rule by ID property to Yes in the Properties Panel. Leave all other properties for the **Association** node in their default settings.

Run the **Association** node by right-clicking the node in the process flow diagram and selecting Run from the pop-up menu.

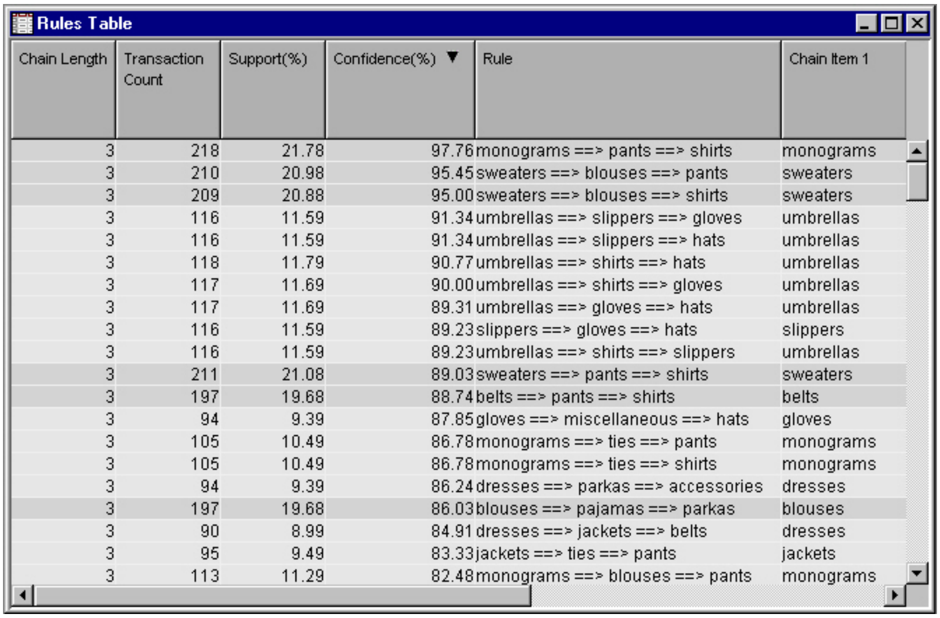
When the process flow diagram run completes, open the Association Results window.

We want to look for association rules that have relatively high confidence and relatively low support. In the Statistics Plot scatter plot, use your mouse pointer to draw a rectangle around the red markers for 3-item-chains that have confidence scores of greater than 50% and support scores of less than 15%. The rules that are associated with the selected scatter plot points will also be selected in the other Association charts.



```

Variable Summary


                Measurement     Frequency
Role               Level          Count


ID               NOMINAL            1
SEQUENCE         INTERVAL           1
TARGET           NOMINAL            1




Sequence Report
```

The **Association** node run saves two hundred rules. To see the rules, from the Results window menu, select **View** ⇨ **Rules** ⇨ **Rules Table**. The rules that were selected in the scatter plot will be selected in the Rules Table. You can see by sorting the columns for Confidence and Support that the selected rules all have confidence scores above 50% and support scores below 15%.

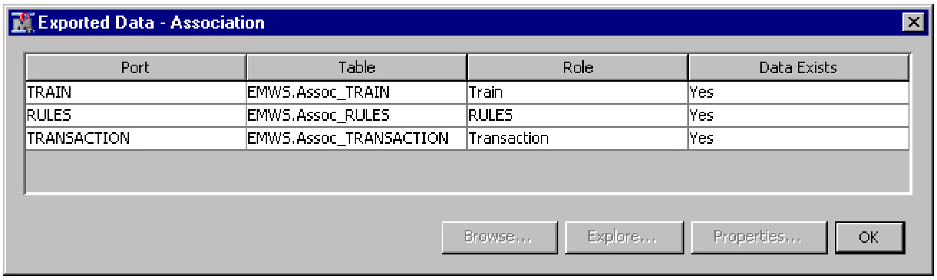


Chain Length	Transaction Count	Support(%)	Confidence(%) ▼	Rule	Chain Item 1
3	218	21.78	97.76	monograms ==> pants ==> shirts	monograms
3	210	20.98	95.45	sweaters ==> blouses ==> pants	sweaters
3	209	20.88	95.00	sweaters ==> blouses ==> shirts	sweaters
3	116	11.59	91.34	umbrellas ==> slippers ==> gloves	umbrellas
3	116	11.59	91.34	umbrellas ==> slippers ==> hats	umbrellas
3	118	11.79	90.77	umbrellas ==> shirts ==> hats	umbrellas
3	117	11.69	90.00	umbrellas ==> shirts ==> gloves	umbrellas
3	117	11.69	89.31	umbrellas ==> gloves ==> hats	umbrellas
3	116	11.59	89.23	slippers ==> gloves ==> hats	slippers
3	116	11.59	89.23	umbrellas ==> shirts ==> slippers	umbrellas
3	211	21.08	89.03	sweaters ==> pants ==> shirts	sweaters
3	197	19.68	88.74	belts ==> pants ==> shirts	belts
3	94	9.39	87.85	gloves ==> miscellaneous ==> hats	gloves
3	105	10.49	86.78	monograms ==> ties ==> pants	monograms
3	105	10.49	86.78	monograms ==> ties ==> shirts	monograms
3	94	9.39	86.24	dresses ==> parkas ==> accessories	dresses
3	197	19.68	86.03	blouses ==> pajamas ==> parkas	blouses
3	90	8.99	84.91	dresses ==> jackets ==> belts	dresses
3	95	9.49	83.33	jackets ==> ties ==> pants	jackets
3	113	11.29	82.48	monograms ==> blouses ==> pants	monograms

The output rules data set has columns for the left-hand and right-hand sides of the rule. The rules that are selected in the plots have relatively high confidence and relatively low support. These observations consist of sequences such as **umbrellas** ⇨ **slippers** ⇨ **gloves**. In sequences, the right-hand side of the rule is always the last item in the sequence chain. This helps us understand patterns in market baskets.

Examine the Association Output

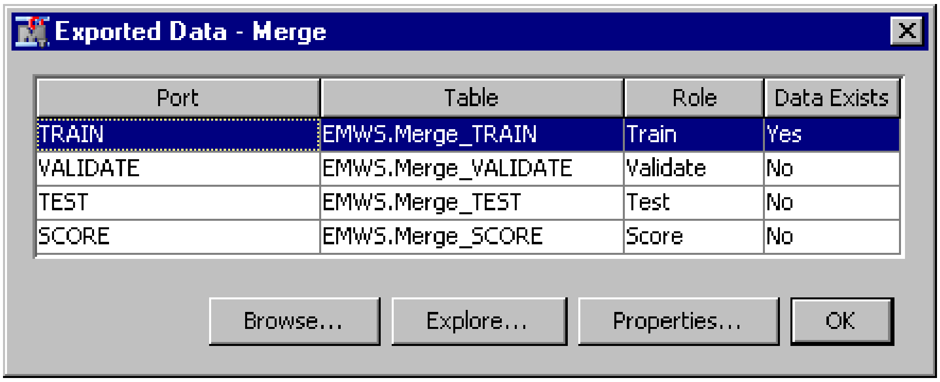
Close the **Association** node Results window and return to your process flow diagram. To view the Association node's output training data, verify first that the **Association** node is selected in the Diagram Workspace. Then, in the Association node Properties panel, select the ... button to the right of the Exported Data property to open the Exported Data — Association window.



Exported Data - Association

Port	Table	Role	Data Exists
TRAIN	EMWS.Assoc_TRAIN	Train	Yes
RULES	EMWS.Assoc_RULES	RULES	Yes
TRANSACTION	EMWS.Assoc_TRANSACTION	Transaction	Yes

Select the row for TRAIN and click **Explore** to open the training data set from the **Association** node.



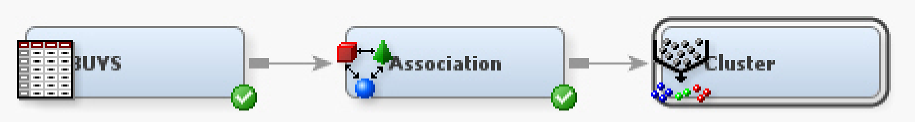
Exported Data - Merge

Port	Table	Role	Data Exists
TRAIN	EMWS.Merge_TRAIN	Train	Yes
VALIDATE	EMWS.Merge_VALIDATE	Validate	No
TEST	EMWS.Merge_TEST	Test	No
SCORE	EMWS.Merge_SCORE	Score	No

The Train data set is a new table that has one row per ID value (customer) and one column for each rule. These new columns are binary indicators with true values if the customer has executed this rule. The training data table can be used to model customers based on their transaction record.

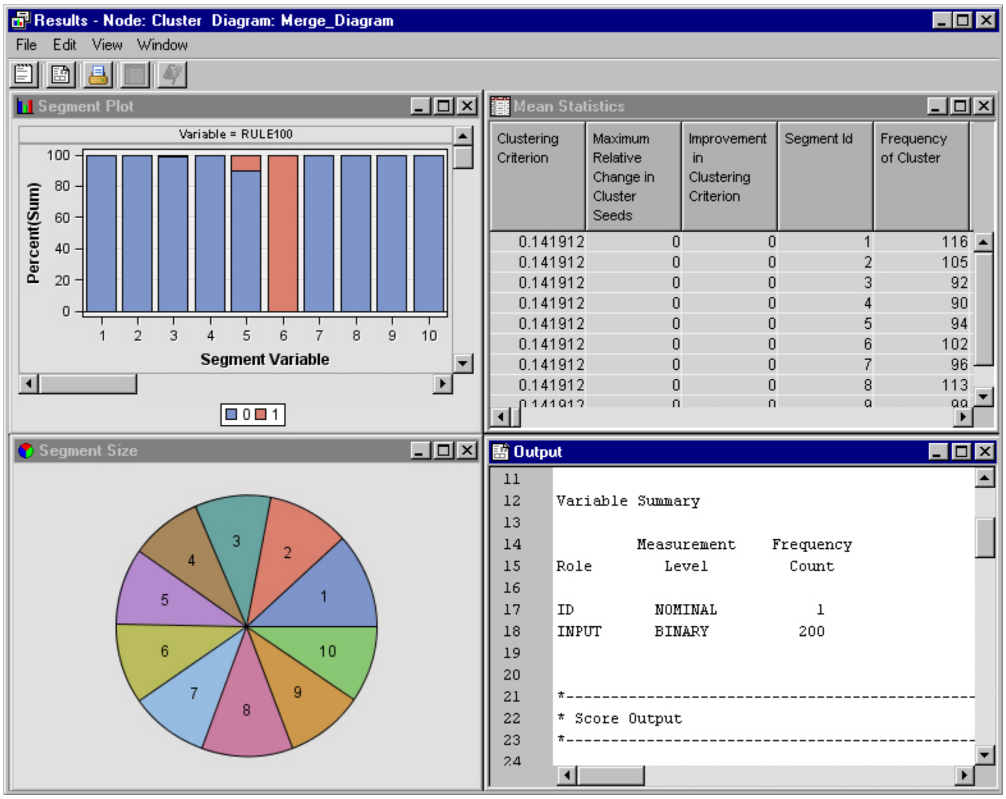
Cluster Individuals Based on Transactions

The training data set from the **Association** node can be used to cluster customers by their purchasing behavior as encoded in the rules variables. Add a Cluster node to your process flow diagram:



Select the **Cluster** node in the Diagram Workspace. Look in the properties panel for the **Cluster** node, and in the Train properties, locate the Number of Clusters property group, and set the Specification Method property to User Specify.

Run the **Cluster** node by right-clicking it in the Diagram Workspace and selecting Run from the pop-up menu. When the run completes, open the Cluster Results window.



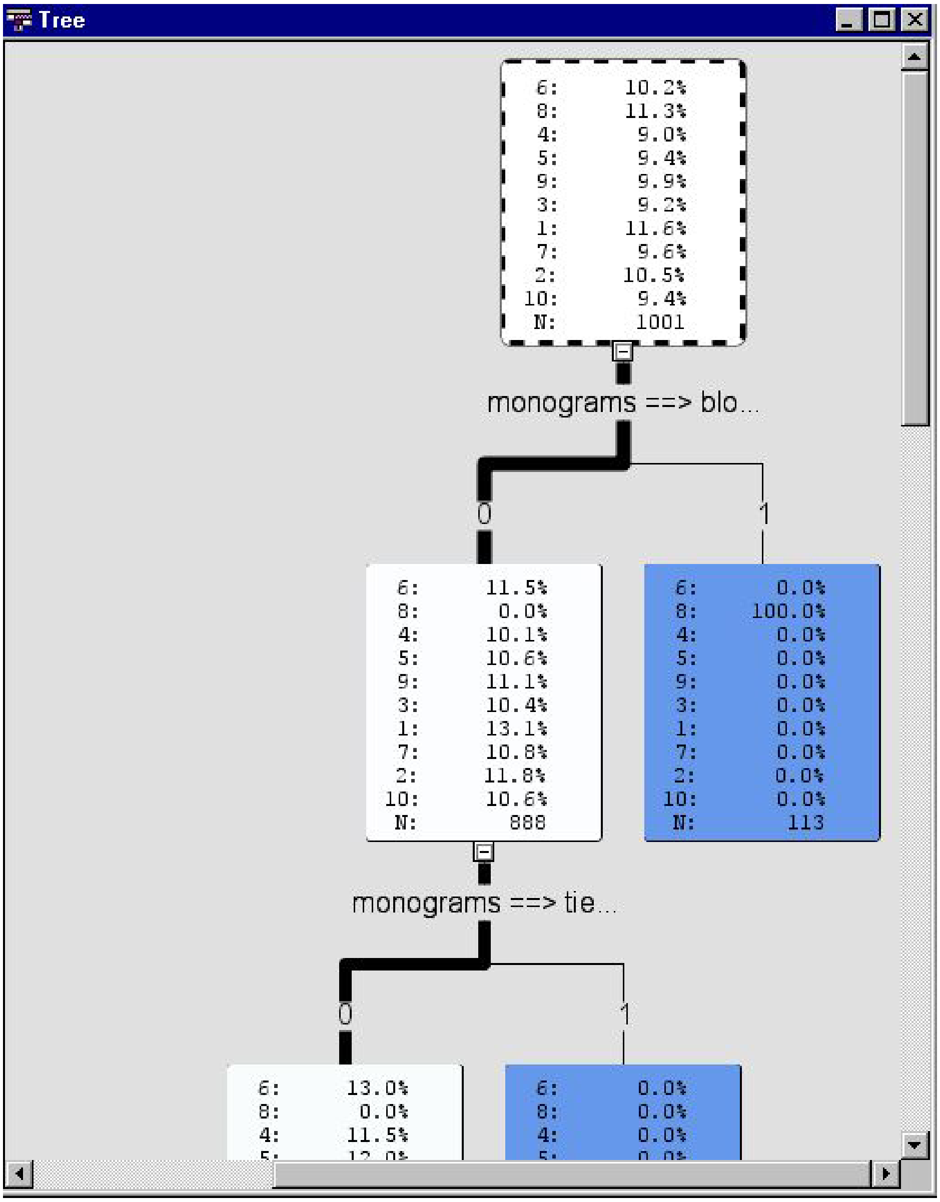
Clustering Criterion	Maximum Relative Change in Cluster Seeds	Improvement in Clustering Criterion	Segment Id	Frequency of Cluster
0.141912	0	0	1	116
0.141912	0	0	2	105
0.141912	0	0	3	92
0.141912	0	0	4	90
0.141912	0	0	5	94
0.141912	0	0	6	102
0.141912	0	0	7	96
0.141912	0	0	8	113
0.141912	0	0	9	99

```

Variable Summary


           Measurement    Frequency
Role          Level         Count

ID          NOMINAL           1
INPUT       BINARY          200


*------------------------------------------------
* Score Output
*------------------------------------------------

```

Ten clusters are produced. The segment plots show one plot for each significant rule and one bar for each cluster. Bars are colored by the value of the variable in the cluster. The rules variables are all binary so that the bar plots are only in two colors.

You can use a decision tree algorithm to investigate the overall variables' importance in discriminating the clusters. You can view the decision tree list in the Results Output window, or from the Clustering Results window main menu, select **View** ⇨ **Cluster Profile** ⇨ **Tree**.

The cluster profile Tree window appears. It is a convenient and explainable means of investigating the clusters.



```
6:      10.2%
8:      11.3%
4:       9.0%
5:       9.4%
9:       9.9%
3:       9.2%
1:      11.6%
7:       9.6%
2:      10.5%
10:      9.4%
N:      1001
```

```
6:      11.5%
8:       0.0%
4:      10.1%
5:      10.6%
9:      11.1%
3:      10.4%
1:      13.1%
7:      10.8%
2:      11.8%
10:     10.6%
N:       888
```

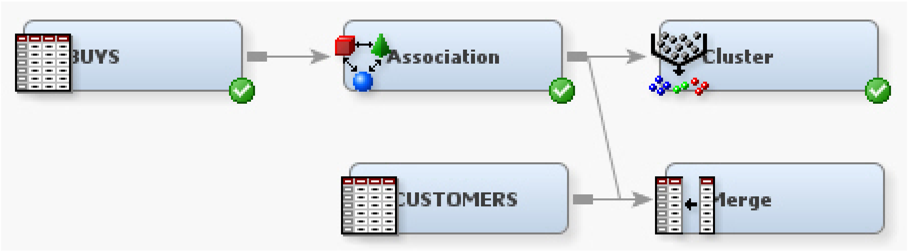
```
6:       0.0%
8:     100.0%
4:       0.0%
5:       0.0%
9:       0.0%
3:       0.0%
1:       0.0%
7:       0.0%
2:       0.0%
10:      0.0%
N:       113
```

```
6:      13.0%
8:       0.0%
4:      11.5%
```

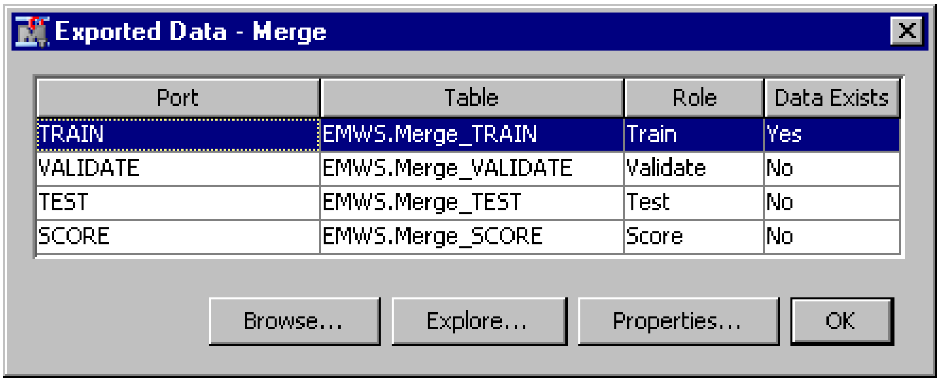
```
6:       0.0%
8:       0.0%
4:       0.0%
```

Merge the Demographic Data with the Transaction Data

Add the CUSTOMERS data source and the **Merge** node to your process flow diagram as shown below:

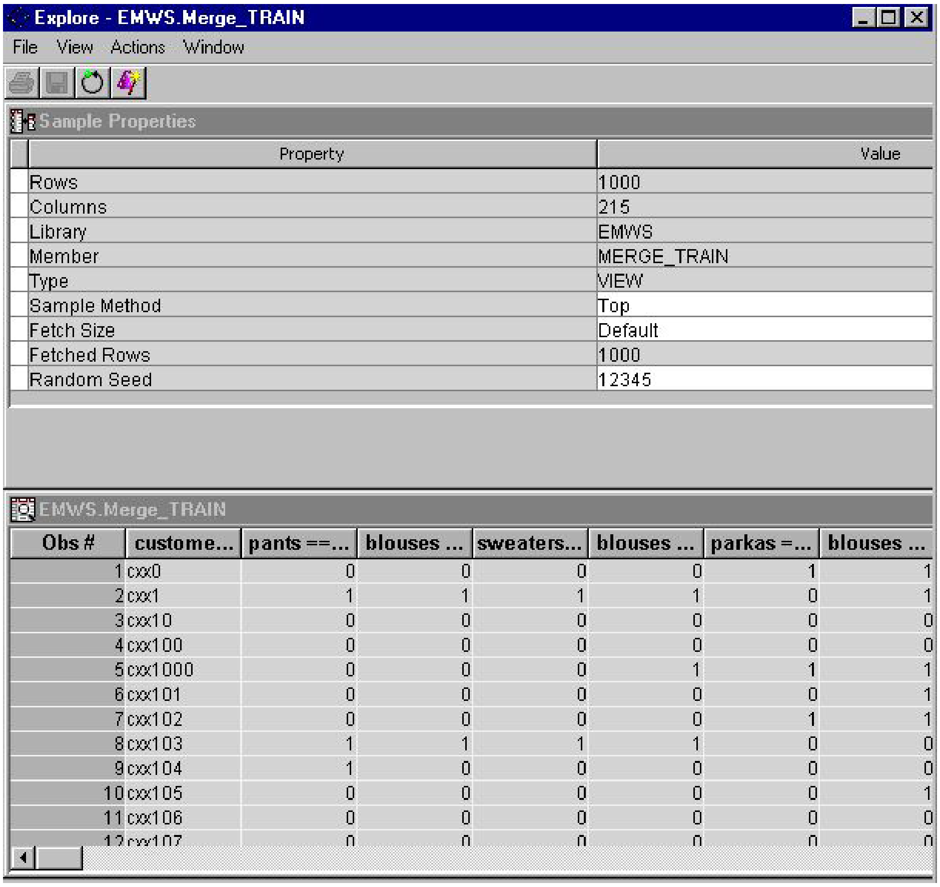


You can leave the **Merge** node in its default state. Because both the CUSTOMERS and BUYS (via Association) data tables contain an ID variable with the same name, the **Merge** node joins the two tables indexing on the Customer_ID field. Run the **Merge** node by right-clicking it in the Diagram Workspace and selecting **Run** from the pop-up menu. When the run completes, go to the Merge node Properties panel, and select the ... button to the right of the Exported Data property to open the Exported Data — Merge window.



Port	Table	Role	Data Exists
TRAIN	EMWS.Merge_TRAIN	Train	Yes
VALIDATE	EMWS.Merge_VALIDATE	Validate	No
TEST	EMWS.Merge_TEST	Test	No
SCORE	EMWS.Merge_SCORE	Score	No

Select the TRAIN row and click **Explore** to view the Merge training data. The resulting table has one row per customer ID with columns for both rules and demographics.



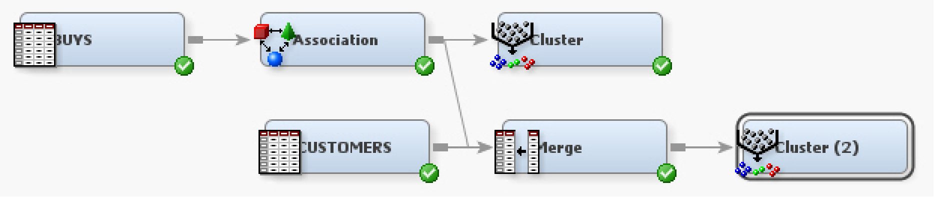
Property	Value
Rows	1000
Columns	215
Library	EMWS
Member	MERGE_TRAIN
Type	VIEW
Sample Method	Top
Fetch Size	Default
Fetched Rows	1000
Random Seed	12345

Obs #	custome...	pants ==...	blouses ...	sweaters...	blouses ...	parkas =...	blouses ...
1	cxx0	0	0	0	0	1	1
2	cxx1	1	1	1	1	0	1
3	cxx10	0	0	0	0	0	0
4	cxx100	0	0	0	0	0	0
5	cxx1000	0	0	0	1	1	1
6	cxx101	0	0	0	0	0	1
7	cxx102	0	0	0	0	1	1
8	cxx103	1	1	1	1	0	0
9	cxx104	1	0	0	0	0	0
10	cxx105	0	0	0	0	0	1
11	cxx106	0	0	0	0	0	0
12	cxx107	0	0	0	0	0	0

Now further analysis can be executed on both attributes; clustering by rules, and clustering by demographics.

Analyze Customers Based on Merged Data

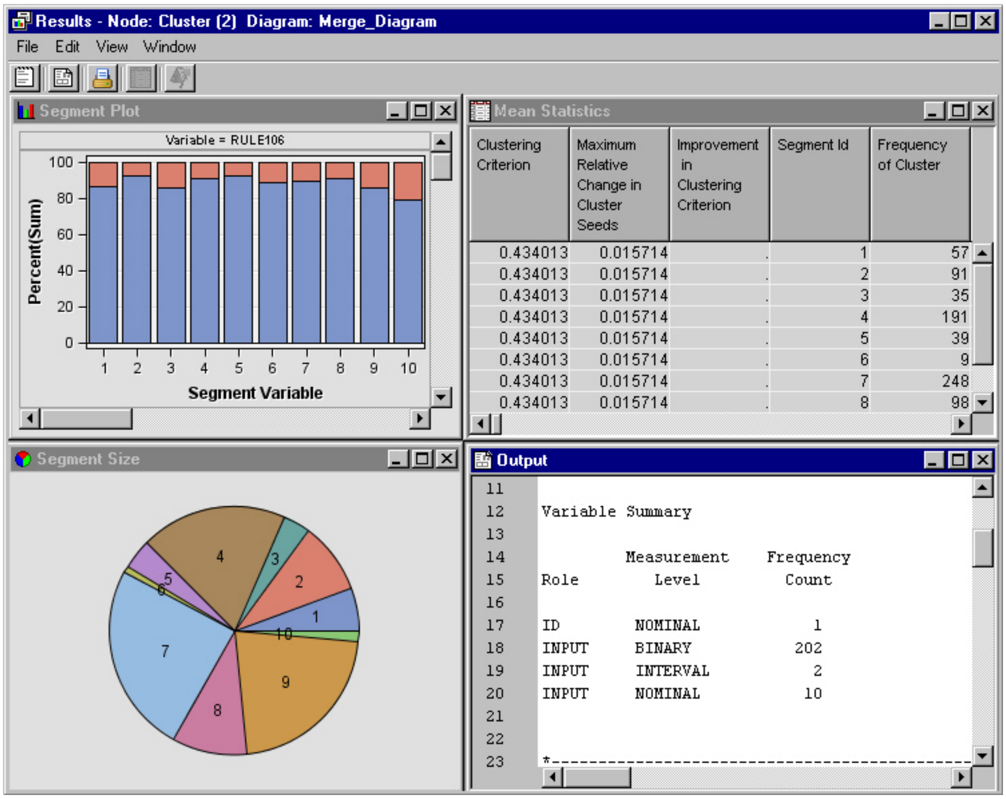
Add a second Cluster node to the process flow diagram, behind the **Merge** node as shown:



As with the previous Cluster node, set the Specification Method property to User Specify.

Run the **Cluster** node by right-clicking it in the Diagram Workspace and selecting **Run** from the pop-up menu. When the run completes, open the Cluster Results window.

The results show clusters that are based on both rules and demographics.



Clustering Criterion	Maximum Relative Change in Cluster Seeds	Improvement in Clustering Criterion	Segment Id	Frequency of Cluster
0.434013	0.015714	.	1	57
0.434013	0.015714	.	2	91
0.434013	0.015714	.	3	35
0.434013	0.015714	.	4	191
0.434013	0.015714	.	5	39
0.434013	0.015714	.	6	9
0.434013	0.015714	.	7	248
0.434013	0.015714	.	8	98

```
Variable Summary

           Measurement    Frequency
Role          Level         Count

ID          NOMINAL             1
INPUT       BINARY            202
INPUT       INTERVAL            2
INPUT       NOMINAL            10


*-----------------------------------------------
```

Chapter 27

Sample Node

Sample Node

Overview of the Sample Node

The Sample node is located in the Sample tab of the Enterprise Miner nodes toolbar.

The Sample node enables you to extract a sample from your input data source. Sampling is recommended for extremely large databases because it can tremendously decrease model fitting time. As long as the sample is sufficiently representative, patterns that appear in the data as a whole will be traceable in the sample. Sampling also closes the gap between huge data sets and human limitations.

The Sample node performs simple random sampling, nth-observation sampling, stratified sampling, first-n sampling, or cluster sampling of an input data set. For any type of sampling, you can specify either a number of observations or a percentage of the population to select for the sample. If you are working with rare events, the Sample node can be configured for oversampling, or stratified sampling. When you adjust the frequency for oversampling, the Sample node creates (or adjusts) a frequency variable with the sampling weights.

The Sample node writes the sampled observations to an output data set. The Sample node saves the seed values that are used to generate the random numbers for the samples so that you may replicate the samples. You can configure the Sample node to display samples in a view or in a data set.

The Sample node must be preceded by a node that exports at least one Raw, Train, Transaction, Document, Test, or Score data set. The Input Data node normally precedes the Sample node. If there is more than one predecessor data set, then the Sample node automatically selects one of the data sets for sampling. The other predecessor data sets are not exported to successor nodes in the process flow.

To partition the sample into training, validation, and test data sets, follow the Sample node with a Data Partition node. In general, any node can follow a Sample node.

Sample Node Properties

Sample Node General Properties

The following general properties are associated with the Sample node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Sample node that is added to a diagram will have a Node ID of Smpl. The second Sample node added to a diagram will have a Node ID of Smpl2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Sample window. The Imported Data — Sample window contains a list of the ports that provide data sources to the Sample node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Sample window. The Exported Data — Sample window contains a list of the output data ports that the Sample node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Sample Node Train Properties

The following train properties are associated with the Sample node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the Sample node. Select the ... button to open a window containing the variables table. You can set the Sample Role, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Output Type** — Use the Output Type property to indicate if the node should create data sets, or data step views.
- **Sample Method** — Use the Sample Method property to specify the Sample method that you want to use.

  You can choose from the following sampling methods:

  - **Default** — When the Default method is selected, if the target variable is a class variable, then the sample is stratified on the target variable. Otherwise, random sampling is performed.
  - **Cluster** — When Cluster sampling is selected, samples are drawn from a cluster of observations that are similar in some way. For example, a data miner might want to get all the records of each customer from a random sample of customers. If you select Cluster Sampling as your method, you must use the "Sample Node Variables Table" on page 394 to set the Sample Role on page 394 and specify a cluster variable. If you perform cluster sampling, the Sample node creates a data set. Cluster sampling cannot be used to create a data view.
  - **First N** — When First N sampling is selected, the first n observations are selected from the input data set for the sample. You must specify the quantity n either as a percentage or as an absolute number of observations, by using the respective properties Observations and Percentage. To enable these options, set the Type property to Number of Observations or Percentage.

    Note: The First N method of sampling can produce a sample that is not representative of the population, particularly if the input data set is not in random order.
  - **Random** — When random sampling is selected, each observation in the data set (population) has the same probability of being selected for the sample, independently of the other observations that happen to fall into the sample. For example, if observation 1345 is randomly drawn as the first member of the sample, we know that each member of the data set still has an equal chance of being the second member of the sample.
  - **Stratify** — When stratified sampling is selected, you choose nominal, binary, or ordinal variables from the input data set to form strata (or subsets) of the total population. Within each stratum, all observations have an equal probability of being selected for the sample. Across all strata, however, the observations in the input data set generally do not have equal probabilities of being selected for the sample. You perform stratified sampling to preserve the strata proportions of the population within the sample. This may improve the classification precision of fitted models.
  - **Systematic** — When systematic sampling is selected, the percentage of the population that is required for the sample is computed based on the number that you specify in the Observations or Percentage property. The tool divides 100 by the number in the Percentage property value field to come up with a number. The random sample process selects all observations that are multiples of this number for the sample.

    For example, if you specify a 5% sample from a population, then the random sample process selects observations that are multiples of 100/5=20. The node first randomly selects a start position from observations 1 through 20, and then it selects every nth observation from that position. For example, assume observation 10 is randomly selected as the start position. The tool then selects the remaining observations that are multiples of 20 (the 30th, 50th, 70th observations, and so on).

The systematic method of sampling can produce a sample that is not representative of the population, particularly if the input data set is not in random order. If there is a structure to the input data set, then the systematic sample may reflect only a part of that structure. For example, suppose the input data set contains information from 20 zip code regions, listed in the same order throughout the data set. In that case, a sample of every 20th observation would contain only the observations from a single zip code region.

- **Random Seed** — The Sample node displays the seed value used in the random number function for each sample. The default seed value is set to 12345. Type in a new seed directly to change the seed value. The Sample node saves the seed value that is used for each sample so that you can replicate samples exactly.

Sample Node Train Properties: Size

- **Type** — Use the Type property to specify the method you want to use to determine the sample size.
  - **Computed** — SAS computes the sample size that is required to capture rare events with the probability that you enter in the Pvalue field.
  - **Number of Observations** — the sample size is determined by the number that you enter in the Observations property value field.
  - **Percentage** — (default setting) the sample size is determined by the percentage number that you enter in the Percentage property value field.
- **Observations** — When the Type property is set to **Number of Observations**, use the Observations property to specify the sample size n, where n is the number of observations to use from the input data set. Permissible values are nonnegative integers.
- **Percentage** — When the Type property is set to **Percentage**, use the Percentage property to specify the sample size as a proportion of the input data set observations. Permissible values for the Percentage property are real numbers greater than zero. The default value for the Percentage property is 10.0.
- **Alpha** — When the Type property is set to **Computed**, use the Alpha property to specify the alpha value that you want to use when calculating the final number n of observations in a sample. Permissible values are nonnegative real numbers. The default alpha value is 0.01.
- **Pvalue** — When the Type property is set to **Computed**, use the Pvalue property to specify the p-value that you want to use when calculating the final number of observations for a sample. Permissible values are nonnegative real numbers. The default p-value is 0.01.
- **Cluster Method** — When the Sample Method property is set to Cluster, use the Cluster Method property to specify the cluster sample building method that you want to use.
  - **First N** — Using First N clusters sampling, the Sample node includes the first sequential n clusters that are associated with the specified cluster variable.
  - **Random** — (default setting) Using simple random cluster sampling, every cluster that is associated with the cluster variable has the same probability of being selected in the sample, independently of the other clusters that are associated with the same cluster variable. For example, if a specified variable cluster 1345 is randomly drawn as the first member of the sample, we know that

all other clusters associated with that variable still have an equal chance of being the second member of the sample.

- **Systematic** — Using systematic cluster sampling, the Sample node computes the percentage of the variable-specified clusters that are required for the sample. The Sample node selects for the sample all matching variable clusters that are multiples of this number.

Sample Node Train Properties: Stratified

- **Criterion** — Use the Criterion property to specify the sampling criterion that you want to use during stratified sampling.

  You can select from the following criteria:

  - **Proportional** — In proportional stratified sampling, the proportion of observations in each stratum is the same in the sample as it is in the population.
  - **Equal** — The equal property requires the Sample node to sample the same number of observations from each stratum. That is, the total sample size is divided by the number of strata to determine how many observations to sample from each stratum.
  - **Optimal** — With optimal allocation, both the proportion of observations within strata and the relative standard deviation of a specified variable within strata are the same in the sample as in the population. Usually, the within-strata standard deviations are computed for the target variable. When you select the Optimal stratified sampling Criterion, you must specify a Deviation Variable using the "Sample Node Variables Table" on page 394 .
  - **Level Based** — If Level Based is selected, then the sample is based on the proportion captured and sample proportion of a specific level. When the Criterion property is set to Level Based, use the Level Based Options properties to specify parameters for the proportion captured, sample proportion, and the level of interest.

- **Ignore Small Strata** — When the Sample Method property is set to Stratify, and the Ignore Small Strata property is also set to **Yes**, any stratum that has a population less than the value n that is specified in the Minimum Strata Size property is excluded from the sample. The default setting for the Ignore Small Strata property is **No**. This option is ignored when using the Level Based stratification criterion.
- **Minimum Strata Size** — When the Method property is set to **Stratify**, use the Minimum Strata Size property to specify the value n for the minimum number of observations that are required to construct a valid stratum. Permissible values are integers greater than or equal to 1. The default value for the Minimum Strata Size property is 5. This option is ignored when using the Level Based stratification criterion.

Sample Node Train Properties: Level Based Options

You can configure Level Based Options when the Criterion property in the Stratified properties group is set to Level Based. The Level Based properties specify the Level Based stratification criterion when a single stratification variable is used. If more than one stratification variable is used, the Level Based Options settings are ignored, and the Criterion property in the Stratified properties group is automatically set to Proportional.

- **Level Selection** — Use the Level Selection property to specify the level of interest. The available choices are **Event** and **Rarest Level**. If **Event** is selected, then the level is based on the variable ordering. The default ordering is ascending for input variables and descending for target variables.

- **Level Proportion** — Use the Level Proportion property to specify the proportion of the selected level of interest to be included in the sample.
- **Sample Proportion** — Use the Sample Proportion property to specify what proportion of the sample should contain the selected level of interest.

To further illustrate Level and Sample Proportion properties, consider performing level-based sampling from a 1000-observation training data source that has a binary target.

The training data source has the following binary target variable value distribution:

	# Obs.	% Obs.
TARGET=0	750	75
TARGET=1	250	25
Total	1000	100

If you create a sample using Level Based as your Criterion, and set Level Proportion = 60% and Sample Proportion = 25%, the created sample will have the following composition:

	# Obs.	% Obs.	Notes
TARGET=0	450	75	60% of 750 is 450 observations
TARGET=1	150	25	60% of 250 is 150 observations
Total	600	100	

You can see that 150 out of 250, or 60% of the observations with the selected level of interest (TARGET=1) from the data source are included in the sample. You can also see that 25 out of 100, or 25% of the observations in the sample contain the TARGET=1 event.

Using the same data source, suppose you create another level based sample and set Level Proportion = 10% and Sample Proportion 100%. The new created sample will have the following composition:

	# Obs.	% Obs.	Notes
TARGET=0	0	0	
TARGET=1	25	100	100% of 25 is 25 observations
Total	25	100	

You can see that 25 out of 250, or 10% of the observations with the selected level of interest (TARGET=1) from the data source are included in the sample. You can also see that 25 out of 25, or 100% of the observations in the sample contain the TARGET=1 event.

Sample Node Train Properties: Oversampling

The following properties configure Stratified Random Sampling (*oversampling*) properties. Oversampling is used most often to create models to predict rare events. Random sampling often does not provide enough targets to train a predictive model for rare events. Oversampling biases the sampling to provide enough target events to effectively train a predictive model.

- **Adjust Frequency** — Set the Adjust Frequency property to **Yes** to adjust the frequency for oversampling and create a biased stratified sample. The biased stratified sample uses a frequency variable that contains the sampling weights to adjust the target level proportions. If you configure the Sampling node for oversampling and no frequency variable exists, Enterprise Miner creates one. Oversampling is only performed when the Adjust Frequency option is set to **Yes**. The extent of the adjustment depends on how biased your sample is with respect to the input data source.
- **Based on Count** — Set the Based on Count property to **Yes** if you want to base your frequency variable adjustment on counts. When the Based on Count property is set to **No**, frequency variable adjustments are based on percentages. The default setting for the Based on Count property is **No**.
- **Exclude Missing Levels** — Set the Exclude Missing Levels property to **Yes** if you want to exclude strata where the stratification variables contain missing values. When Exclude Missing Levels is set to **Yes**, the frequency variable for excluded strata is set to 0. The default setting for the Exclude Missing Levels property is **No**.

Sample Node Report Properties

The following report properties are associated with the Sample node:

- **Interval Targets** — When set to **Yes**, the Interval Targets property generates summary statistics for the interval target variables based on the sample and training data.
- **Class Targets** — When set to **Yes**, the Class Targets property generates charts for the class target variables based on the sample and training data.

Sample Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Sample Node Variables Table

In the Variables window, use the table to specify the sample role of specific variables. You can choose the following roles: Cluster, Deviation, Stratification, None, and Default. See "Setting the Sample Role" on page 394 for more information. If a sampling role is dimmed and unavailable, then that sample role would not be applicable for that particular variable.

The following are read-only attributes:

- **Name** — displays the name of the variable
- **Label** — displays the text label to be used for variable in graphs and output
- **Role** — displays the variable role
- **Level** — displays the level for class variables
- **Type** — displays the variable type
- **Order** — displays the variable order
- **Report** — displays the Boolean setting for creating reports
- **Format** — displays the variable format
- **Informat** — displays the SAS Informat for the variable
- **Length** — displays the length of a variable
- **Lower Limit** — displays the minimum value for variable
- **Upper Limit** — displays the maximum value for variable
- **Creator** — displays the user who created the process flow diagram
- **Comment** — displays the user-supplied comments about a variable
- **Format Type** — displays the format type of a variable

Setting the Sample Role

If you choose cluster or stratify as the sampling method, then you can specify the variables upon which you want to cluster or stratify in the Variables table.

To set the Sampling role:

1. Click the ... button in the Variables property value field in the Properties panel to open the Variables table.
2. In the Variables table, click the Sample Role column of the row of the variable that you want to set.
3. From the pop-up menu, choose **Cluster**, **Deviation**, **Stratification**, **None** or **Default**. If a sampling role is dimmed and unavailable, then that sample role would be not be applicable for that particular variable.

Sample Node Results

You can open the Results window of the Sample node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
    - **Settings** — displays a window with a read-only table of the Sample node properties configuration when the node was last run.
    - **Run Status** — indicates the status of the Sample node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
    - **Variables** — displays a window with a read-only table of the Sample node variables by Sample Role, Name, Role, and Level.
    - **Train Code** — the code that Enterprise Miner used to train the node.
    - **Notes** — displays any user-created notes of interest.
- **SAS Results**
    - **Log** — the SAS log of the Filter node run.
    - **Output** — the SAS output of the sampling run. The output displays a variable summary and summary statistics for class and/or interval targets in the Sample node's output data set. The variables summary displays the variable roles, levels, and counts. The class and/or interval target summary statistics display the variables by value, formatted value, count, and percentage.
    - **Flow Code** — is not available in the Sample node.
- **Scoring**
    - **SAS Code** — is not available in the Sample node.
    - **PMML Code** — the Sample node does not generate PMML code.
- **Summary Statistics** — The summary statistics for class and interval target variables shows the distribution of levels for the sample and training data for class targets. The summary statistics also provide minimum, maximum, mean, and other related statistics for the sample and training data for interval targets. Summary statistics are only produced for variables where the Report properties are set to Yes.
    - **Interval Variables** — The Interval Variables summary uses the INTRVL data set to display a table of descriptive statistics based on the sample and original data for the interval target variables.

      The statistics are:
        - **Maximum** — the maximum value in the range for an interval target.
        - **Mean** — the arithmetic mean of the values of an interval target.
        - **Minimum** — the minimum value in the range for an interval target.
        - **Non Missing** — the number of non missing observations for an interval target.
        - **Missing** — the number of missing observations for an interval target.
        - Standard Deviation — the standard deviation of the values of an interval target.
    - **Class Variables** — displays bar charts that compare the distribution of the class target variables in the incoming data set with the sample data of the same variables.

- **Table** — displays a table that contains the underlying data used to produce the selected chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph Wizard to create ad-hoc plots based on the underlying data used to produce the selected table. The Graph Wizard menu item is dimmed and unavailable unless a Results chart or table is open and selected.

References

SAS Institute Inc 1998. *SAS Institute Best Practices Paper: “Data Mining and the Case for Sampling: Solving Business Problems Using SAS Enterprise Miner Software”*. Cary, NC: SAS Institute Inc.

Chapter 28
Time Series Node

Time Series Node

Overview of the Time Series Node

Over time, businesses collect large amounts of data related to the business activities and transactions that they conduct with their customers, suppliers, and monitoring devices. The **Time Series** node enables you to better understand trends and seasonal variations in time series data, such as the buying patterns of your customers, your buying patterns from your suppliers, or the ongoing performance of your equipment and machinery.

For example, you might have many suppliers and many customers, both with very large sets of associated transactional data. Traditional data mining tasks become difficult because of the size of the data set. By condensing the information into a time series, you can discover trends and seasonal variations in the data, such as customer and supplier habits that might not have been visible in the transactional data.

Transactional data is timestamped data that is collected over time at no particular frequency. By contrast, time series data is timestamped data that is collected over time at a specific frequency. The following table displays examples of transactional data, the data that results from conversion to time series data, and the analysis that might be done on the time series data.

Table 28.1 Transactional Data versus Time Series Data

Transactional Data	Time Series Data	Analysis
Internet data	Web hits per hour	Web hits by hour and by hour of day
Point of Sales (POS) data	Sales per month	Sales per month and by month of year
Inventory data	Inventory draws per week	Inventory draws per week and by week of month
Call Center data	Calls per day	Calls per day and by day of week
Trading data	Trades per weekday	Trades per weekday and by day or week

Time Series Node Data Set Requirements

Required Variable Types

The **Time Series** node requires that the following roles be assigned to variables in the data source:

- **Target** — You must define a single interval level target variable.
- **TimeID** — You must define a single numeric interval time ID variable. The time ID provides the time stamp information.

The following variable is optional:

- **CrossID** — you can define any number of cross ID variables. A cross ID variable must have a non-interval measurement. A cross ID variable represents a cross-sectional dimension to the time series data.

Required Data Set Role

The data set that you analyze with the **Time Series** node must have a data set role of **Transaction**.

Time Series Node Properties

Time Series Node General Properties

The following basic properties are associated with the **Time Series** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Time Series** node that is added to a diagram has a Node ID of TIME. The second **Time Series** node that is added to a diagram has a Node ID of TIME2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Time Series window. The Imported Data — Time Series window contains a list

of the ports that provide data sources to the **Time Series** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Time Series window. The Exported Data — Time Series window contains a list of the output data ports that the **Time Series** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Time Series Node Train Properties

The following train properties are associated with the **Time Series** node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **Time Series** node. Select the ... button to open a window that contains the variables table. You can set the Use property to **Yes**, **No**, or **Default**, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Accumulation** — specifies how the **Time Series** node accumulates time series data set observations within each time period.
  - **Total** — (default setting) observations are accumulated based on the total sum of their values.
  - **Average** — observations are accumulated based on the means of their values.
  - **Minimum** — observations are accumulated based on the smallest of their values.
  - **Median** — observations are accumulated based on the median of their values.
  - **Maximum** — observations are accumulated based on the largest of their values.
  - **First** — observations are accumulated based on their first values.
  - **Last** — observations are accumulated based on their last values.
- **Transformation** — specifies the time series transformation algorithm that you want the **Time Series** node to apply to your time series data.

- **None** — (default setting) no transformation is applied
- **Log** — logarithmic transformation
- **Square Root** — square-root transformation
- **Logistic** — logistic transformation
- **Box-Cox** — Box-Cox transformation

- **Box-Cox Parameter** — When the Transformation property of the **Time Series** node is set to **BoxCox**, use the Box-Cox Parameter property to specify the Box-Cox parameter value that you want to use. The permissible BoxCox parameter values are real numbers from -5.0 to 5.0, with a default of 0.0.
- **Apply Differencing** — When set to **Yes**, the Apply Differencing property of the **Time Series** node applies differencing to the accumulated time series. The default setting for the Apply Differencing property is **No**.
- **Difference Order** — When the Apply Differencing property of the **Time Series** node is set to **Yes**, use the Difference Order property to specify the order of differencing that is used. Permissible values are positive integers. The minimum and the default values of the Difference Order property are both 1.

Time Series Node Train Properties: Time Interval

- **Interval Selection** — specifies whether the **Time Series** node should automatically choose the time interval, or whether the user specifies the time interval.
  - **Automatic** — (default) the interval selection is defined by the node.
  - **User Specified** — the interval selection is defined by the values that you put in the Time Interval and Length of Cycle properties.
- **Specify an Interval** — When the Interval Selection property of the **Time Series** node is set to User Defined, you must use the Specify an Interval property to specify the SAS time interval.
  - **Year** — Yearly
  - **Semiyear** — Semiannual
  - **Quarter** — Quarterly
  - **Month** — Monthly
  - **Semimonth** — 1st and 16th of each month
  - **Ten Days** — 1st, 11th, and 21st of each month
  - **Week** — Weekly
  - **Weekday** — Daily, ignoring weekend days
  - **Day** — Daily
  - **Hour** — Hourly
  - **Minute** — Every Minute
  - **Second** — Every Second
- **Time of Day** — Set the Time of Day property of the **Time Series** node to Yes if you want to include the time of day in your Time Interval. The Time of Day property setting is ignored if the Time Interval property is set to Hour, Minute, or Second. The default setting for the Time of Day property is No.

- **Seasonal Cycle Selection** — Use the Seasonal Cycle Selection property of the **Time Series** node to specify the selection method of seasonal cycle. The resulting length of cycle is applied to the Seasonal, Trend, and Seasonal Decomposition analysis methods. When the Seasonal Cycle Selection property is set to User Specified, use the Length of Cycle property to specify a value. When the Seasonal Cycle Selection property is set to Default, the cycle length depends on the time interval:
  - **Year** — 1 Cycle
  - **Semiyear** — 2 Cycles
  - **Quarter** — 4 Cycles
  - **Month** — 12 Cycles
  - **Semimonth** — 24 Cycles
  - **Ten Days** — 36 Cycles
  - **Week** — 52 Cycles
  - **Weekday** — 5 Cycles
  - **Day** — 7 Cycles
  - **Hour** — 24 Cycles
  - **Minute** — 60 Cycles
  - **Second** — 60 Cycles
- **Length of Cycle** — Use the Length of Cycle property of the **Time Series** node to specify the seasonal cycle length if the Seasonal Cycle Selection property is set to User Specified. A valid value must be greater than 1.

Time Series Node Train Properties: Missing Value

- **Set Value** — Use the Set Value property of the **Time Series** node to specify how you want missing value assignments to be performed.

  The choices are as follows:
  - **Missing** — (default setting) missing values are not changed.
  - **Average** — missing values are imputed with the average value.
  - **Minimum** — missing values are imputed with the minimum value.
  - **Maximum** — missing values are imputed using the maximum value.
  - **Median** — missing values are imputed using the median value.
  - **First** — missing values are imputed using the first value in the time series.
  - **Last** — missing values are imputed using the last value in the time series.
  - **Previous Value** — missing values are imputed using the previous value in the time series.
  - **Next Value** — missing values are imputed using the next value in the time series.
  - **Constant** — missing values are imputed using a user-specified numeric constant. You must set the constant value in the Constant Value for Missing Observations property.
- **Constant Value for Missing Observations** — When the Set Value property of the **Time Series** node is set to **Constant**, use the Constant Value for Missing Observations property to specify the numerical value n that you want to use to

replace missing values. Permitted values are nonnegative real numbers. The default value is 0.0.

Time Series Node Train Properties: Analysis Method

- **Select an Analysis** — Use the Select an Analysis property of the **Time Series** node to specify the analysis method that the **Time Series** node uses to generate exported data.

  The choices are as follows:

  - **Seasonal**
  - **Trend**
  - **Correlation**
  - **Seasonal Decomposition**

- **Transpose** — Use the Transpose property of the **Time Series** node to transpose or suppress the transposing of the rows and columns of the exported data set. Transposed data sets are often suitable for classification. Setting the property to **Yes** transposes the rows and columns. Setting the Transpose property to **No** does not transpose data positions. The default setting for the Transpose property is **Yes**.

Time Series Node Train Properties: Seasonal Analysis

- **Exported Statistics** — Use the Exported Statistics property of the **Time Series** node to specify the season statistics to be exported to a successor node after a seasonal analysis. To perform seasonal analysis, the data must encapsulate more than one seasonal cycle.

  The following seasonal statistics are available:

  - **Sum** — (default setting) the season value is the sum of the values.
  - **Mean** — the season value is the mean of the values.
  - **Minimum** — the season value is the minimum value.
  - **Maximum** — the season value is the maximum value.
  - **Median** — the season value is the median value.

Time Series Node Train Properties: Trend Analysis

- **Exported Statistics** — Use the Exported Statistics property of the **Time Series** node to specify the trend statistic that is exported to a successor node after a trend analysis.

  The following trend statistics are available:

  - **Sum** — (default setting) the season value is the sum of the values.
  - **Mean** — the season value is the mean of the values.
  - **Minimum** — the season value is the minimum value.
  - **Maximum** — the season value is the maximum value.
  - **Median** — the season value is the median value.

- **Use Default Number of Time Periods** — When set to **No**, the Use Default Number of Time Periods property of the **Time Series** node suppresses the default number of time periods that is stored in the output data. When the Use Default Number of Time Periods property is set to **No**, you must specify the number of time periods that you want to use in the Number of Time Periods property. The default setting of the Use Default Number of Time Periods property is **Yes**.

- **Number of Time Periods** — When the Use Default Number of Time Periods property of the **Time Series** node is set to **No**, you must use the Number of Time Periods property to specify the number of time periods that you want to use for your trend analysis. The allowable values for the Number of Periods property are integers greater than 0. The default setting for the Number of Periods property is 24.

Time Series Node Train Properties: Correlation Analysis

- **Exported Statistics** — Use the Exported Statistics property of the **Time Series** node to specify the time domain statistic that you want to be exported to a successor node.

  The time domain statistic choices are as follows:

  - **Autocovariances**
  - **Autocorrelations**
  - **Normalized ACF** — Normalized Autocorrelations
  - **Partial Autocorrelations**
  - **Normalized PACF** — Normalized Partial Autocorrelations
  - **Inverse Autocorrelations**
  - **Normalized IACF** — Normalized Inverse Autocorrelations
  - **White Noise** — White Noise Test Statistics

- **Use Default Number of Lags** — When the Use Default Number of Lags property of the **Time Series** node is set to **No**, the default number of lags that are stored in the output data is overridden. You must use the Number of Lags property to specify the number of lag periods that you want your correlation analysis to use. The default setting for the Use Default Number of Lags property is **Yes**.

- **Number of Lags** — When the Use Default Number of Lags property of the **Time Series** node is set to **No**, the number of lag periods stored in the data is overridden. You must use the Number of Lags property to specify the number of lags that you want to use in your correlation analysis instead of the default lags stored in the data. If the Use Default Number of Lags property is set to **Yes**, the value in the Number of Lags setting is ignored. Permissible values for the Number of Lags property must be positive integers greater than or equal to 1. The default value is the lesser of 24 or three times the normal seasonal cycle.

Time Series Node Train Properties: Seasonal Decomposition

- **Exported Component** — Use the Exported Component setting of the **Time Series** node to specify which seasonal decomposition component you want to use to perform classic seasonal decomposition on your time series data.

  - **Trend-Cycle**
  - **Seasonal-Irregular**
  - **Seasonal**
  - **Trend Cycle Seasonal** — (default setting)
  - **Irregular**
  - **Seasonal Adjusted**
  - **Percent Change Seasonal Adjusted**

  - **Trend**
  - **Cycle**
- **Type** — Use the Type property of the **Time Series** node to specify the type of decomposition that you want to perform on your time series data.
  - **Default** — The default setting is **Additive or Multiplicative**
  - **Additive**
  - **Multiplicative**
  - **Log Additive**
  - **Pseudo Additive**
  - **Additive or Multiplicative**
- **Lambda** — Use the Lambda property of the **Time Series** node to specify the Hodrick-Prescott filter parameter that you want to use for trend-cycle decomposition. Permissible values are integers greater than 0. The default value for the Lambda property is 1600.
- **Use Default Number of Periods** — When set to **No**, the Use Default Number of Periods property of the **Time Series** node overrides the default number of periods specified between the Start and End properties. Instead, the number of periods that you specify in the Number of Time Periods property is used. The default setting for the Use Default Number of Periods property is **Yes**.
- **Number of Time Periods** — When the Use Default Number of Periods property of the **Time Series** node is set to **No**, set the Number of Time Periods property to specify the number of time periods to be stored in the decomposition output.

Time Series Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Time Series Node Results

You can open the Results window of the **Time Series** node after a successful run by clicking **Results** in the Run Status window, or by right-clicking the node in the diagram workspace and clicking **Results** on the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **Time Series** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **Time Series** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **Time Series** node run.
  - **Output** — the SAS output of the **Time Series** node run.
  - **Flow Code** — the SAS code that is used to produce the output that the **Time Series** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **Time Series** node does not generate PMML code.
- **Plots**
  - **Season** — displays plots from a Seasonal analysis. By default, the plots display the Seasonal Index as the category and the Sum as the response variable.
  - **Trend** — displays plots from a Trend analysis. By default, the plots display the Time Index as the category and the Sum as the response variable.
  - **Decomposition** — displays plots of the seasonal decomposition statistics. The plots use the Time Index variable as the category, and use the Time Series ID variable as the group variable.
  - **Correlation** — displays plots of the correlation statistics. The plots use the Time Lag variable as the category, and use the Time Series ID variable as the group variable.
- **Table** — displays a table that contains the underlying data that was used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Time Series Node Output Data Sources

Output Data Sources

Depending on the analysis method selected at run time, the **Time Series** node can export the following data sets:

- TIME_SEASON — the seasonal statistics for the data set.
- TIME_TREND — the trend statistics for the data set.
- TIME_OUTDS — the output training data set.
- TIME_CORRSTAT — the time domain statistics for the data set.
- TIME_DECOMP — the decomposition data set.

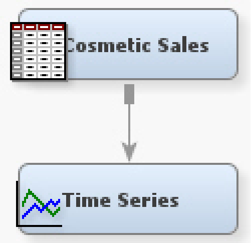
Viewing the Time Series Node Output Data Sources

You can view the Time Series output data sets through the Properties panel window after the node run is complete. Select the ... button to the right of the Exported Data property in the **Time Series** node Properties panel. This opens the Exported Data — Time Series window.

Time Series Node Example

Introduction

This example creates the process flow diagram shown below.



- “Create the Transaction Data” on page 406
- “Connect and Run the Time Series Node” on page 407
- “View the Time Series Model Results” on page 407

Create the Transaction Data

Create a data source that uses the SAS sample cosmetic sales data set.

1. From the main menu, select **File** ⇨ **New** ⇨ **Data Source**.
2. Select **SAS Table Import** as the import type and click **Next**.
3. Enter `SAMPSIO.COSMETIC` in the **Table** field. Click **Next**.
4. Click **Next** in the Table Information window.
5. Click **Advanced** in the Metadata Configuration window. Click **Next**.
6. In the Column Metadata table, assign a role of **Time ID** to the MNTH_YR variable, and change the level of the MNTH_YR variable from **Nominal** to **Interval**.

7. Assign a role of **Target** to the SALES variable. Make sure that the Level of the SALES variable is **Interval**.

8. Assign a role of **Cross ID** to the variables for SKU, GROUP, and STATE. Click **Next**.

9. In the Data Source Attributes window, enter `Cosmetic Sales` in the **Name** field. In the **Role** field, click the arrow and select **Transaction**. Click **Next**.

10. In the Summary window click **Finish**.

11. Expand the Data Sources folder in the Project Panel and drag the Cosmetic Sales data set to the Diagram Workspace.

Connect and Run the Time Series Node

1. Select the **Sample** tab of the SAS Enterprise Miner node toolbar, and drag a **Time Series** node onto the diagram workspace.

2. Connect the **Cosmetic Sales** node to the **Time Series** node.

3. Use the default property settings of the **Time Series** node.

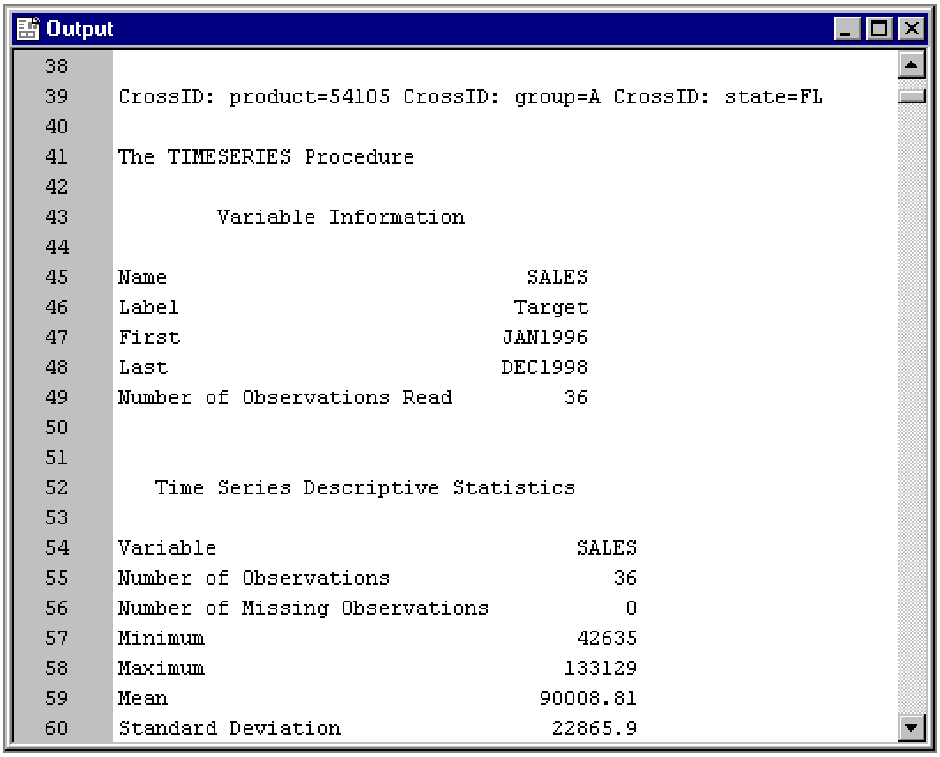
4. Right-click the **Time Series** node and select **Run**.

5. Click **Results** in the Run Status window after the node successfully runs.

View the Time Series Model Results

1. The Time Series Results window displays the seasonal plots and the Output window.

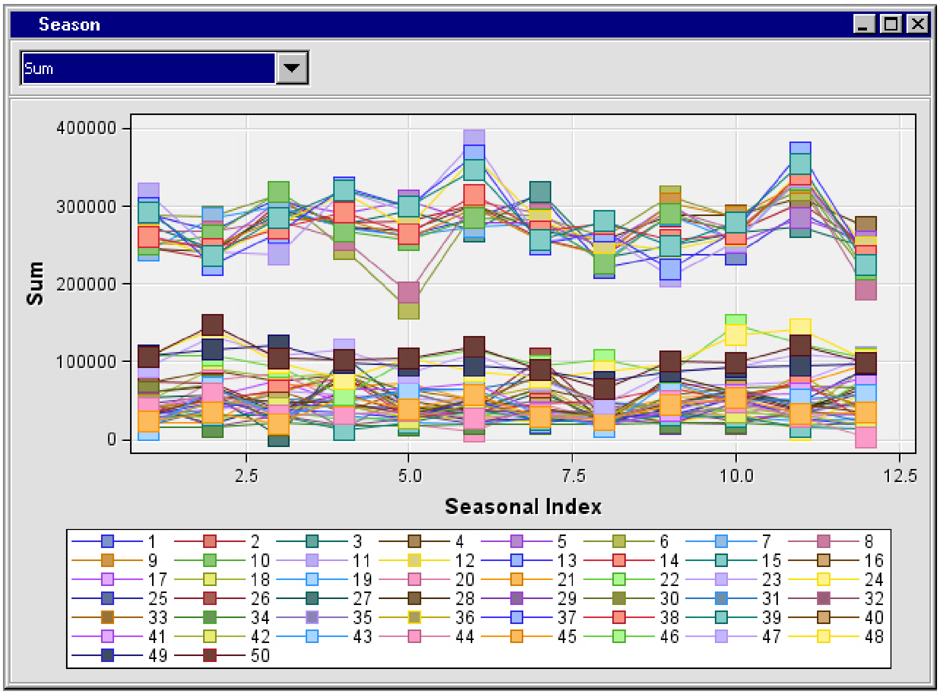
2. Expand the Output window. The Output window contains a summary of PROC TIMESERIES for each combination of the levels of the Cross ID variables. In this example, the Cross ID variables SKU, Group, and State have five, three, and five levels, respectively. Therefore, there are 5 * 3 * 5 = 75 combinations.



```

CrossID: product=54105 CrossID: group=A CrossID: state=FL

The TIMESERIES Procedure

        Variable Information

Name                             SALES
Label                           Target
First                          JAN1996
Last                           DEC1998
Number of Observations Read         36


   Time Series Descriptive Statistics

Variable                             SALES
Number of Observations                  36
Number of Missing Observations           0
Minimum                              42635
Maximum                             133129
Mean                              90008.81
Standard Deviation                 22865.9
```

3. Minimize the Output window.

4. Expand the Season window.



5. Right-click the Season plot and select **Data Options**.

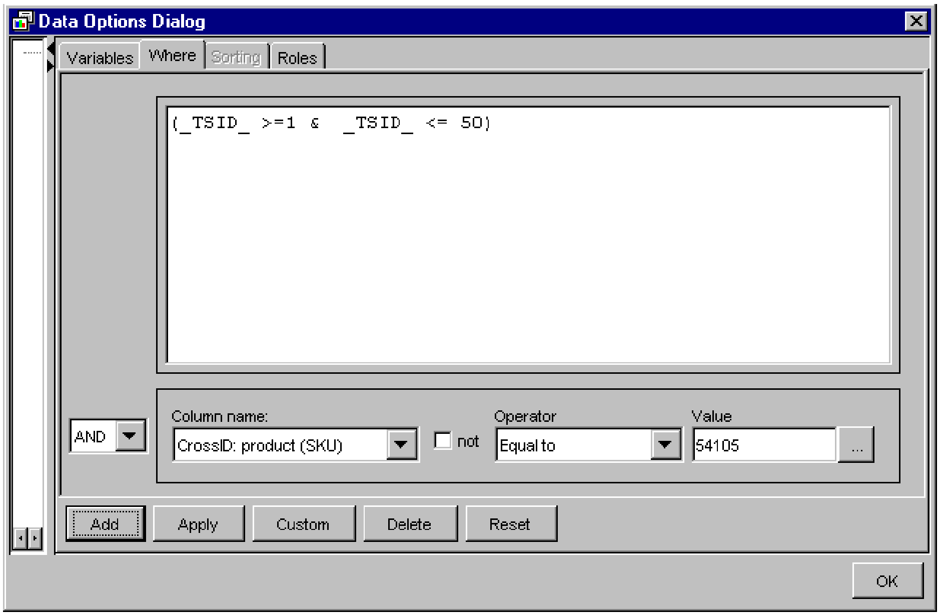
6. In the Data Options Dialog window, select the **Where** tab.

7. Click **Add** to display the code-generating lists.

8. Select **CrossID: product (SKU)** from the **Column name** drop-down list.

9. Select **Equal to** from the **Operator** drop-down list.

10. Open the **Value** field and select **54105** from the list. Click **OK**.



```
(_TSID_ >=1 &  _TSID_ <= 50)
```

Column name: CrossID: product (SKU)

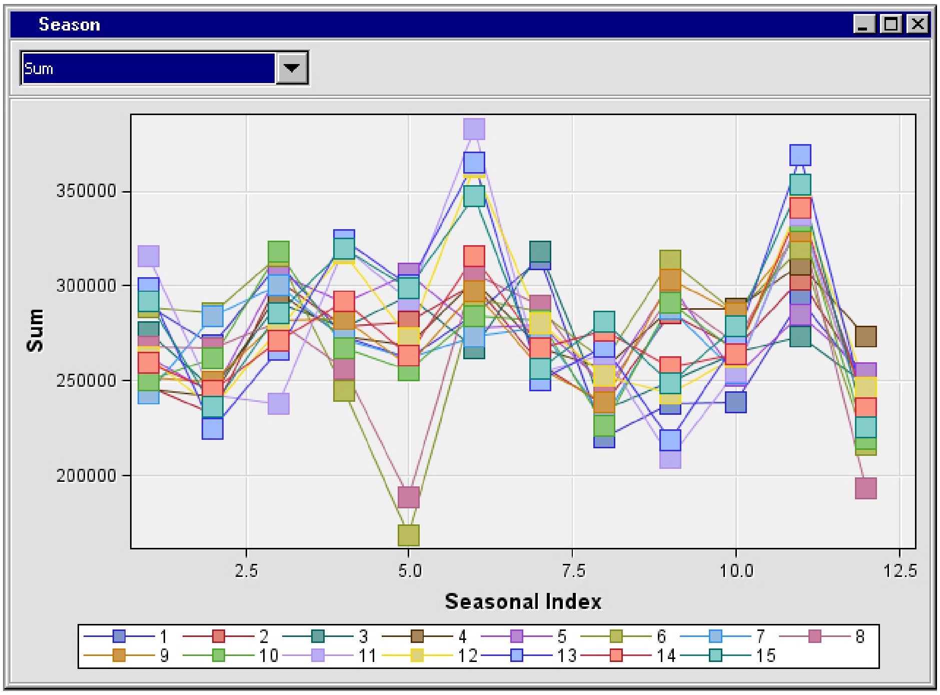
□ not

Operator: Equal to

Value: 54105

Click **Apply** and then click **OK**.

11. The resulting plot shows the time series that corresponds to SKU 54105, which was represented by the upper set of lines in the plot.


Season
Sum
350000
300000
250000
200000
Sum
2.5
5.0
7.5
10.0
12.5
Seasonal Index
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Part 11

Node Reference: Explore Nodes

Chapter 29
Association Node

Association Node

Overview of the Association Node

Association Discovery

The Association node belongs to the Explore category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA). You can use the Association node to perform association discovery and sequence discovery.

Association discovery is the identification of items that occur together in a given event or record. This technique is also known as market basket analysis. The databases behind online transaction processing systems often provide the data sources for association discovery. Association discovery rules are based on frequency counts of the number of times items occur alone and in combination in the transaction records. An association discovery rule can be expressed as "if item A is part of an event, then item B is also part of the event X percent of the time."

Associations can be written using the form A ==> B, where A (or the left hand side) is called the antecedent and B (the right hand side) is called the consequent. Both sides of an association rule can contain more than one item.

An example of an association rule might be, "If shoppers buy a jar of salsa, then they buy a bag of tortilla chips." In this example, the antecedent is, "buy a jar of salsa," and the consequent is, "buy a bag of tortilla chips." Association rules should not be interpreted as a direct causation.

Association rules define some affinity between two or more items. Association analysis does not create rules about repeating items, such as "if item A is part of an event, then another item A is also part of the event X percent of the time." In association analysis, it doesn't matter whether an individual customer buys one or multiple units of item A: only the presence of item A in the market basket is relevant. However, Identifying creditable associations between two or more different items can help the business technologist make decisions such as when to distribute coupons, when to put a product on sale, or how to present items in store displays.

Sequence Discovery

The Association node also enables you to perform sequence discovery. Sequence discovery goes one step further than association discovery by taking into account the ordering of the relationships among items (the rules additionally imply a timing element). For example, rule A ==> B implies that event B occurs after event A occurs.

Here are two hypothetical sequence rules.

- Of those customers who currently hold an equity index fund in their portfolio, 15% of them will open an international fund in the next year.
- Of those customers who purchase a new computer, 25% of them will purchase a laser printer in the next month.

Association Performance Measurements

Definitions

Association discovery uses the following performance measures to evaluate association rules:

Support — The level of support indicates how often the association combination occurs within the transaction database. In other words, support quantifies the probability of a transaction that contains both item A and item B. Support for the association rule A ==> B can be expressed mathematically as the ratio:

$$\frac{\textit{transactions that contain both items A and B}}{\textit{all transactions}}$$

Confidence — The strength of an association is defined by its confidence factor. Given the association rule A ==> B, the confidence for the association rule is the conditional probability that a transaction contains item B, given that the transaction already contains item A. Confidence for the association rule A ==> B can be expressed mathematically as the ratio:

$$\frac{\textit{transactions that contain both items A and B}}{\textit{transactions that contain item A}}$$

Expected Confidence — Given the association rule A ==> B, the expected confidence for the rule is the proportion of all transactions that contain item B. The difference between confidence and expected confidence is a measure of the change in predictive power due to the presence of item A in a transaction. Expected confidence provides an indication of what the confidence would be if there were no relationship between the items. Expected confidence for the association rule A ==> B can be expressed mathematically as the ratio:

$$\frac{transactions\ that\ contain\ item\ B}{all\ transactions}$$

Lift — Given the association rule A ==> B, the lift of the association rule is defined as the ratio of the rule's confidence to the rule's expected confidence. In other words, lift is the factor by which the confidence exceeds the expected confidence. Larger lift ratios tend to indicate more interesting association rules. The greater the lift, the greater the influence of an item A in a transaction has on the likelihood that item B will be contained in the transaction. Lift can be used as a general measure of the affinity that exists between the two items of interest.

A creditable rule has a large confidence factor, a large level of support, and a value of lift greater than 1. Rules that have a high level of confidence, but have little support should be interpreted with caution.

Association Performance Measurement Examples

To further illustrate these concepts, consider the following hypothetical information from a grocery store's transaction database.

Database Transaction Detail	Count
Total transactions in database	2000
Transactions containing the item "detergent"	55
Transactions containing the item "soda"	335
Transactions containing the item "chips"	450
Transactions containing the item "orange juice"	125
Transactions containing the items "detergent" and "orange juice"	35
Transactions containing the items "chips" and "soda"	150
Transactions containing the items "chips" and "detergent"	115
Transactions containing the item "chips" and "soda" and "orange juice"	15

Now, let's calculate performance measurement statistics for some association rules.

- What proportion of transactions contain both items "chips" and "soda"?
  - Support for the association rule "chips" ==> "soda" is the ratio of transactions that contain both chips and soda to all transactions: (150/2000) = 0.075 = 7.5% support.

    That means 7.5% of all customer transactions contain both chips and soda.
- What is the probability that a transaction that contains the item "chips" will also contain the item "soda"?

  - Confidence for the association rule "chips" ==> "soda" is the ratio of transactions that contain both chips and soda to transactions that contain chips: (150/450) = . 3333 = 33.33% confidence.

    That means that if a customer's transaction includes chips, there is a 33.33% chance that customer's transaction will also include soda.

- What is the expected confidence for the association rule "chips" ==> "soda"?
  - The expected confidence of the association rule "chips" ==> "soda" is equal to the ratio of transactions that contain soda to all transactions: (335/2000) = .1675 = 16.75% expected confidence.

    That means that 16.75% of all customer transactions are expected to contain soda, regardless of what other items are purchased.

- What is the lift for the association rule "chips" ==> "soda"?
  - The lift for the association rule "chips" ==> "soda" is the ratio of the confidence of the rule to the expected confidence of the rule: lift = (confidence/expected confidence) = (.3333/.1675) = 1.99

    That means that a customer whose transaction includes chips is 1.99 times more likely to also include soda in that same transaction, as compared to customers whose transactions does not include chips.

Association Node Data Set Requirements

Input Data Set Role

The data set that you analyze with the Association node must have a data role of **Transaction**.

Input Data Format

To perform association discovery, the input data set must have a separate observation for each product purchased by each customer, as illustrated in the following table. You must assign the **ID** role to one variable and the **target** model role to another variable when you create the data source.

Customer	Product
Skip Hops	Natural Choice Tomato Soup
Skip Hops	Washington Carver Honey Ale
Mary Lamb	Natural Choice Black Bean Soup
Mary Lamb	Otto's Oatmeal

To perform sequence discovery, the input data set must have a separate observation for each product purchased by each customer at each visit. In addition to assignment of ID and target roles, you must apply a **sequence** model role to a time stamp variable. The sequence variable is used for timing comparisons. It can have any numeric value including date/time values. The time or span from observation to observation in the input data set must be on the same scale.

Customer	Visit	Product
Skip Hops	1	Natural Choice Tomato Soup
Skip Hops	1	Brown Cow 2% Milk
Skip Hops	1	Washington Carver Honey Ale
Skip Hops	2	Lucky Dog Chow
Mary Lamb	1	Natural Choice Black Bean Soup
Mary Lamb	1	Brown Cow 2% Milk
Mary Lamb	2	Washington Carver Honey Ale
Mary Lamb	2	Otto's Oatmeal

The target variable can be a character variable or a non-interval numeric variable (binary, nominal, or ordinal).

If your data is not in this form, you can write a SAS DATA step or PROC TRANSPOSE step in the SAS Code node to create new a data set that contains a separate observation for each product purchased by each customer. See "Example 1: Writing SAS Code to Create Transaction Data" on page 432 for details.

Missing Values in the Association Node

For numeric target variables, missing values constitute a separate item or target level. They are listed in the rules as a period. For character target variables, completely blank values constitute a separate item. Missing character values are also listed in the rules as a period.

All observations with missing ID values are considered a single valid transaction. Transactions with missing sequence values are ignored entirely in a sequence analysis.

Managing Class Variables with a Large Number of Levels in the Association Node

Enterprise Miner is designed to handle class variables that have a large number of variable levels. In the past, analyzing data sets that contained class variables with very large numbers of levels could exhaust memory resources or could require very large amounts of processor time to execute. To handle such data sets gracefully, Enterprise Miner uses a SAS macro variable called EM_TRAIN_MAXLEVELS.

When the Association node processes a data set that contains variables with a very large number of classes or levels, it compares the number of levels in class variables to two values: the Association node class variable threshold value of 100,000, and the value stored in the macro variable EM_TRAIN_MAXLEVELS. If the number of levels in a class variable exceeds the greater of the two values, Enterprise Miner generates an error and halts the process flow.

To use the EM_TRAIN_MAXLEVELS macro variable to control variable levels in transaction data with the Association node, it must be set to some value greater than 100,000. To set the macro variable to a new value of 120,000, for example, type `%let EM_TRAIN_MAXLEVELS = 120000` in the SAS Program Editor and then submit the statement to SAS for processing. If you always want to change this default setting, you can include the EM_TRAIN_MAXLEVELS macro variable statement in your Enterprise Miner project start code.

Association Node Properties

Association Node General Properties

The following general properties are associated with the Association node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Association node added to a diagram will have a Node ID of Assoc. The second Association node added to a diagram will have a Node ID of Assoc2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Association window. The Imported Data — Association window contains a list of the ports that provide data sources to the Association node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Association window. The Exported Data — Association window contains a list of the output data ports that the Association node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Association Node Train Properties

The following train properties are associated with the Association node:

- **Variables** — specifies the properties of each variable in the data source that you want to use. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Yes**, **No**, or **Default** in the table.

- **Maximum Number of Items to Process** — Use the Maximum Number of Items to Process property to specify the upper limit on the number of items, or the maximum number of levels (items) of the target variable that you want to allow the Association node to process. Permissible values for the Maximum Number of Items to Process property are integers greater than 2. The default setting for Maximum Number of Items to Process is 100,000.

- **Rules** — Use the Rules property of the Association node to configure and transpose individual association rules in the Rules Selector window. The "Association Node Rules Selector" on page 431 window uses the Association node rules table. The Association rules table is created after a successful Association node run. The Rules Selector window cannot be opened until the node has been run. To open the Rules Selector window after a successful node run, select the Association node in the process flow diagram, then locate the Rules property in the Properties Panel. Select the ... button in the right column of the Rules property to open the Rules Selector window.

Association Node Train Properties: Association

- **Maximum Items** — Use the Maximum Items property to specify the maximum size of any given item set that you want to consider in an association. For the example, the default value of 4 items indicates that up to four-way associations are performed. The minimum number of Maximum Items is 1. Permissible values are non-negative integers. If you set Maximum Items to 1 and your data source does not contain a sequence variable, then Maximum Items is reset to 2.

- **Minimum Confidence Level** — Use the Minimum Confidence Level property to specify the minimum confidence level that you want to meet in order to generate a rule. The default level is 10%. If you are interested in rules that have a certain level of confidence, such as 50%, then it is to your advantage to set the minimum confidence to this level. Otherwise, the node can generate too many rules. Permissible values are real numbers between 0 and 100. The default frequency is 10%.

- **Support Type** — Use the Support Type property in the association group to specify whether the association analysis should use the count support or the percentage support. The default support type setting is **Percent**.

- **Support Count** — When the Association Support Type property is set to **Count**, use the Support Count property to specify the minimum level of support to claim that items are associated (that is, occur together in the database). Permissible values are integers between 0 and 100. The value in this field is expressed as a count value. Because the theoretical number of item sets can grow very fast, your system can run out of disk and/or memory resources. For example, with 50 different items, you have 1,225 potential 2-item sets and 19,600 3-item sets. With 5,000 items, you have over 12 million 2-item sets. You might want to set the support level to a higher number to reduce the item sets to a more manageable number. The default setting for the Support Count property is 1.

- **Support Percentage** — When the Support Type property is set to **Percentage**, use the Support Percentage property to specify the minimum level of support to claim that items are associated (that is, occur together in the database). Permissible values are real numbers between 0 and 100. The support percentage figure that you specify refers to the proportion of the largest single item frequency, and not the end support. The default frequency is 5%.

Association Node Train Properties: Sequence

- **Chain Count** — Use the Chain Count property to specify the maximum number of items that can be included in a sequence. Permissible values are positive integers between 2 and 10. The default value is 3. If you specify a number for the Chain Count property that is larger than any chain that is found in the data, then the chain length in the results will be smaller than the value you specified.
- **Consolidate Time** — Use the Consolidate Time property to specify whether consecutive visits to a location over a given interval can be consolidated into a single visit for analysis purposes. For example, a customer goes to the store at 7:00 AM to buy eggs and bacon (the unit of measure is an hour). She returns to the store at 1:00 PM to buy several other items. She makes an additional visit at 9:00 PM to buy cold medicine and a box of tissues. She returns to the store two days later at noon to buy orange juice, chicken noodle soup, and more cold medicine. In this example, the sequence variable (VISIT) has 4 entries. If you want to perform sequence discovery on a daily basis, then you would need to enter 24 hours in the Consolidate Time property. This would enable multiple visits on the same day to be consolidated as a single visit. Permissible Consolidate Time values are real numbers larger than 0.
- **Maximum Transaction Duration** — Use the Maximum Transaction Duration property to define the maximum transaction window length. By default, all possible sequences are identified. If you want to be restricted to a particular time limit, you can specify the transaction window length; for example, printers bought within three months after a PC purchase. Permissible Maximum Duration property entries are real numbers larger than 0.
- **Support Type** — Use the Support Type property in the sequence group to specify whether the sequence analysis should use the count support or the percentage support. The default support setting is **percentage support**.
- **Support Count** — When the Sequence Support Type property is set to **Count**, use the Support Count property to specify the minimum level of support for a sequence in the sequence analysis. If a sequence in the database has a count that is less than the specified value, then that sequence is excluded from the output data set. The default setting for the Support Count property is 1.
- **Support Percentage** — When the Sequence Support Type property is set to **Percentage**, use the Support Percentage property to specify the minimum level of support for a sequence in the sequence analysis. If a sequence has a frequency that is less than the specified percentage of the total number of transactions in the database, then that sequence is excluded from the output data set. Permissible values are real numbers between 0 and 100. The default frequency is 2%.

Association Node Train Properties: Rules

- **Number to Keep** — Use the Number to Keep property to define the maximum number of rules that you want to keep for the results. The default setting is 200 rules. Choices include settings for 50, 100, 200, 500, 1,000, 2,000, 5,000, and 10,000 rules.
- **Sort Criterion** — Use the Sort Criterion property to specify the statistic that you want to use to sort the generated rules. The default sort criterion for an association analysis is the lift statistic. The default sort criterion for a sequence analysis is the support statistic. The available statistic settings are **Default**, **Confidence**, **Count**, **Expected Confidence**, **Lift**, and **Support**.
- **Number to Transpose** — Use the Number to Transpose property to control the number of rules that will be transposed and used to create the train data set (Rule By ID data set). The default setting is 200 rules. Choices include settings for 50, 100, 200, 500, 1,000, 2,000, and 5,000 rules.

- **Export Rule by ID** — Set the Export Rule by ID property to **Yes** if you want to export the Rule-by-ID Data and to display the "Rule Description Table" on page 427 in the Results window. The default setting for this property is **No**.
- **Recommendation** — Use the Recommendation property to specify that scoring is done only for recommendations. If you want to determine which rules are met for a given ID, then you should not use this property. The default setting is **No**.

Association Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Sequence Analysis Settings That Affect Computing Resources

The SAS algorithm behind the Associations node is active during both association analysis and sequence analysis. Before a sequence analysis can be performed, the PROC ASSOC algorithm performs a combinatorial analysis of the submitted variables, and acts as a filter to generate the item set, also called the candidates list. The item set is a list of the combinations of items that will be used during sequence analysis.

When specifying Association node properties for a sequence analysis model, support properties should be configured in both the Association group and the Sequence group of the Association node Train properties. To overcome computing resource constraints, some sequence analysis models that have large numbers of combinations may need to increase the values of the default support levels in the Association property settings group.

The Association support and Sequence support property settings are interrelated. When you increase the minimum support for the associations analysis, there will be less candidates to consider for the sequence analysis.

Association Node Results

General Information

You can open the Results window of the Association node by selecting the Results button in the Run window that appears immediately after a successful Association node run. You can also right-click the Association node in the process flow diagram and select **Results**. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

The plots in the results of association and sequence analyses are based on a subset of the association rules that were created.

Association Node Results Window Main Menu

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Association node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Association node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — the code that Enterprise Miner used to train the node.
- **SAS Results**
  - **Log** — the SAS log of the Association node run.
  - **Output** — the SAS output of the Association node run. The output displays the summary statistics for the original partitioned data set and for each resulting partition.
  - **Flow Code** — the SAS code used to produce the output that the Association node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Rules**
  - **Rule Matrix** — opens the “Rule Matrix and Rules Table” on page 423 window.
  - **Statistics Line Plot** — opens the “Statistics Line Plot” on page 424 window.
  - **Statistics Plot** — opens the “Statistics Plot” on page 425 window.
  - **Confidence Plot** — opens the “Confidence Plot” on page 425 window for an association analysis. This menu item is not available for a sequence analysis.
  - **Rule Description** — opens the “Rule Description Table” on page 427 window.
  - **Rules Table** — opens the “Rules Table” on page 426 that contains information about the association rules that are generated. The contents of the Rules table is similar to the Rules Output Data Set.
  - **Link Graph** — opens the “Link Graph” on page 428 window.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard that you can use to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Results Graphs and Tables

General Information

The Association node Results window produces the following graphic plots and tables that are useful for Association results interpretation.

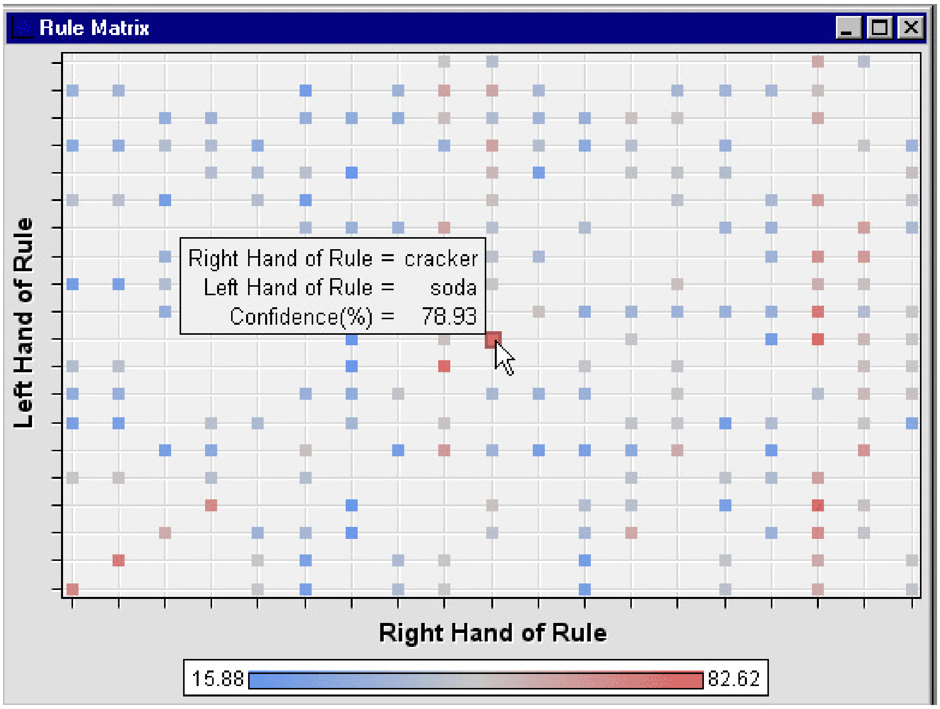
- "Rule Matrix and Rules Table" on page 423
- "Statistics Line Plot" on page 424
- "Statistics Plot" on page 425
- "Confidence Plot" on page 425
- "Rule Description Table" on page 427
- "Rules Table" on page 426
- "Link Graph" on page 428

You can modify and enhance Results graph attributes, whether you want to graphically explore effects of different variable roles, add or change response variables, or modify graph attributes to emphasize particular results or enhance appearance for presentation purposes.

For more information about manipulating plot variable roles and response variables, see the Data Options Dialog on page 261 topic in the Enterprise Miner User Interface Help. For more information about modifying individual plot attributes, see the section on the Graph Properties window on page 263.

Rule Matrix and Rules Table

Association rules are made up of a left-hand item and one or more right-hand items. The left-hand item is called the Left Hand of Rule. The right-hand items are called the Right Hand of Rule. The rule matrix displays a grid plot of the items in the Left Hand of Rule on the Y-axis and the items in the Right Hand of Rule on the X-axis.



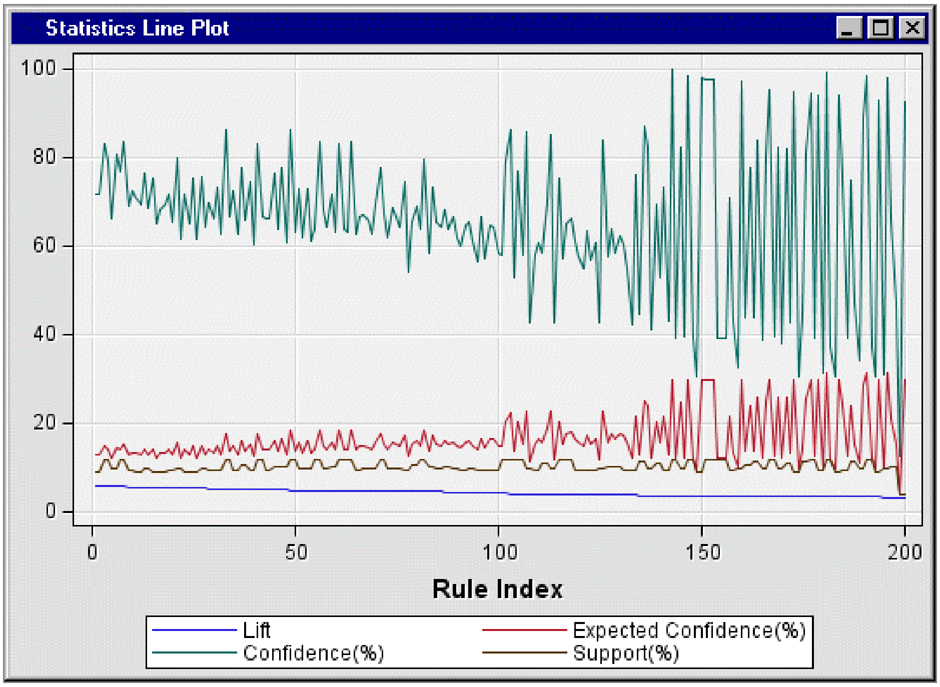
Right Hand of Rule = cracker
Left Hand of Rule = soda
Confidence(%) = 78.93

A data point in the plot corresponds to a transaction rule. The ToolTip displays the rule and the Confidence (%) value of the rule when you position the mouse pointer over a selected data point.

The data underlying the rule matrix plot can be viewed in the "Rules Table" on page 426 .

Statistics Line Plot

The statistics line plot displays a line plot of various values for each rule on the Y-axis. Here is an example of the statistics line plot.



The X-axis represents the values of Rule Index.

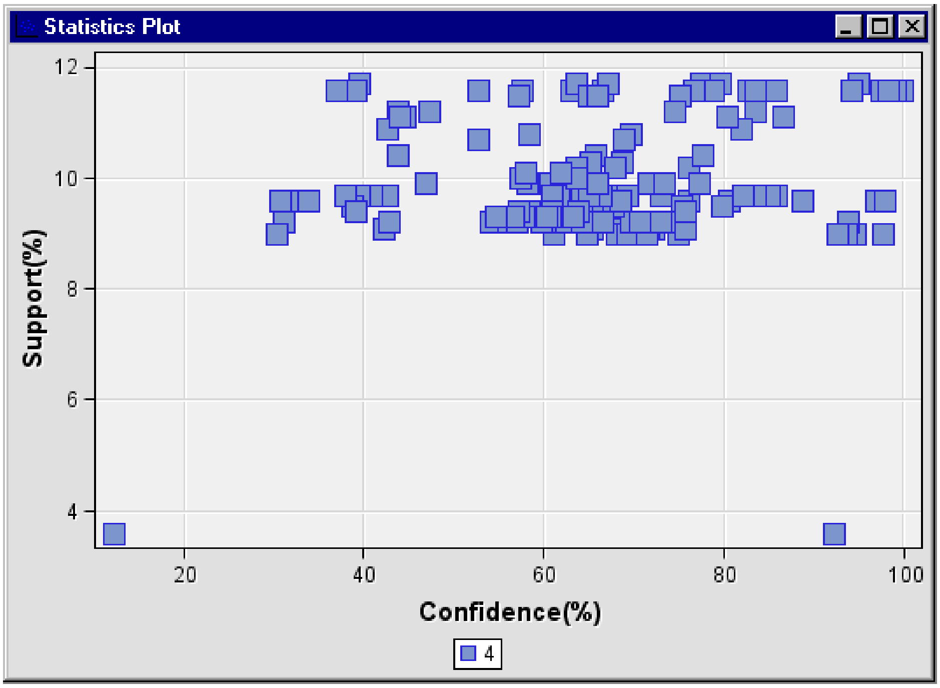
- **Lift** — is not available in a sequence analysis.
- **Expected Confidence (%)** — is not available in a sequence analysis.
- **Confidence (%)**
- **Support (%).**

The Statistics Line Plot and Rule Matrix windows are linked. When you select a data point in the rule matrix, the associated data points in the statistics line plot are highlighted.

The data underlying the statistics line plot can be viewed in the "Rules Table" on page 426 .

Statistics Plot

The statistics plot displays a scatter plot of Support (%) on the Y-axis versus Confidence (%) on the X-axis. The points are colored either by the number of items (association) or the chain size (sequence). Here is an example of the statistics plot.



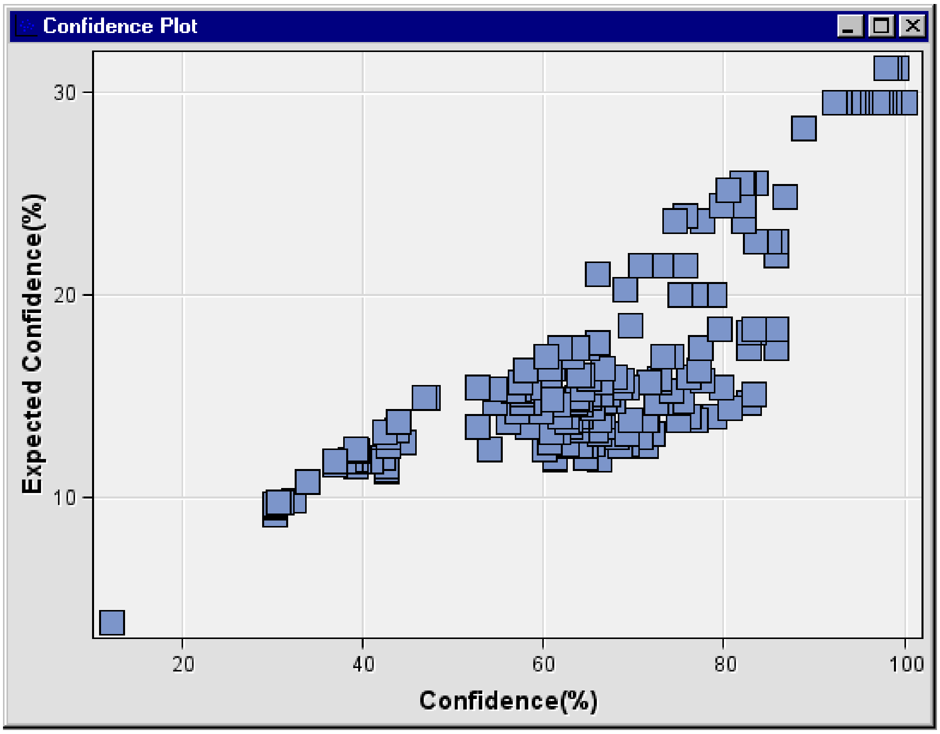
The X-axis and Y-axis represent the Confidence (%) and the Support (%), respectively.

The data underlying the statistics histogram can be viewed in the "Rules Table" on page 426 .

Confidence Plot

The confidence plot is available for an association analysis only.

The confidence plot displays a scatter plot of Confidence (%) on the X-axis versus Expected Confidence (%) on the Y-axis. If the confidence values are close to the expected confidence, the data points will be close to a 45-degree line that starts from the origin. Here is an example of the confidence plot.

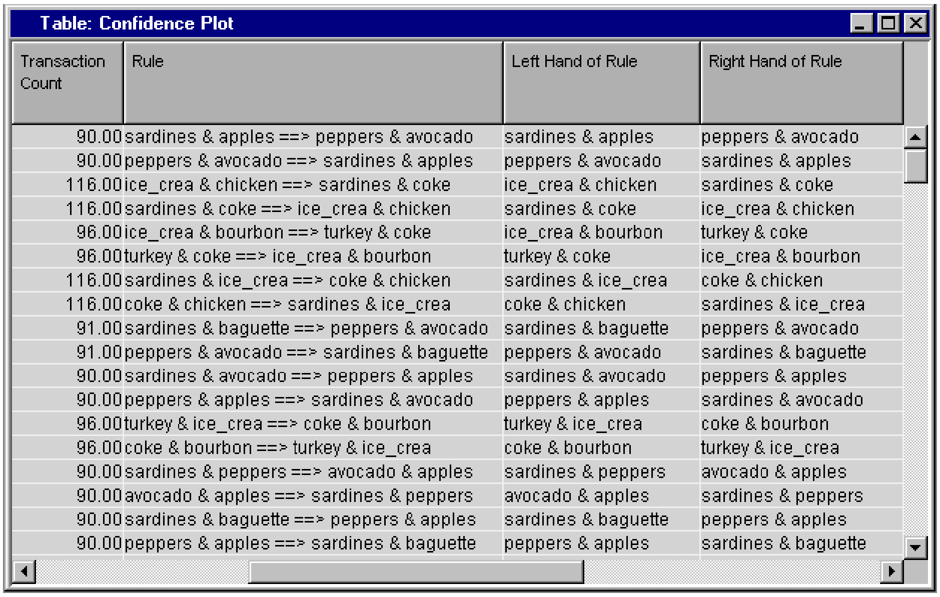


The data underlying the confidence plot can be viewed in the "Rules Table" on page 426 .

Rules Table

The rules table displays the information about the association rules that are generated in the node. The rules table contains the underlying data that is used to generate the rule matrix plot, statistics line plot, statistics plot, and the confidence plot.

To view the rules table while any of the above plots are open, click the plot to select it. Then, from the Results window main menu, select **View** ⇨ **Table**.



Transaction Count	Rule	Left Hand of Rule	Right Hand of Rule
90.00	sardines & apples ==> peppers & avocado	sardines & apples	peppers & avocado
90.00	peppers & avocado ==> sardines & apples	peppers & avocado	sardines & apples
116.00	ice_crea & chicken ==> sardines & coke	ice_crea & chicken	sardines & coke
116.00	sardines & coke ==> ice_crea & chicken	sardines & coke	ice_crea & chicken
96.00	ice_crea & bourbon ==> turkey & coke	ice_crea & bourbon	turkey & coke
96.00	turkey & coke ==> ice_crea & bourbon	turkey & coke	ice_crea & bourbon
116.00	sardines & ice_crea ==> coke & chicken	sardines & ice_crea	coke & chicken
116.00	coke & chicken ==> sardines & ice_crea	coke & chicken	sardines & ice_crea
91.00	sardines & baguette ==> peppers & avocado	sardines & baguette	peppers & avocado
91.00	peppers & avocado ==> sardines & baguette	peppers & avocado	sardines & baguette
90.00	sardines & avocado ==> peppers & apples	sardines & avocado	peppers & apples
90.00	peppers & apples ==> sardines & avocado	peppers & apples	sardines & avocado
96.00	turkey & ice_crea ==> coke & bourbon	turkey & ice_crea	coke & bourbon
96.00	coke & bourbon ==> turkey & ice_crea	coke & bourbon	turkey & ice_crea
90.00	sardines & peppers ==> avocado & apples	sardines & peppers	avocado & apples
90.00	avocado & apples ==> sardines & peppers	avocado & apples	sardines & peppers
90.00	sardines & baguette ==> peppers & apples	sardines & baguette	peppers & apples
90.00	peppers & apples ==> sardines & baguette	peppers & apples	sardines & baguette

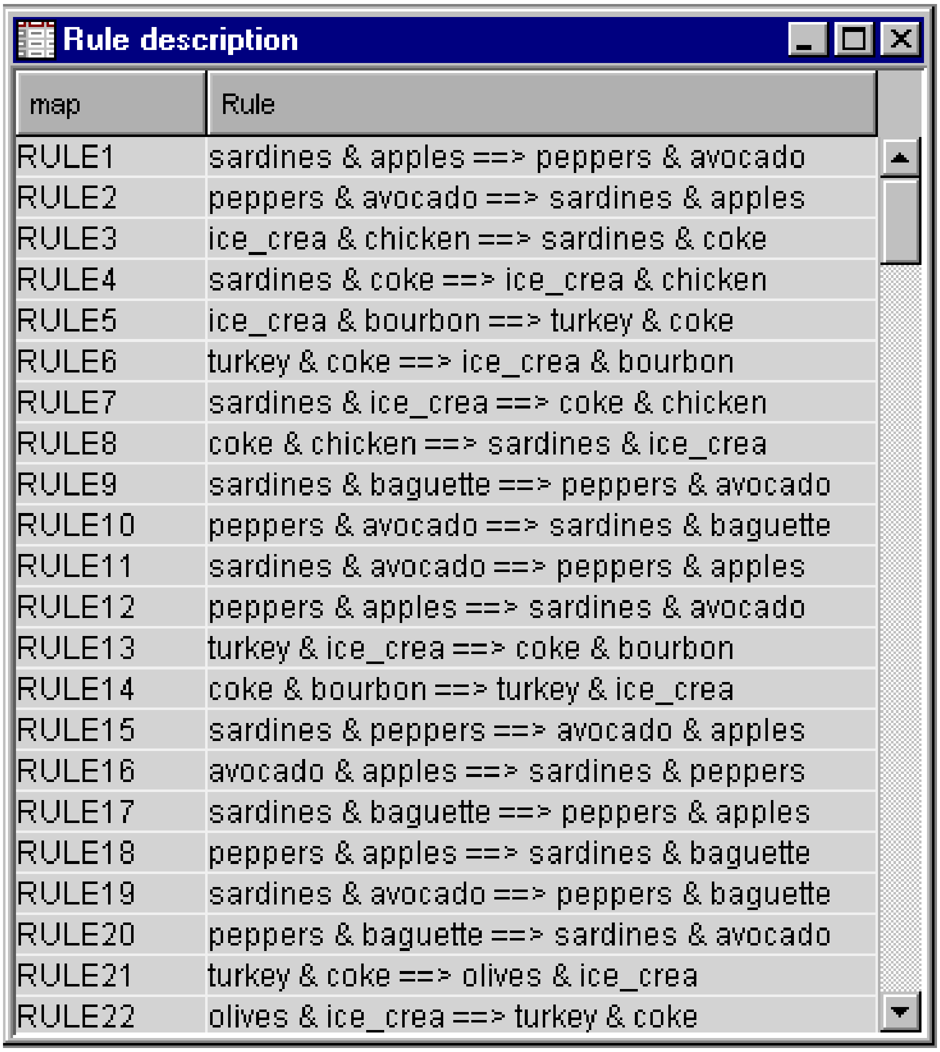
Alternately, you can open the rules table without using a plot by selecting **View ⇨ Rules ⇨ Rules Table** from the Results window main menu. You can right-click any column cell in the Rules table and select Show Variable Names to change the column headers from the default descriptive variable labels to the variable names that are used in SAS code. To restore variable labels from variable names, right-click any column cell and select Show Variable Labels.

The rules table contains the following columns:

- **Relations** — the number of items in the rule.
- **Expected Confidence (%)** — the percent of expected confidence, which is the number of transactions that satisfy the right-hand side of the rule divided by the total number of transactions.
- **Confidence (%)** — the percent confidence, which is the number of transactions that satisfy the rule divided by the number of transactions that satisfy the left-hand side of the rule.
- **Support (%)** — the percent support, which is the percentage of the total number of transactions that qualify for the rule.
- **Lift** — the lift ratio, which is the percent confidence divided by the percent expected confidence.
- **Transaction Count** — the number of transactions that meet the rule.
- **Rule** — the text of the rule, for example, A & B ==> C & D.
- **Left Hand of Rule** — identifies the left side of the rule, where the rule is expressed: _LHAND ==> _RHAND.
- **Right Hand of Rule** — identifies the right side of the rule, where the rule is expressed: _LHAND ==> _RHAND.
- **Rule Item 1** — individual items that make up the rule. Rule item 1 is the first item of the left-hand side.
- **Rule Item 2** — individual items that make up the rule. Rule item 2 is the second item of the left-hand side.
- **Rule Item 3** — individual items that make up the rule. Rule item 3 is the arrow ==>.
- **Rule Item 4** — individual items that make up the rule. Rule item 4 is the first item of the right-hand side.
- **Rule Item 5** — individual items that make up the rule. Rule item 5 is the second item of the right-hand side.
- **Rule Index** — an integer value that is used to uniquely identify each rule.
- **Transpose Rule** — a binary value that indicates whether the rule was transposed to create the Rule By ID data set that is to be used as training data.

Rule Description Table

The rule description table displays the rules and the map ID for each rule. The rule map ID corresponds to the rule index number that is displayed in the rules table.

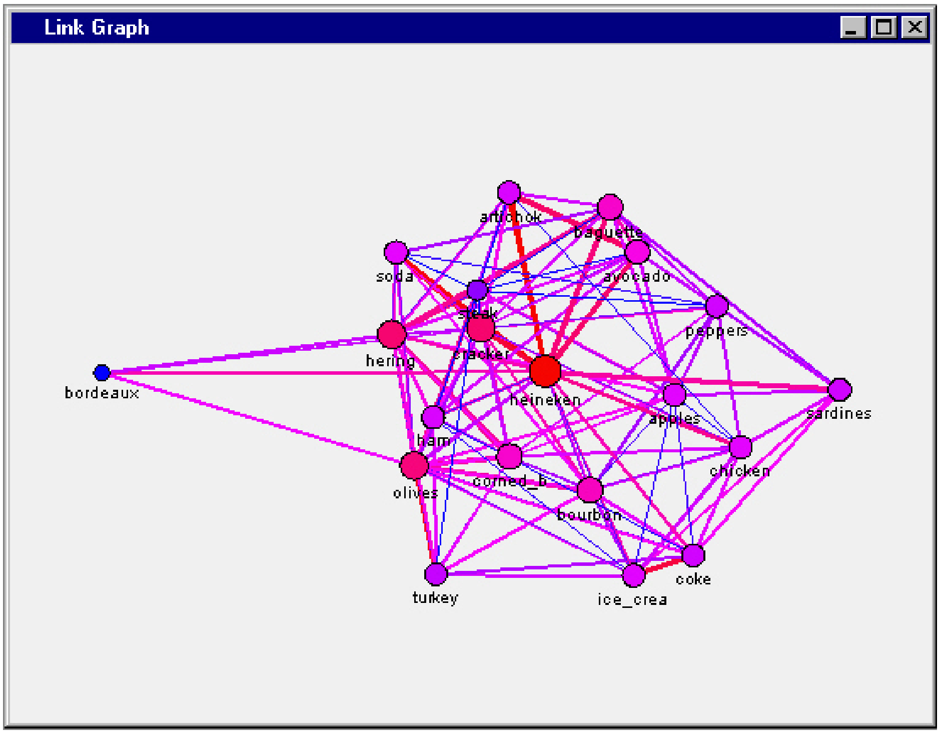


map	Rule
RULE1	sardines & apples ==> peppers & avocado
RULE2	peppers & avocado ==> sardines & apples
RULE3	ice_crea & chicken ==> sardines & coke
RULE4	sardines & coke ==> ice_crea & chicken
RULE5	ice_crea & bourbon ==> turkey & coke
RULE6	turkey & coke ==> ice_crea & bourbon
RULE7	sardines & ice_crea ==> coke & chicken
RULE8	coke & chicken ==> sardines & ice_crea
RULE9	sardines & baguette ==> peppers & avocado
RULE10	peppers & avocado ==> sardines & baguette
RULE11	sardines & avocado ==> peppers & apples
RULE12	peppers & apples ==> sardines & avocado
RULE13	turkey & ice_crea ==> coke & bourbon
RULE14	coke & bourbon ==> turkey & ice_crea
RULE15	sardines & peppers ==> avocado & apples
RULE16	avocado & apples ==> sardines & peppers
RULE17	sardines & baguette ==> peppers & apples
RULE18	peppers & apples ==> sardines & baguette
RULE19	sardines & avocado ==> peppers & baguette
RULE20	peppers & baguette ==> sardines & avocado
RULE21	turkey & coke ==> olives & ice_crea
RULE22	olives & ice_crea ==> turkey & coke

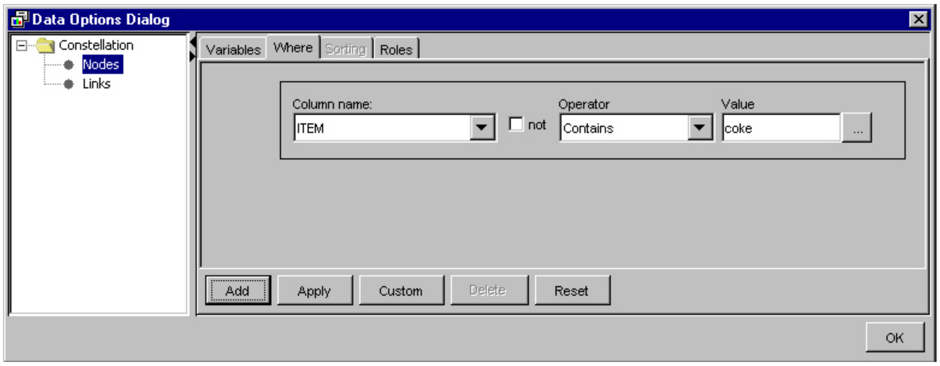
Link Graph

The link graph displays association results by using nodes and links. The default size and color of a node indicates the transaction counts in the Rules data set. Larger nodes have greater counts than smaller nodes. The color and thickness of links between nodes indicate the confidence level of a rule. The thicker the links are, the higher confidence the rules have.

The link graph sometimes shows too many nodes and links to display them clearly. Here is an example that was created using SAMPSIO.ASSOCS and with the Association node Chain Count property set to 6:

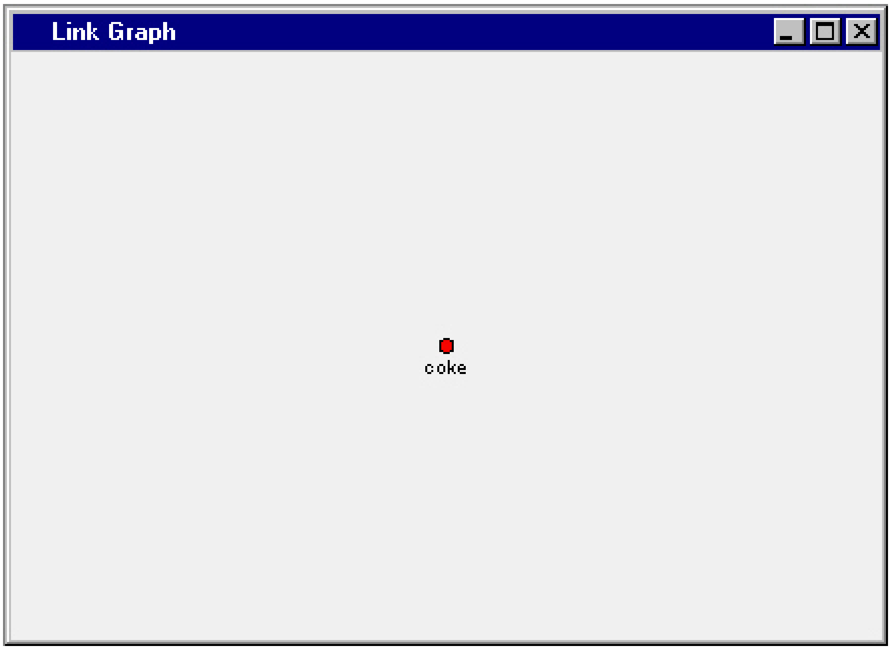


To make the link graph easier to read, you can create a WHERE clause to display a subset of the nodes and their links. Suppose that you want to select the nodes that contains the item coke only. You right-click in the link graph and select **Data Options**. In the Data Options window, select the Where tab. Use the Where tab to create a WHERE clause that selects nodes where **label** contains **coke**, then click the **Apply** and **OK** buttons.



Column name: ITEM
□ not
Operator: Contains
Value: coke

When you return to the link graph, it displays only nodes that contains the item **coke**.



If you want to manipulate more of the properties of the graphic elements in the link graph, right-click inside the plot and select Graph Properties again from the pop-up list.

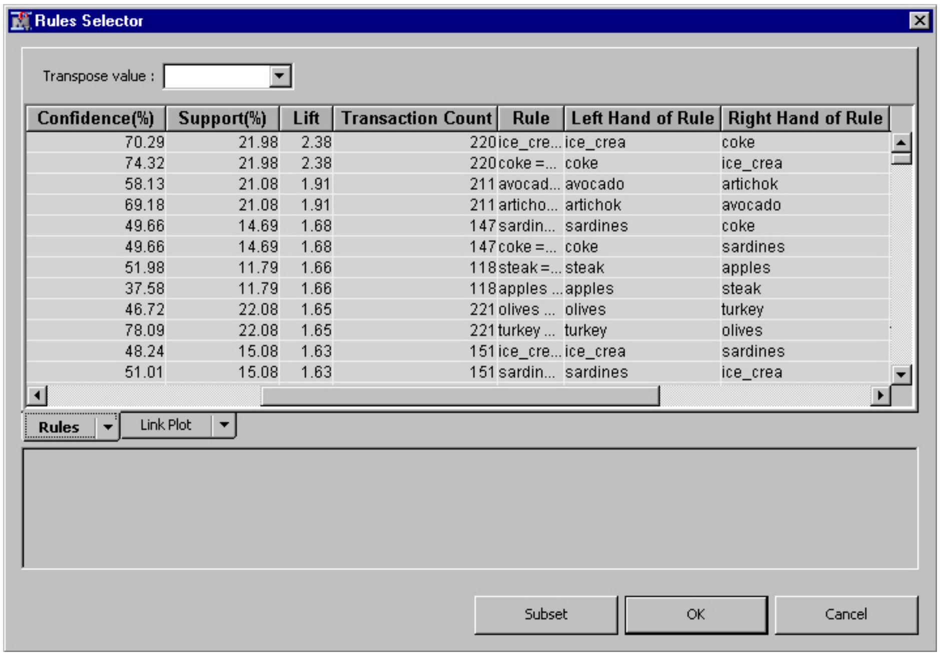
The Properties window contains three tabs that you can use to customize your link graph plots.

- The Graph tab of the Properties window has a Style section that has style options in a drop-down menu. The Graph tab also contains a check box that you can use to toggle the chart tips setting on and off.
- The Nodes tab of the Properties window has sections that you can use to customize the way that nodes are displayed in your link graph. You can modify the layout, size, and shape of your link graph. The layout section allows you to specify auto layout options, or a user layout. The Size section enables you to increase the default size of the nodes. The Shape section enables you to change the displayed node shape to squares, diamonds, circles, triangles, or only labels. The Nodes tab also contains check boxes that you can use to map color from data values, and toggle the display of node labels in the link graph on and off.
- The Links tab of the Properties window has three sections that you can use to customize the way node links are displayed in your link graph. You can specify a width that is fixed, or one that is mapped from data values. You can specify whether to show the threshold slider. When this option is selected you can use it with a scroll bar at the bottom of your link graph. It will show the links that satisfy the specified threshold that you specify. If the threshold slider is selected, you can choose to scale link values. The Link visibility section allows you to specify how the Link graph should display links. You can show all links, or show only those links that lead to a node, away from a node, or are among nodes. Check boxes also allow you to specify whether you want to map color from data values, be able to select links, or show links in a directed format. To see how these options work, try deselecting **Show all links**, and only select **Show links out of selection**. Select **Show Directed Links**. Then click **Apply**, and **OK**. Now click a node in your link graph.

Association Node Rules Selector

Enterprise Miner provides a tool called the Rules Selector. The Rules Selector is an interactive window for the Association node that allows users to decide which of the generated association rules should be transposed. The Rules Selector window is not available until after the Association node has successfully run.

To open the Rules Selector window after a node run, ensure that the Association node is selected in the process flow diagram, then locate the Rules property in the Association node's Properties Panel. Select the ... button in the right column of the Rules property to open the Rules Selector window.



Confidence(%)	Support(%)	Lift	Transaction Count	Rule	Left Hand of Rule	Right Hand of Rule
70.29	21.98	2.38	220	ice_cre...	ice_crea	coke
74.32	21.98	2.38	220	coke =...	coke	ice_crea
58.13	21.08	1.91	211	avocad...	avocado	artichok
69.18	21.08	1.91	211	articho...	artichok	avocado
49.66	14.69	1.68	147	sardin...	sardines	coke
49.66	14.69	1.68	147	coke =...	coke	sardines
51.98	11.79	1.66	118	steak =...	steak	apples
37.58	11.79	1.66	118	apples ...	apples	steak
46.72	22.08	1.65	221	olives ...	olives	turkey
78.09	22.08	1.65	221	turkey ...	turkey	olives
48.24	15.08	1.63	151	ice_cre...	ice_crea	sardines
51.01	15.08	1.63	151	sardin...	sardines	ice_crea

You can select individual or multiple association rules to transpose in the Rules tab of the Rules Selector window. Click individual rows to select them. You can also batch-select successive rows using <SHIFT>-click and non-successive rows using <CTRL>-click. Use the Transpose Value list box to choose Yes or No as your transpose setting. Click Subset to open the Rules Selector Where Builder window if you want to create a subset of your rules list. You can view rule changes graphically by examining the Link Plot tab. Select OK to accept or Cancel to negate association rules transpositions that were made in the Rules Selector window.

Association Node Output Data Sources

The Association node passes the following data source to the successor node:

- **Rules** (<libref>.ASSOC_Rules) — lists the first 200 rules that are found in a analysis.
- **Transaction** — the input transaction data for the Association node.

Association Node Examples

Example 1: Writing SAS Code to Create Transaction Data

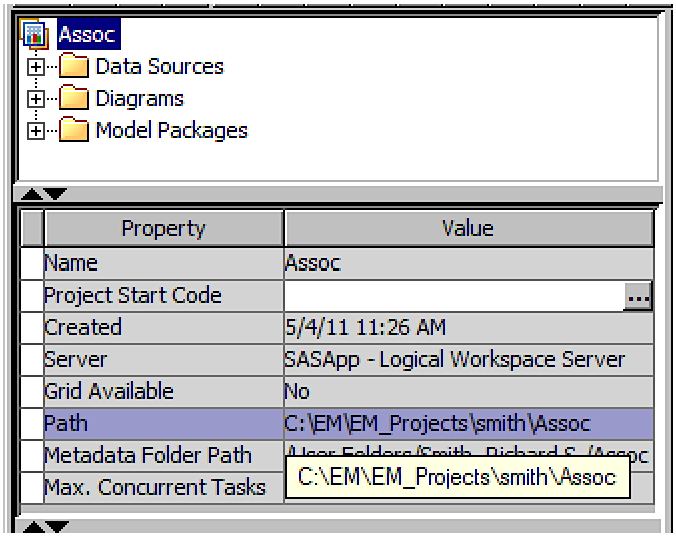
The Association node requires a transaction data set as the input. In a transaction data set, each observation represents a product that a customer purchased. However, it is very likely that your data is not in this format and not ready to be analyzed. This example shows you how to use SAS Code to transform your data set if the data is already in one of the following formats.

- Each observation corresponds to multiple items that a customer purchased over a period of time.
- Each observation corresponds to multiple items that a customer purchases at a visit.

The SAS code in the example creates a time-stamped variable. If you want to perform an association analysis that does not require a time-stamped variable, change the role of the time-stamped variable to **rejected**.

This Association node example assumes that you have already created and opened both a project and a diagram to contain this example. This example uses a Project named **Assoc** and a diagram named **Example 1**. For additional information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. You will store the association data that you generate for this example in the project directory that you created for this example. To find the path to your project directory, select your project in the Project Navigator, and write down the value for your **Path** property. This example uses the path `C:\EM\EM_Projects\smith\Assoc`. Your path will be different.



Property	Value
Name	Assoc
Project Start Code	
Created	5/4/11 11:26 AM
Server	SASApp - Logical Workspace Server
Grid Available	No
Path	C:\EM\EM_Projects\smith\Assoc
Metadata Folder Path	/User Folders/Smith, Richard S./Assoc
Max. Concurrent Tasks	

C:\EM\EM_Projects\smith\Assoc

Open the Program Editor window by selecting from the main menu **View** ⇨ **Program Editor**. Copy and paste the code below into your Program Editor window.

You will need to modify the code so the opening LIBNAME assignment references your project directory.

In the Program Editor window, locate the initial LIBNAME statement and replace the text, `c:\EM\EM_Projects\smith\Assoc` with the path to your project directory. Be sure to keep the single quote marks around your path specification.

```
/*** LIBNAME Assignment ***/
```

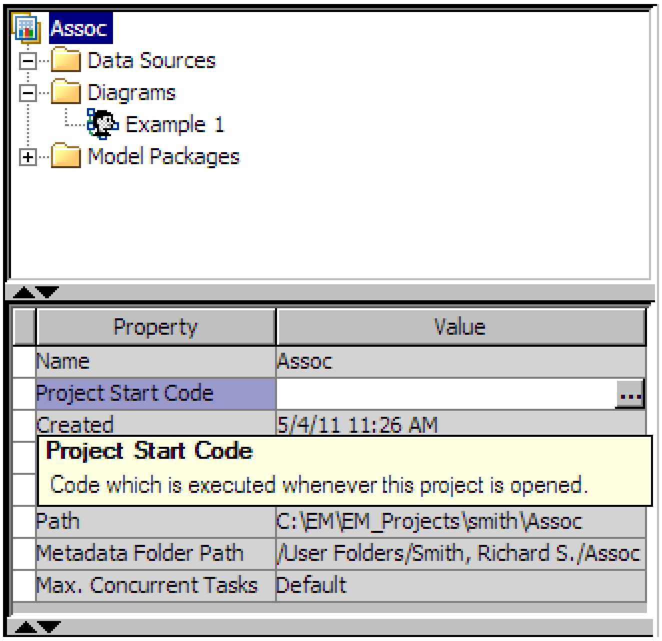
```
libname assocs 'c:\EM\EM_Projects\smith\Assoc';

/*** Part 1: Generating Data Set TEST1 ***/

DATA assocs.test1;
     input customer item1 $ item2 $ item3 $
        item4 $ item5 $ item6 $ item7 $;
     DATALINES;
     1 herring corned_b olives ham turkey
        bourbon ice_cream
     2 baquette soda herring cracker heineken
        olives corned_b
     3 avocado cracker artichok heineken ham
        turkey sardines
     ;
run;

/*** Part 2: Creating Data for Association Analysis ***/

PROC TRANSPOSE
  data=assocs.test1
  out=assocs.assocs1;
     var item1-item7;
  by customer;
run;

/*** Part 3: Creating Data for Sequence Analysis ***/

DATA assocs.seq1;
    set assocs.test1;
    array item(7) item1-item7;
  customer=0;
     do time=1 to 7;
        customer+1;
        product=item(time);
        output;
     end;
     drop item1-item7;
run;

/*** Part 4: Generating Data Set Test2 ***/

DATA assocs.test2;
     input customer time item1 $ item2 $ item3 $
        item4 $;
     datalines;
     1 1 herring corned_b olives .
     1 2 ham turkey bourbon ice_cream
     2 1 baquette soda . .
     2 2 herring cracker heineken olives
     2 3 corned_b . . .
     3 1 avocado cracker artichok heineken
     3 2 ham turkey sardines .
     ;
run;
```

```
/*** Part 5: Creating Data for Sequence Analysis ***/

PROC TRANSPOSE
   data=assocs.test2
   out=assocs.seq2(rename=(col1=product)
   drop=_name_
   where=(product is not missing));

   var item1-item4;
   by customer time;

run;
```

Use **Actions** ⇨ **Run** to run your code in the Program Editor.

If you correctly updated the LIBNAME statement and submitted the code, you generated the data sets required for this example.

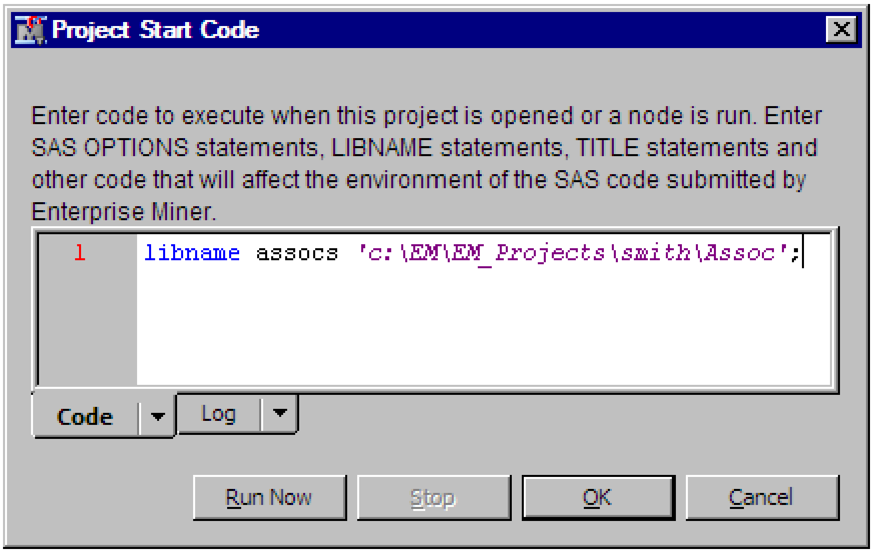
2. You need to put the LIBNAME statement you customized in Step 1 in your project start-up code to complete this example. Select the project name in your Project panel, then select the ellipsis button ... to the right of the **Project Start Code** property to open up the Project Start Code window.



Property	Value
Name	Assoc
Project Start Code	...
Created	5/4/11 11:26 AM
Path	C:\EM\EM_Projects\smith\Assoc
Metadata Folder Path	/User Folders/Smith, Richard S./Assoc
Max. Concurrent Tasks	Default

Project Start Code
Code which is executed whenever this project is opened.

Copy and paste your LIBNAME statement (with your unique project directory path) in the Project Start Code window. This example uses a LIBNAME statement to with the path **`C:\EM\EM_Projects\smith\Assoc`**. Your LIBNAME statement should be different, and should instead contain the path to your project directory.

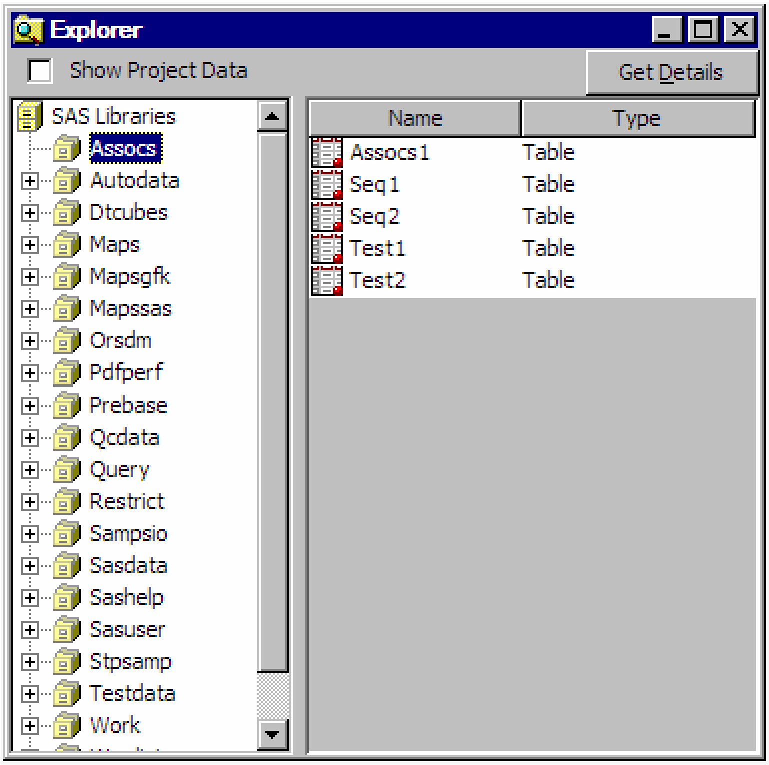


Enter code to execute when this project is opened or a node is run. Enter SAS OPTIONS statements, LIBNAME statements, TITLE statements and other code that will affect the environment of the SAS code submitted by Enterprise Miner.

```
libname assocs 'c:\EM\EM_Projects\smith\Assoc';
```

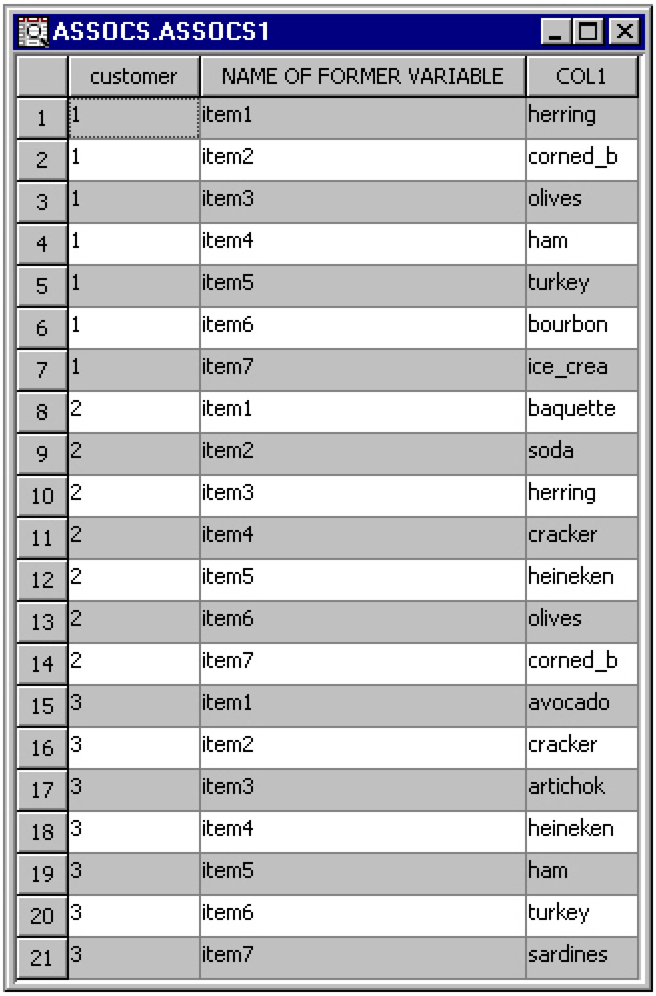
After entering your custom LIBNAME statement, click the **Run Now** button, then click the **OK** button. Your Enterprise Miner project now knows where files associated with the *assocs* library are located.

3. To view the data sets that you generated, use **View** ⇨ **Explorer** from the Enterprise Miner main menu to open the SAS Explorer. The SAS Explorer allows you to view the libraries and data sets that are associated with your project. Open the **Assocs** folder to verify that your data sets were created.



Name	Type
Assocs1	Table
Seq1	Table
Seq2	Table
Test1	Table
Test2	Table

In the Assocs library, there should be the five association data sets that you just created. You can double-click a data set in the right side of the Explorer window to open a view of the table:

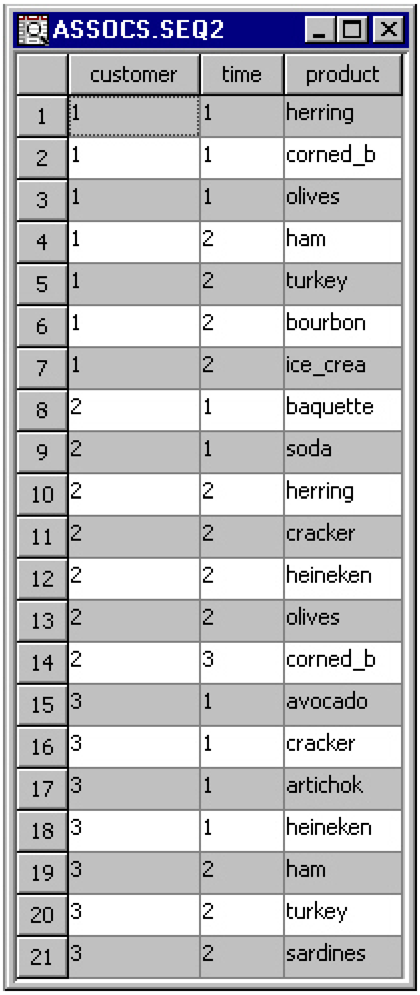


	customer	NAME OF FORMER VARIABLE	COL1
1	1	item1	herring
2	1	item2	corned_b
3	1	item3	olives
4	1	item4	ham
5	1	item5	turkey
6	1	item6	bourbon
7	1	item7	ice_crea
8	2	item1	baquette
9	2	item2	soda
10	2	item3	herring
11	2	item4	cracker
12	2	item5	heineken
13	2	item6	olives
14	2	item7	corned_b
15	3	item1	avocado
16	3	item2	cracker
17	3	item3	artichok
18	3	item4	heineken
19	3	item5	ham
20	3	item6	turkey
21	3	item7	sardines

The Assocs.Assocs1 data set is an example of an association analysis data set. In Part 2 of the example code, PROC TRANSPOSE is used to transform the data so that each observation in the resulting data set represents what a customer purchases during a visit. No time-stamped variable is created by the TRANSPOSE procedure. Therefore, the resulting data set Assocs.Assocs1 is only suitable for an association analysis.

The Assoc.Test1 and Assoc.Test2 are examples of time series and transaction data sets. Each observation in the Assocs.Test1 data set records the items that a customer purchased over a period of time. Each observation in the Assocs.Test2 data set records the items that a customer purchased during a visit.

4. Part 3 of the example code shows the use of SAS DATA step code to transform the data and to create a time-stamped variable. The following is a partial display of the resulting data.



ASSOCS.SEQ2

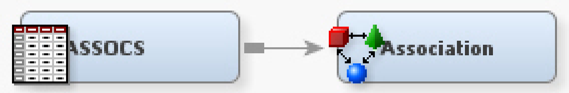
	customer	time	product
1	1	1	herring
2	1	1	corned_b
3	1	1	olives
4	1	2	ham
5	1	2	turkey
6	1	2	bourbon
7	1	2	ice_crea
8	2	1	baquette
9	2	1	soda
10	2	2	herring
11	2	2	cracker
12	2	2	heineken
13	2	2	olives
14	2	3	corned_b
15	3	1	avocado
16	3	1	cracker
17	3	1	artichok
18	3	1	heineken
19	3	2	ham
20	3	2	turkey
21	3	2	sardines

5. Now you have created the data sets that are in the correct format for the Association node. You can right-click the data sets in the SAS Explorer window and select **New Data Source** to open the Data Source Wizard that will convert the SAS data set into an Enterprise Miner Data Source. Then, you follow the discussions in "Creating a Data Source Using the Data Source Wizard" on page 293 to create data sources from the data sets you just created, for example, ASSOCS.SEQ2. In the Data Source Wizard — Column Metadata window, ensure the following variables are assigned the correct roles.

   - **CUSTOMER**: ID role
   - **PRODUCT**: target role
   - **TIME**: sequence role for a sequence analysis, or rejected role for an association analysis.

   In the Data Source Wizard — Data Source Attributes window, change the data source role from **Raw** to **Transaction**.

Example 2: Performing an Association Analysis



Suppose that you are a marketing analyst for a grocery chain. You want to identify items that are frequently purchased together. This information might help you make decisions such as when to distribute coupons, when to put a product on sale, or how to present items in store displays. To perform the association analysis, follow these steps.

To create a process flow diagram follow these steps:

1. Create a data source ASSOCS by using the SAS sample data set called SAMPSIO.ASSOCS. Using either the Basic or the Advanced Metadata Advisor.

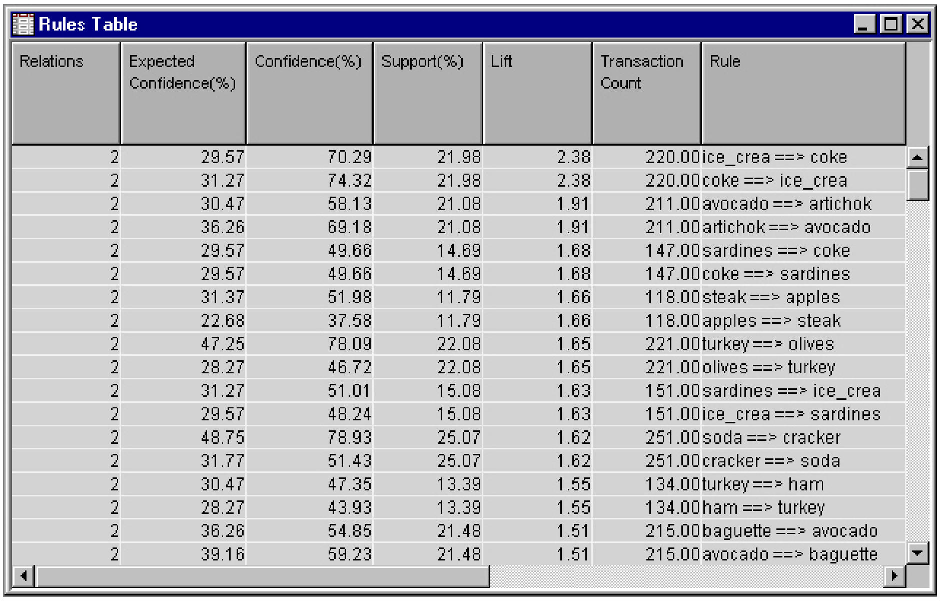
   Assign the following roles to these variables:

   - **CUSTOMER**: ID role
   - **PRODUCT**: Target role
   - **TIME**: Rejected role.

2. Assign the data source the role of **Transaction** in the Data Source Attributes window of the Data Source Wizard and save SAMPSIO.ASSOCS.
3. Add the data source SAMPSIO.ASSOCS to your diagram workspace.
4. Add an Association node to the diagram workspace and connect it to the data source ASSOCS.
5. Under the **Association** subgroup, change the **Maximum Items** property to 2. Run the Association node.
6. After the node runs successfully, open the Results window.

Next, you will want to view the results from the association analysis. The Results window displays several plots and a Rules Table for the association rules. You use the **Rule** item under the View main menu to select a plot. You open the Rules Table by selecting **View** ⇨ **Rules** ⇨ **Rules Table** from the menu.

1. Open the Rules Table. This table contains the relations, expected confidence, confidence, support, lift, transaction counts, the association rule, the left-hand of the rule, the right-hand of the rule, individual rule items, the rule index, and the transpose rule.

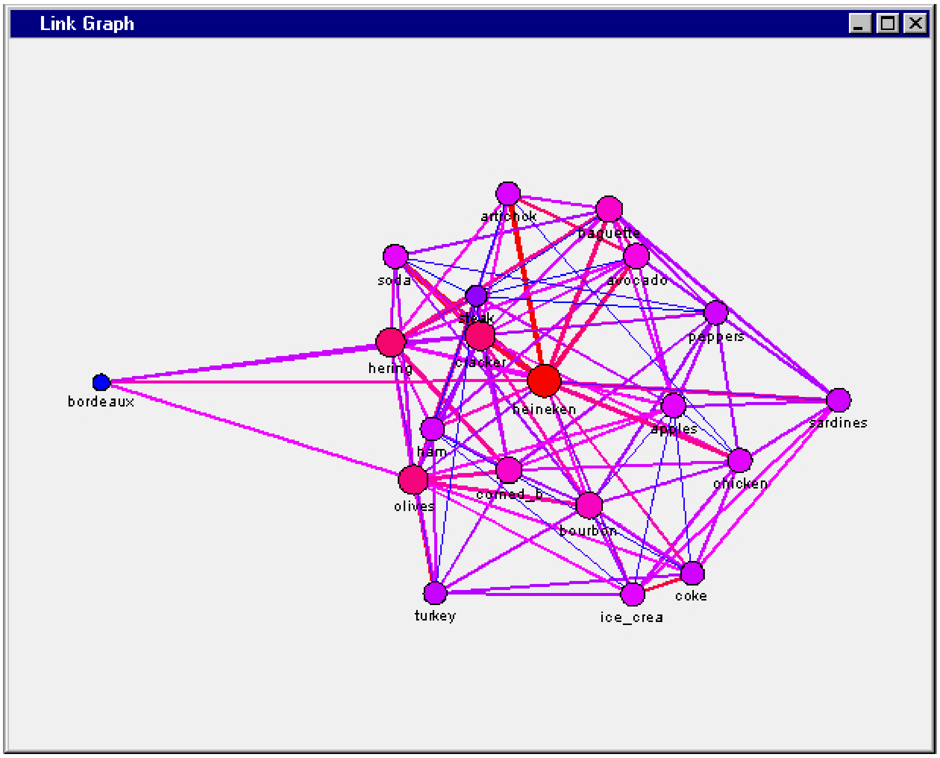


Rules Table

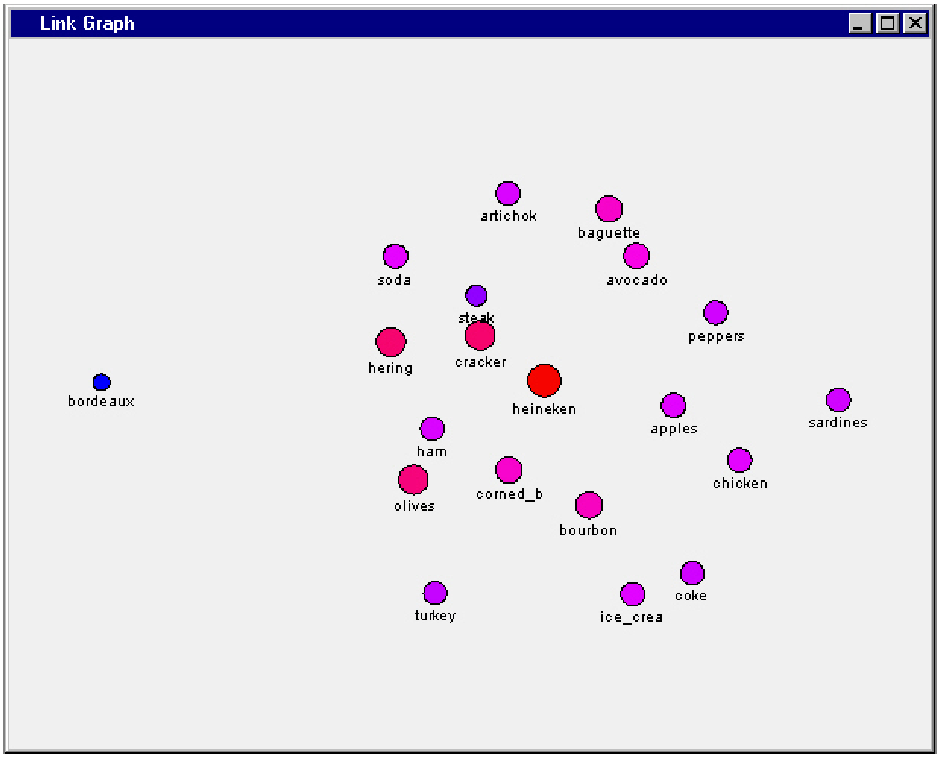
Relations	Expected Confidence(%)	Confidence(%)	Support(%)	Lift	Transaction Count	Rule
2	29.57	70.29	21.98	2.38	220.00	ice_crea ==> coke
2	31.27	74.32	21.98	2.38	220.00	coke ==> ice_crea
2	30.47	58.13	21.08	1.91	211.00	avocado ==> artichok
2	36.26	69.18	21.08	1.91	211.00	artichok ==> avocado
2	29.57	49.66	14.69	1.68	147.00	sardines ==> coke
2	29.57	49.66	14.69	1.68	147.00	coke ==> sardines
2	31.37	51.98	11.79	1.66	118.00	steak ==> apples
2	22.68	37.58	11.79	1.66	118.00	apples ==> steak
2	47.25	78.09	22.08	1.65	221.00	turkey ==> olives
2	28.27	46.72	22.08	1.65	221.00	olives ==> turkey
2	31.27	51.01	15.08	1.63	151.00	sardines ==> ice_crea
2	29.57	48.24	15.08	1.63	151.00	ice_crea ==> sardines
2	48.75	78.93	25.07	1.62	251.00	soda ==> cracker
2	31.77	51.43	25.07	1.62	251.00	cracker ==> soda
2	30.47	47.35	13.39	1.55	134.00	turkey ==> ham
2	28.27	43.93	13.39	1.55	134.00	ham ==> turkey
2	36.26	54.85	21.48	1.51	215.00	baguette ==> avocado
2	39.16	59.23	21.48	1.51	215.00	avocado ==> baguette

The lift, confidence, and level of support are three important evaluation criteria of association discovery. Lift is the ratio of the confidence factor to the expected confidence. Lift is a factor by which the likelihood of consequent increases given an antecedent. Values of lift greater than one are desirable. For the first rule in the example display, the two-item rule **ice_crea ==> coke** indicates that if a customer buys ice cream, 70.29% of the time he/she will buy coke. From the second rule in the display, if a customer buys coke, 74.32% of the time that he/she will buy ice cream. The level of support for these rules are the same, approximately 22%, because support is the proportion of the customers in the study who buy these two items. Note that the lift values for these rules are greater than 1.00.

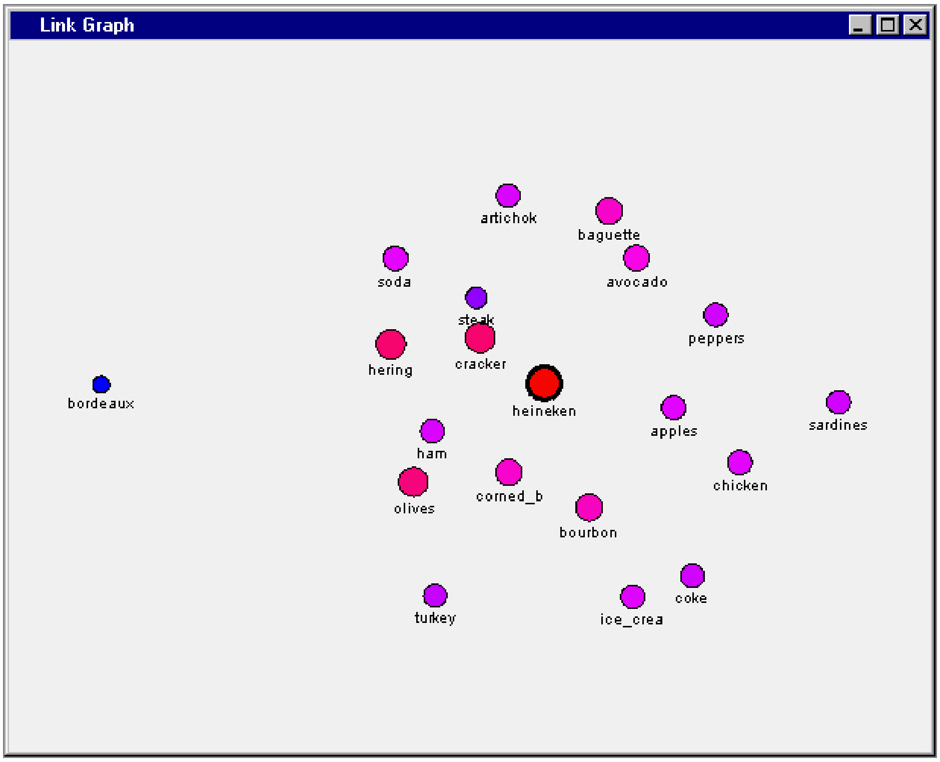
2. Open the link graph by selecting **View** ⇨ **Rules** ⇨ **Link Graph** from the Results main menu.



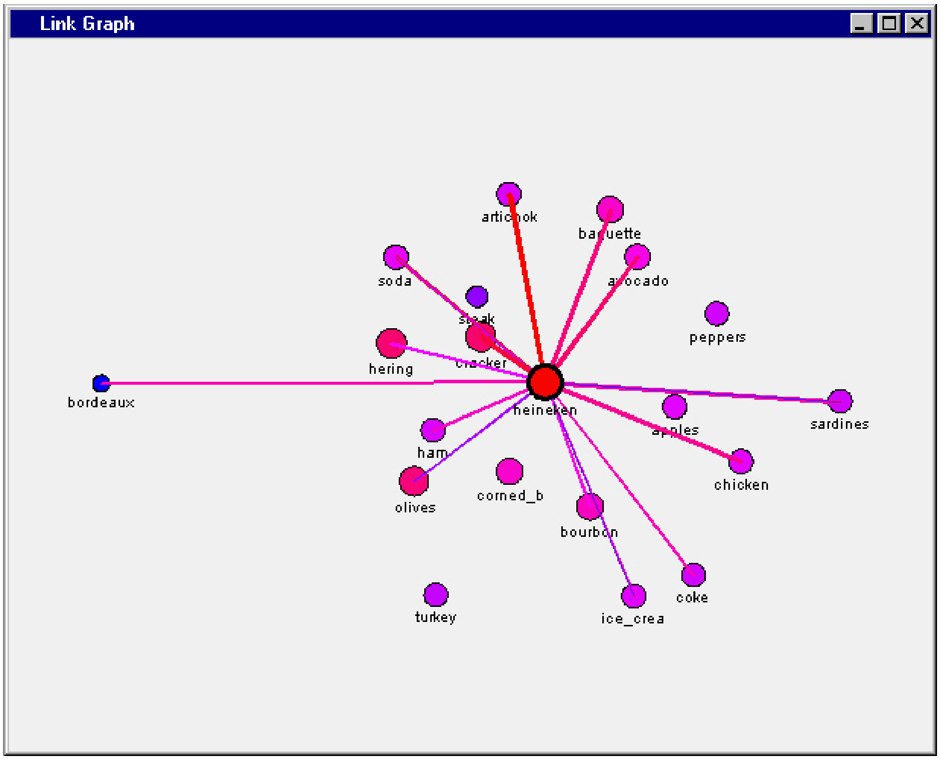
3. Right-click inside the link graph, and select Graph Properties from the pop-up menu. In the Properties window, select **Links** in the left panel to display the links properties. Deselect the **Show All Links** setting in the Link Visibility box. The three component link check boxes (Show links into selection, Show links out of selection, and Show links among selection) become available. Deselect all three of the component link check boxes, click **OK**, and re-examine the link graph. Observe that the largest node is for the item heineken because Heineken is the most popular product that customers purchased.



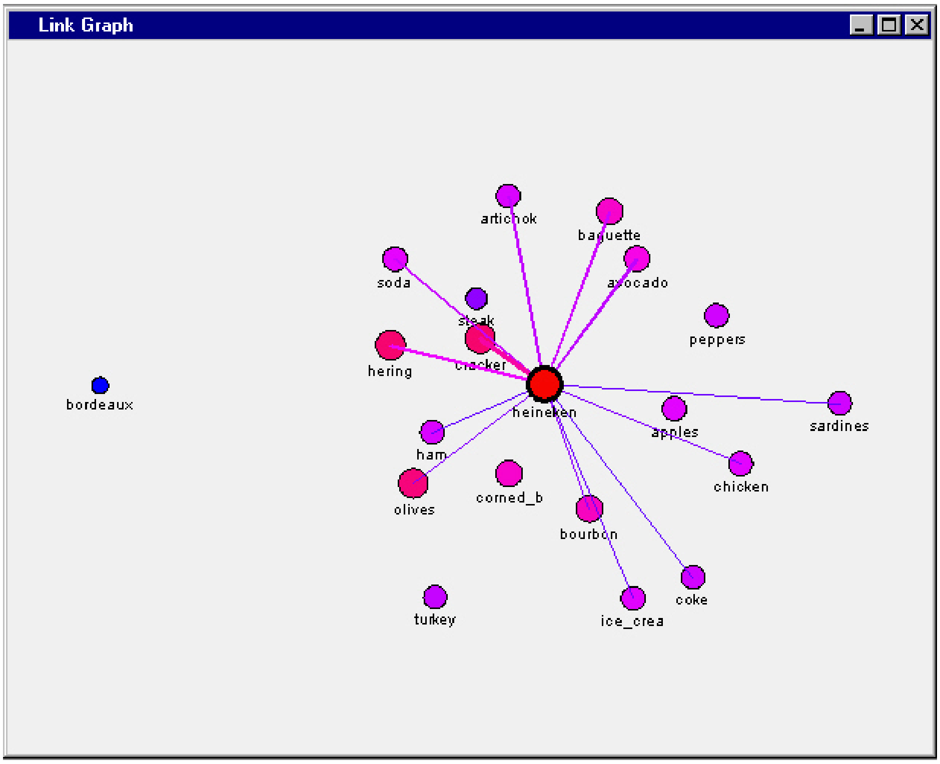
4. We want to examine the heineken node. Click the heineken node in the center of the plot to select it.



Right-click the selected heineken node, and then select Graph Properties from the menu. When the Properties — Links window opens, place it side-by-side with the Link Graph window, and then re-select the **Link Visibility** check boxes for **Show links into section** and **Show links out of selection**. Click the **Apply** button and watch the Link Graph window. The plot now displays links that go into and come from the selected heineken node.



5. Bordeaux has a link to heineken. Do people buy bordeaux before (as an antecedent) or after (as a consequent) they buy heineken? If you open the Graph Properties window once more, you can toggle the Link Visibility settings. The **Show links into selection** setting displays the links for the rules that have heineken as the antecedent. The **Show links out of selection** setting displays the links for the rules that have heineken as the consequent. The **Show links into selection** display below identifies the items that customer frequently purchase before they buy the product heineken. The plot below shows that the customers in the analyzed data bought bordeaux after purchasing heineken.


Link Graph
artichok
baguette
avocado
soda
steak
peppers
hering
cracker
bordeaux
heineken
sardines
apples
ham
chicken
olives
corned_b
bourbon
coke
turkey
ice_crea

Chapter 30
Cluster Node

Cluster Node

Overview of the Cluster Node

Use the **Cluster** node to perform observation clustering, which can be used to segment databases. Clustering places objects into groups or clusters suggested by the data. The objects in each cluster tend to be similar to each other in some sense, and objects in different clusters tend to be dissimilar. If obvious clusters or groupings could be developed prior to the analysis, then the clustering analysis could be performed by simply sorting the data.

The clustering methods in the **Cluster** node perform disjoint cluster analysis on the basis of Euclidean distances computed from one or more quantitative variables and seeds that are generated and updated by the algorithm. You can specify the clustering criterion that is used to measure the distance between data observations and seeds. The observations are divided into clusters so that every observation belongs to at most one cluster.

To perform clustering, you must have a raw data set or a training data set feeding into the **Cluster** node.

After clustering is performed, the characteristics of the clusters can be examined graphically using the Clustering Results Package. The consistency of clusters across variables is of particular interest. The multidimensional charts and plots enable you to graphically compare the clusters.

Note that the cluster identifier for each observation can be passed to other nodes for use as an input, ID, group, or target variable. The default is group variable.

The **Cluster** node also exports the cluster statistics and the cluster seed data sets to the successor nodes.

Preprocessing Data for the Cluster Node

The **Cluster** node can impute missing values of database observations. However, you might want to preprocess the data in other ways before using the **Cluster** node. For example, you might want to remove outliers, as they often appear as individual clusters, and they might distort other, more important clusters. For most applications, the variables should be transformed so that equal distances are of equal practical importance.

The **Cluster** node accepts binary, nominal, ordinal, and interval data. Binary, nominal, and ordinal variables are coded as numeric dummy variables for processing. See "Coding of the Class Variables in the Cluster Node" on page 459 for more information.

Variables with large variances tend to have more effect on the resulting clusters than variables with small variances. If all variables are measured in the same units (for example, dollars), then standardization might not be necessary. Otherwise, some form of standardization is recommended.

Nonlinear transformations of the variables can change the number of population clusters. Therefore, you should be careful when using nonlinear transformations.

The following nodes are particularly useful in preprocessing data for cluster analysis:

- Use the **Sampling** node to select a random sample of the data for initial analysis.
- Use the **Transform Variables** node to standardize the variables in some way.
- Use the **Data Partition** node to partition the data into training, validation, and test data sets. Partitioning has no effect on the cluster analysis itself. However the **Cluster** node can score these data sets.

Cluster Node Properties

Cluster Node General Properties

The following general properties are associated with the **Cluster** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first cluster node added to a diagram will have a Node ID of Clus. The second cluster node added to a diagram will have a Node ID of Clus2, and so on.
- **Imported Data** — The Imported Data property accesses the Imported Data — Cluster window. The Imported Data — Cluster window contains a list of the ports that provide data sources to the **Cluster** node. Select the ... button to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — accesses the Exported Data — Cluster window. The Exported Data — Cluster window contains a list of the output data ports that the **Cluster** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Cluster Node Train Properties

The following train properties are associated with the **Cluster** node:

- **Variables** — Select the ... button to open the Variables — Clus table, which enables you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use and Report variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before you click **Apply**.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Cluster Variable Role** — Use the Cluster Variable Role property to specify the model role that you want to be assigned to the cluster variable. By default, the cluster variable (segment identifier) is assigned a role of group. Model roles are Segment, ID, Input, and Target. The model role of Segment is useful for BY-group processing. Note that the segment identifier retains the selected variable role as it is passed to subsequent tools in the process flow diagram. The default setting for the Cluster Variable Role property is **Segment**.

- **Internal Standardization** — Use the Internal Standardization property to specify the internal standardization method that the **Cluster** node should use.

  - **None** — the variables are not standardized before clustering.

- **Standardization** — (default setting) the variable values are divided by the standard deviation. The mean is not subtracted.
- **Range** — the variable values are divided by the range. The mean is not subtracted.

Cluster Node Train Properties: Number of Clusters

- **Specification Method** — Use the Specification Method property to choose the method that SAS Enterprise Miner will use to determine the maximum number of clusters. Choose between Automatic and User-Specified methods.
  - The **Automatic** setting (default) configures SAS Enterprise Miner to automatically determine the optimum number of clusters to create.

    When the **Automatic** setting is selected, the value in the **Maximum Number of Clusters** property in the **Number of Clusters** section is not used to set the maximum number of clusters. Instead, SAS Enterprise Miner first makes a preliminary clustering pass, beginning with the number of clusters that is specified as the **Preliminary Maximum** value in the **Selection Criterion** properties.

    After the preliminary pass completes, the multivariate means of the clusters are used as inputs for a second pass that uses agglomerative, hierarchical algorithms to combine and reduce the number of clusters. Then, the smallest number of clusters that meets all four of the following criteria is selected.

    - The number of clusters must be greater than or equal to the number that is specified as the Minimum value in the Selection Criterion properties.
    - The number of clusters must have cubic clustering criterion statistic values that are greater than the CCC Cutoff that is specified in the **Selection Criterion** properties.
    - The number of clusters must be less than or equal to the **Final Maximum** value.
    - A peak in the number of clusters exists.

    Note: If the data to be clustered is such that the four criteria are not met, then the number of clusters is set to the first local peak. In this event, the following warning is displayed in the Cluster node log:

```
WARNING: The number of clusters selected based on the CCC values may not be valid. Pleas
```

    After the number of clusters is determined, a final pass runs to produce the final clustering for the **Automatic** setting.

  - The **User-Specified** setting enables you to use the Maximum value in the Selection Criterion properties to manually specify an integer value greater than 2 for the maximum number of clusters. Because the Maximum Number of Clusters property is a maximum, it is possible to produce a number of clusters that is smaller than the specified maximum.

    To force the number of clusters to some exact number n, specify n for both the Maximum and Minimum values in the Selection Criterion properties. It won't matter whether you choose Automatic or User Specify as your Specification Method. Either method will produce exactly n clusters.

- **Maximum Number of Clusters** — Use the Maximum Number of Clusters property to specify the maximum number of clusters that you want to use in your analysis. Permissible values are nonnegative integers. The minimum value for the Maximum Number of Clusters property is 2, and the default value is 10.

Cluster Node Train Properties: Selection Criterion

- **Clustering Method** — If you select Automatic as your Specification Method property, Clustering Method specifies how SAS Enterprise Miner calculates clustering distances.
  - **Average** — the distance between two clusters is the average distance between pairs of observations, one in each cluster.
  - **Centroid** — the distance between two clusters is defined as the (squared) Euclidean distance between their centroids or means.
  - **Ward** — (default) the distance between two clusters is the ANOVA sum of squares between the two clusters summed over all the variables. At each generation, the within-cluster sum of squares is minimized over all partitions obtainable by merging two clusters from a previous generation.
- **Preliminary Maximum** — Preliminary Maximum specifies the maximum number of clusters to create during the preliminary training pass. The default setting for the Maximum property is 50. Permissible values are integers greater than 2.
- **Minimum** — Minimum specifies the minimum number of clusters this is acceptable for a final solution. The default setting for the Minimum property is 2. Permissible values are integers greater than 2.
- **Final Maximum** — Final Maximum specifies the maximum number of clusters that are acceptable for a final solution. The default setting for the Maximum property is 20. Permissible values are integers greater than 2.
- **CCC Cutoff** — If you select Automatic as your Specification Method, the CCC Cutoff specifies the minimum cubic clustering cutoff criteria. The default setting for the CCC Cutoff property is 3. Permissible values for the CCC Cutoff property are integers greater than 0.

Cluster Node Train Properties: Encoding of Class Variables

The encoding method specifies how to compute one or more quantitative variables to be used for computing distances. These quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables computed for an encoding is called the dimension of the encoding.

- **Ordinal Encoding** — Use the Ordinal Encoding property to specify the encoding method that you want to use on the ordinal class variables in your data. Each encoding method derives one or more quantitative variables to be used for computing distances. The quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables computed for an encoding is called the *dimension*. The available ordinal encoding methods are **Rank**, **Index**, **Bathtub**, and **Thermometer**. The default ordinal encoding method is Rank.

  To illustrate the choices below, assume the following data set:

  ```
  DATA;
  INPUT x $ @@;
  CARDS;
  d c b a d c b d c d
  ```

  Each matrix below represents the ordinal class variable encoding method that is applied to the example data set.

  - **Rank Encoding** — (Default Setting) The dimension is 1.

    ```
    LEVEL=ORDINAL(RANK)
        X   T_X
        --------
    ```

```
    a   0.05
    b   0.20
    c   0.45
    d   0.80
```

  - **Index Encoding** — The dimension is 1.

    ```
    LEVEL=ORDINAL(INDEX)
        X       T_X
        ------------
        a         0
        b   0.333333
        c   0.666667
        d         1
    ```

  - **Thermometer Encoding** — The dimension is the number of categories minus 1.

    ```
    LEVEL=ORDINAL(THERMOMETER)
        X  Xa  Xb  Xc
        -------------
        a   0   0   0
        b   1   0   0
        c   1   1   0
        d   1   1   1
    ```

  - **Bathtub Encoding** — The dimension is the number of categories minus 1.

    ```
    LEVEL=ORDINAL(BATHTUB)
        X       Xa       Xb       Xc
        ----------------------------
        a -0.63246 -0.63246 -0.63246
        b  0.63246 -0.63246 -0.63246
        c  0.63246  0.63246 -0.63246
        d  0.63246  0.63246  0.63246
    ```

- **Nominal Encoding** — Use the Nominal Variable Encoding property to specify the encoding that you want to use on nominal variables. The encoding method specifies how to compute one or more quantitative variables to be used for computing distances. The quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables that are computed for an encoding is called the *dimension*. The available nominal encoding methods are GLM, Deviation, and Reference. GLM is the default nominal class variable encoding method.

  To illustrate the choices below, assume the following data set:

  ```
  DATA;
  INPUT x $ @@;
  CARDS;
  d c b a d c b d c d
  ```

  Each matrix below represents the nominal class variable encoding method that was applied to the example data set.

  - **GLM Encoding** — The dimension is the number of categories.

    ```
    LEVEL=NOMINAL(GLM)
        X Xa Xb Xc Xd
        -------------
        a    1  0  0  0
        b    0  1  0  0
        c    0  0  1  0
    ```

```
    d    0  0  0  1
```

- **Deviation Encoding** — The dimension is the number of categories minus 1.

```
LEVEL=NOMINAL(DEVIATION)
    X  Xa  Xb  Xc
    --------------
    a   1   0   0
    b   0   1   0
    c   0   0   1
    d  -1  -1  -1
```

- **Reference Encoding** — The dimension is the number of categories minus 1.

```
LEVEL=NOMINAL(REFERENCE)
    X Xa Xb Xc
    -----------
    a  1  0  0
    b  0  1  0
    c  1  1  1
    d  0  0  0
```

Cluster Node Train Properties: Initial Cluster Seeds

- **Seed Initialization Method** — Use the Seed Initialization Method property to specify the method that you want to use to compute the initial cluster seeds.
  - **Default** — If the Initial property is set to Default and the Drift During Training property is set to Yes, then MacQueen's k-means algorithm is used to compute initial cluster seeds.
  - **First** — Selects the first s complete cases as initial seeds.
  - **MacQueen** — Uses MacQueen's *k*-means algorithm to compute the initial seeds. When you select MacQueen's algorithm, the Drift During Training property is automatically changed to Yes. If you select MacQueen and later decide you want to choose another method, you must first set the Drift During Training property to No.
  - **Full Replacement** — (Outlier) Selects initial seeds that are very well separated, using a full replacement algorithm.
  - **Princomp** — The principal components setting initializes seeds on an evenly spaced grid in the plane of the first two principal components. If the number of rows is less than or equal to the number of columns, then the first principal component is placed to vary with the column number, and the second principal component is placed to vary with the row number. If the number of rows is greater than the number of columns, then the first principal component is placed to vary with the row number, and the second principal component is placed to vary with the column number.
  - **Partial Replacement** — (Separate) Selects initial seeds that are well separated, using a partial replacement algorithm.
- **Minimum Radius** — Use the Minimum radius property to specify the minimum or the maximum distance that you want to allow between cluster seeds. The Minimum Radius value must be a real number greater than zero. The default setting for the Minimum Radius value is 0.0.
- **Drift During Training** — When the Drift During Training property is set to **Yes**, the **Cluster** node uses incremental training for one pass. This permits seeds to drift as the algorithm selects initial seeds. Temporary clusters are formed by assigning

each observation to the cluster with the nearest seed. Each time an observation is assigned, the cluster seed is updated as the current mean of the cluster. The default setting for the Drift During Training property is **No**.

Cluster Node Train Properties: Training Options

- **Use Defaults** — Use the Use Defaults property to specify whether you want to use default node settings for training. Set the Use Defaults property to No if you want to customize the training settings. The default setting for the Training Defaults property is Yes.
- **Settings** — Use the Settings property to open a window that you can use to specify training settings for the **Cluster** node. When the Use Defaults property is set to No, you can select the ... button to open a Cluster Training Options window, where you can configure the following properties:
  - **Learning Rate** — Use the Learning Rate property to specify the learning rate for training. The value specified for Learning Rate also becomes the default value for the Initial Rate and Final Rate properties. Permissible values for the Learning Rate property are real numbers between 0 and 1.
  - **Initial Rate** — Use the Initial Rate property to specify the initial learning rate for training. If a value is specified for the Learning Rate property, that value also becomes the value for Initial Rate. Acceptable values are real numbers between 0 and 1. The default setting for the Initial Rate property is 0.5.
  - **Final Rate** — Use the Final Rate property to specify the final learning rate for training. If a value is specified for the Learning Rate property, that value also becomes the value for Final Rate. Acceptable values are real numbers between 0 and 1. The default setting for the Initial Rate property is 0.02.
  - **Step Number to Reach Final Rate** — Use the Step Number to Reach Final Rate property to specify the step number at which the learning rate reaches the final learning rate. The acceptable values are integers greater than or equal to 1. The default setting for the Step Number to Reach Final Rate property is 1000.
  - **Maximum Number of Iterations** — Use the Maximum Number of Iterations property to specify the maximum number of clustering iterations that you want to be performed. Permissible values are integers greater than or equal to 1. Default values for the Maximum Number of Iterations property vary according to the clustering method that is used.
  - **Maximum Number of Steps** — Use the Maximum Number of Steps property to specify the maximum number of steps to perform during training. Permissible values are integers greater than or equal to 1. The default setting in 1200.
  - **Convergence Criterion Value** — Use the Convergence Criterion Value property to specify the value of the convergence criterion that you want to use in the computation of cluster seeds. The default convergence value is 0.0001.

Cluster Node Train Properties: Missing Values

In scored data sets, missing values in the original variables are not replaced. When you specify a missing values property for a given variable, a new variable is created with a name formed by combining the prefix IM_ with the name of the original variable. The new variable contains all of the nonmissing values of the original variable, as well as the replaced values for any observations where the original variable is missing.

- **Interval Variables** — Use the Interval Variables property from the **Cluster** node to specify how to handle observations that contain missing interval variable values during clustering and scoring. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.

  - **Default** — If the Scoring Imputation Method property is set to None, and the Interval Variables property setting is Default, then the Interval Variables property setting is not used. If the Scoring Imputation Method property is set to Seed of Nearest Cluster, and the Interval Variables property setting is Default, then missing interval variable values are ignored during cluster training, and missing values are imputed using nearest cluster seeds during cluster scoring.
  - **Ignore** — Missing interval variable values are ignored during clustering and scoring. For observations that have ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the nonmissing variables divided by the total variance of all non-rejected variables. The variances are obtained from the DMDB catalog and include the effect of variable standardizations.
  - **Mean** — Missing interval variable values are replaced by the variable mean during clustering and scoring.
  - **Midrange** — Missing interval variable values are replaced by the variable midrange during clustering and scoring.
  - **Omit** — Any observation with any missing values for any of the listed interval variables is omitted from the cluster analysis. Statistics from the DMDB catalog are not adjusted for omitted observations. In scoring, observations that have missing values are assigned a missing value for the distance and cluster number, and no missing values are replaced.

- **Nominal Variables** — Use the Nominal Variables property to specify how to handle observations that contain missing nominal variable values during clustering and scoring. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.
  - **Default** — If the Scoring Imputation Method property is set to None, and the Nominal Variables property setting is Default, then the Nominal Variables property setting is not used. If the Scoring Imputation Method property is set to Seed of Nearest Cluster, and the Nominal Variables setting is Default, then missing nominal variables are ignored during cluster training, and missing variables are imputed using the nearest cluster seed during cluster scoring.
  - **Ignore** — Missing nominal variable values are ignored by the **Cluster** node during clustering and scoring. For observations that have ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the nonmissing variables divided by the total variance of all non-rejected variables. The variances are obtained from the DMDB catalog and include the effects of variable standardizations.
  - **Mean** — Missing nominal variable values are replaced by the variable mean during clustering and scoring.
  - **Mode** — Missing nominal variable values are replaced by the variable mode during clustering and scoring. If there is no unique mode, the mean of all modes is used. The Mode option is available only for DMDB class variables.
  - **Omit** — Any observation that has missing values for any of the listed nominal variables is omitted from the cluster analysis. Statistics from the DMDB catalog are not adjusted for omitted observations. In scoring, observations that have missing values are assigned a missing value for the distance and cluster number, and no missing values are replaced.

- **Ordinal Variables** — Use the Ordinal Variables property to specify how to handle observations that contain missing ordinal variable values during clustering and scoring. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.
  - **Default** — If the Scoring Imputation Method property is set to None, and the Ordinal Variables property is set to Default, the Ordinal Variables property setting is not used. If the Scoring Imputation Method property is set to Seed of Nearest Cluster, and the Ordinal Variables property is set to Default, then missing ordinal variables are ignored during cluster initialization, and missing variables are replaced using the nearest cluster seed during cluster scoring.
  - **Ignore** — Missing ordinal variable values are ignored during clustering and scoring. For observations that have ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the nonmissing variables divided by the total variance of all non-rejected variables. The variances are obtained from the DMDB catalog and include the effects of variable standardizations.
  - **Median** — Missing ordinal variable values are replaced by the variable median during clustering and scoring. The Median option is available only for DMDB class variables.
  - **Mode** — Missing ordinal variable values are replaced by the variable mode during cluster initialization. If there is no unique mode, the mean of all modes is used. The Mode option is available only for DMDB class variables.
  - **Omit** — Any observation with any missing values for any of the listed ordinal variables is omitted from the cluster analysis. Statistics from the DMDB catalog are not adjusted for omitted observations. In scoring, observations that have missing values are assigned a missing value for the distance and cluster number, and no missing values are replaced.
- **Scoring Imputation Method** — Use the Scoring Imputation Method to specify the method for imputing missing values during scoring. You can choose between imputation methods of **None** or **Seed of Nearest Cluster**. The default setting is **None**.

Cluster Node Score Properties

The following score properties are associated with the **Cluster** node:

- **Cluster Variable Role** — Use the Cluster Variable Role property to specify the model role that you want to be assigned to the cluster variable. By default, the cluster variable (segment identifier) is assigned a role of group. Model roles are **Segment**, **ID**, **Input**, and **Target**. The model role of **Segment** is useful for BY-group processing. Note that the segment identifier retains the selected variable role as it is passed to subsequent tools in the process flow diagram. The default setting for the Cluster Variable Role property is Segment.
- **Hide Original Variables** — Use the Hide Original Variables property to specify whether you want to display the original transformed variables in the node output. The default setting for the Boolean Hide Original Variables property is Yes.
- **Cluster Label Editor** — Select the ... button to open an editor that you use to specify and edit descriptions to the various cluster segments. The node will use these strings to create a new variable called _SEGMENT_LABEL_ that describes the associated value of the _SEGMENT_ variable.

Cluster Node Report Properties

The following report properties are associated with the **Cluster** node:

- **Cluster Graphs** — When set to No, the Cluster Graphs property suppresses cluster graphical output showing distribution of variables. The default setting for the Cluster Graphs property is Yes.
- **Tree Profile** — When set to No, the Tree Profile property suppresses the tree profile from the output. The default setting for the Tree Profile property is Yes.
- **Distance Plot and Table** — Use the Distance Plot and Table report to specify whether to generate reports based on the distance between clusters. The default setting for the Distance Plot and Table property is Yes.

Cluster Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Cluster Node Results Window

You can open the Results window of the **Cluster** node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window that includes a read-only table of the **Cluster** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **Cluster** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — the Variables setting in the menu opens a window that contains a table of the variables submitted to the clustering node. The variables table displays the use, report, role, and level for each variable.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the cluster run.

  - **Output** — the SAS output of the cluster run. For the **Cluster** node, the SAS output includes a variable summary, Ward's Minimum Variance Cluster Analysis, Eigenvalues of the Covariance Matrix, RMS Total Sample Standard Deviation, RMS distance between observations, a Cluster history, and a Variable Importance table.
  - **Flow Code** — the SAS code used to produce the output that the **Cluster** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The **PMML Code** menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Summary Statistics** — Three summaries are available: a Mean Statistics table, a Cluster Statistics table, and a Segment Size pie chart.
  - **Mean Statistics** — The Mean Statistics table provides a row of cluster information for every cluster segment that was created during your Cluster node run.

    The Mean Statistics table contains the following columns for each segment row:

    - **Clustering Criterion** — the clustering criterion value. The numerical value is determined by the Clustering Method specified in the Cluster node Properties panel.
    - **Maximum Relative Change in Cluster Seeds** — the maximum relative change in the cluster seed values over the clusters from the previous iteration. The relative change in a cluster seed is the distance between the old seed and the new seed divided by the scaling factor.
    - **Improvement in Clustering Criterion** — the algorithm underlying the **Cluster** node attempts to minimize cubic clustering criterion scores. The improvement in clustering criterion refers to a relative change, or decrease, in the cubic clustering criterion values from one iteration to the next.
    - **Segment ID** — a numerical ID for a given cluster.
    - **Frequency of Cluster** — the sum of frequencies (observations) for nonmissing values in a cluster.
    - **Root-Mean-Square Standard Deviation** — the root mean square across variables of the cluster standard deviations, which is equal to the root mean square distance between observations in the cluster.
    - **Maximum Distance from Cluster Seed** — the maximum distance from the cluster seed to any observation in the cluster.
    - **Nearest Cluster** — the number of the cluster that has the mean closest to the mean of the current cluster.
    - **Distance to Nearest Cluster** — the distance between the centroids (means) of the current cluster and the nearest other cluster.
    - **Input Variable Columns** — The Mean Statistics table contains a column for each input variable that is used in the clustering calculations. Each variable's column contains the cluster means for each input variable, for each segment number.

- **Segment Size** — The Segment Size pie chart displays clusters by segment number. The width of each pie slice reflects each segment's percentage of all the clustered data. Each segment ID appears as you move your mouse pointer across pie slices of interest. To see the data table that SAS Enterprise Miner uses to create the Segment Size pie chart, click the pie chart to select it, then from the Results window main menu, select **View** ⇨ **Table**.
- **Cluster Statistics** — The Cluster Statistics table contains summary columns for each input variable as well as columns for segment values and cumulative statistics summed across all input variables. The Cluster Statistics table contains the following rows for each column:
  - **DMDB_FREQ** — the sum of frequencies (observations) for nonmissing values of each variable.
  - **DMDB_WEIGHT** — the sum of weights for nonmissing values of each variable.
  - **DMDB_MEAN** — the mean value statistic that is computed from all nonmissing values of each variable.
  - **DMDB_STD** — the standard deviation statistic computed from all nonmissing values of each variable. The Cluster Statistics table also provides a DMDB standard deviation statistic that is summarized over all input variables.
  - **LOCATION** — the STDIZE= location measure that incorporates replaced missing values.
  - **SCALE** — the STDIZE= scale measure that incorporates replaced missing values.
  - **DMDB_MIN** — the minimum of all nonmissing values for each variable.
  - **DMDB_MAX** — the maximum of all nonmissing values for each variable.
  - **CRITERION** — the clustering criterion value that is used for all variables, as calculated by the Clustering Method property specified in the Cluster node Properties Panel.
  - **PSEUDO_F** — the Pseudo-F statistic, summarized across all input variables [([(R2)/(c - 1)])/([(1 - R2)/(n - c)])] where R2 is the observed overall R2, c is the number of clusters, and n is the number of observations.
  - **ERSQ** — the approximate expected value of the squared multiple correlation R, which is the proportion of variance accounted for by the clusters under a uniform null hypothesis. If the number of clusters is greater than one-fifth of the number of observations, ERSQ and CCC are given missing values.
  - **CCC** — the cubic clustering criterion value, which is useful in determining the number of clusters in the data. CCC values that are greater than 2 or 3 indicate good clusters, values between 0 and 2 indicate potential clusters (but they should be considered with caution), and large negative values can indicate outliers. If the number of clusters is greater than one-fifth the number of observations, CCC and ERSQ are given missing values.
  - **TOTAL_STD** — the total standard deviation of each variable. The Cluster Statistics table provides total standard deviation statistics that are summed for all clusters and for individual input variables.
  - **WITHIN_STD** — the pooled within-cluster standard deviation of each variable. The Cluster Statistics table provides within-cluster standard

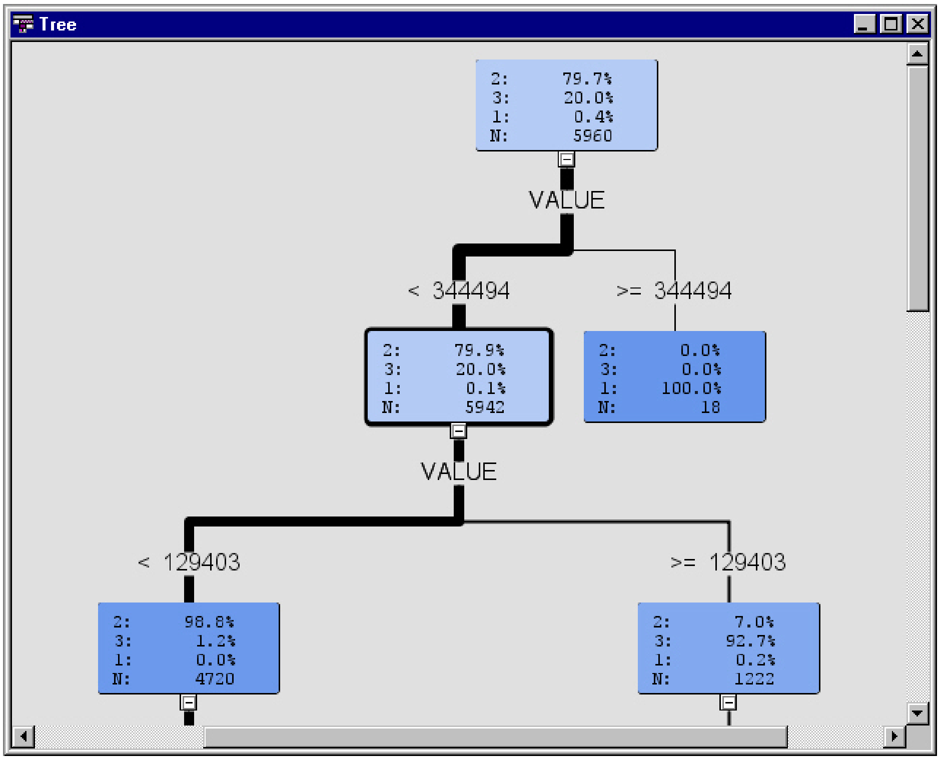
deviation statistics that are pooled for all clusters and for individual input variables.

  - **RSQ** — the squared multiple correlation R, which is the proportion of variance accounted for by the clusters. The Cluster Statistics table provides the RSQ statistic pooled over all input variables and for each input variable.
  - **RSQ_RATIO** — the ratio of between-cluster variance to within-cluster variance (R2/(1 - R2)). The Cluster Statistics table provides the RSQ-RATIO statistic that is summarized over all input variables and for each input variable.
  - **SEED** — seeds for each cluster and variable.
  - **CLUS_MEAN** — the mean of each variable within each cluster.
  - **CLUS_STD** — the standard deviations of each variable within each cluster.
  - **CLUS_MIN** — the minimum of each variable within each cluster.
  - **CLUS_MAX** — the maximum of each variable within each cluster.
  - **CLUS_FREQ** — the sum of frequencies for nonmissing values of each variable within each cluster.

- **CCC Plot** — displays a cubic clustering criterion scatter plot that charts number of clusters against cubic clustering criterion scores.
- **Input Means Plot** — displays a scatter plot that charts the normalized mean values for each input variable. The normalized mean values for each variable are tabulated for each individual cluster as well as across all clusters.

- **Cluster Profile** — view one of the following cluster profiles:
  - **Tree** — shows the decision tree that was used to form the individual clusters. The decision tree is based on your clustered data. The cluster variable (Segment ID) is used as the decision tree target variable. You can use the Cluster Profile Tree to identify influential input variables, such as which variables are most effective for grouping observations into clusters.

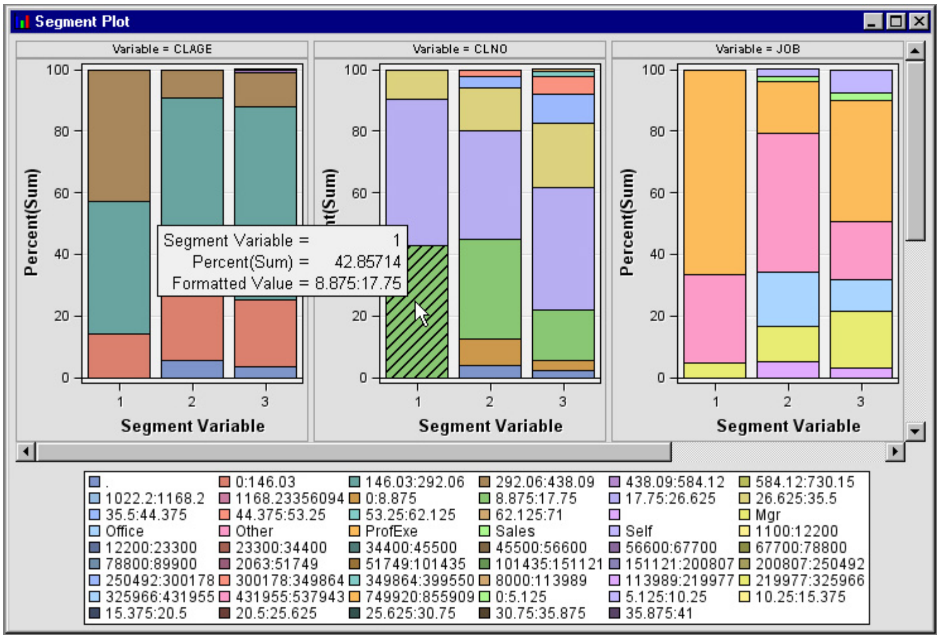
    You can select a node and hide or view a node's children by clicking on the minus or plus.



- **English Rules** — shows in text format the splitting rules that were used to form individual clusters.

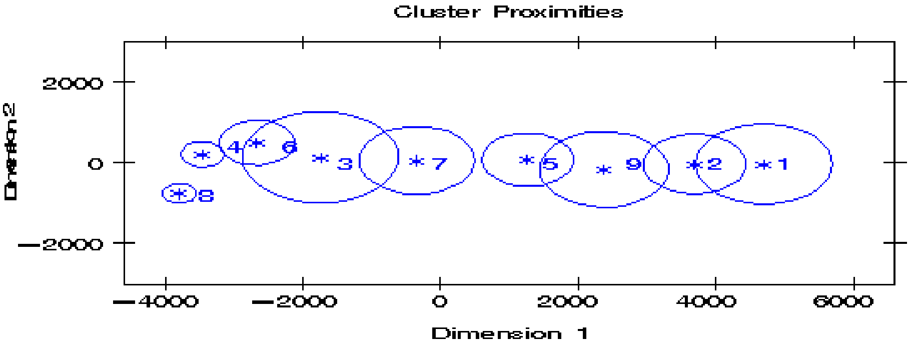
- **Variable Importance** — displays a variables table that contains the number of splitting rules, the number of surrogate rules, and the relative importance of each variable.

- **Segment Plot** — displays a segment plot of the discriminating segment (clustering) variables in the data set. Discriminating variables are variables that have high importance scores. Not all clustering variables will appear in segment plots. Input cluster variables that have a zero importance score are called nondiscriminating variables. Nondiscriminating variables are not included in segment plots because they do not contribute to cluster formation. Position your mouse pointer over a segment within a plot to see the name of the segment variable, the percentage of the segment variable values that are within the segment that you are viewing, and the formatted values used.



Segment Variable = 1
Percent(Sum) = 42.85714
Formatted Value = 8.875:17.75

- **Cluster Distance** — Cluster distance results (distances between cluster means) are summarized in a table of cluster distances and a cluster distance plot.
  - **Plot** — The cluster distance plot provides a graphical representation of the size of each cluster and the relationship among clusters. The graph axes are determined from multidimensional scaling analysis, using a matrix of distances between cluster means as input. The asterisks are the cluster centers, and the circles represent the cluster radii. A cluster that contains only one case is displayed as an asterisk. The radius of each cluster depends on the most distant case in that cluster, and cases might not be uniformly distributed within clusters. Therefore, it might appear that clusters overlap, but in fact, each case is assigned to only one cluster. The distance among the clusters is based on the criteria that are specified to construct the clusters.



  - **Table** — The cluster distance table for n clusters contains n rows and n columns. Each cell in the table displays the Euclidean distance between the centroids of the clusters from the row and column.
- **Table** — Displays a table that contains the underlying data used to produce a chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — Use the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The **Graph Wizard** menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Coding of the Class Variables in the Cluster Node

To incorporate the class variables into the analysis, the **Cluster** node codes the class variables as follows:

- **Binary** — one dummy variable is created. It contains a value of 0 or 1. For example, a binary variable named GENDER is coded as follows:

GENDER	Dummy PURCHASE
Female	0
Male	1

- **Nominal** — one dummy variable is created per level that contains a value of 0 or 1. For example, a nominal variable named REGION is expanded to create the dummy variables as follows:

REGION	Dummy East	Dummy North	Dummy South	Dummy West
East	1	0	0	0
North	0	1	0	0
South	0	0	1	0
West	0	0	0	1

- **Ordinal** — one dummy variable is created for each ordinal input. The smallest ordered value is mapped to 1, the next smallest ordered value is mapped to 2, and so on. For example, an ordinal variable named RISK is coded as follows:

RISK(C)	RISK(N)	Dummy Risk
A	1	1
B	2	2
C	5	3

Note: C and N refer to a character and a numeric ordinal variable, respectively. Interval variables do not require dummy variable coding.

Chapter 31

DMDB Node

DMDB Node

Overview of the DMDB Node

The Data Mining Database (DMDB) node belongs to the Explore group of the SAS SEMMA (Sample, Explore, Modify, Model, Assess) data mining process. The DMDB node creates a data mining database that provides summary statistics and factor-level information for class and interval variables in the imported data set.

In Enterprise Miner 4.3, the DMDB database optimized the performance of the Variable Selection, Tree, Neural Network, and Regression nodes by reducing the number of passes through the data that the analytical engine needed to make when running a process flow diagram. Improvements to the Enterprise Miner 6.2 software eliminated the need to use the DMDB node to optimize the performance of nodes, but the DMDB database can still provide quick summary statistics for class and interval variables at a given point in a process flow diagram.

DMDB Node and Missing Data Set Values

If an observation contains missing values, the DMDB node does not exclude them from the DMDB summary report.

Using the DMDB Node

The DMDB node can follow any Enterprise Miner node.

DMDB Node Properties

DMDB Node General Properties

The following general properties are associated with the DMDB node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first DMDB node that is added to a diagram will have a Node ID of DMDB. The second DMDB node added to a diagram will have a Node ID of DMDB2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — DMDB window. The Imported Data — DMDB window contains a list of the ports that provide data sources to the DMDB node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — DMDB window. The Exported Data — DMDB window contains a list of the output data ports that the DMDB node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

DMDB Node Train Properties

The following train properties are associated with the DMDB node:

- **Variables** — Use the Variables property to view variable information, and change variable values using the DMDB node. Select the ... button to open a window containing the variables table. You can specify the Use and Report value of a variable, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. By default, columns for the Name, Use, Report, Role, and Level of a variable are displayed.

You can modify these values, and add additional columns using the following options:

- **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
- **Reset** — Changes metadata back to its state before use of the Apply button.
- **Label** — Adds a column for a label for each variable.
- **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
- **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
- **Statistics** — Adds statistics metadata for each variable.
- **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

DMDB Node Train Properties: General Properties

- **Interval Variables** — Use the Interval Variables property of the DMDB node to specify whether to compute summary statistics for interval variables in the imported data set. The default setting for the Interval Variables property of the DMDB node is Yes.
- **Class Variables** — Use the Class Variables property of the DMDB node to specify whether to compute summary statistics for class variables in the imported data set. The default setting for the Class Variables property of the DMDB node is Yes.
- **Maximum Number of Levels Cutoff** — Use the Maximum Number of Levels Cutoff property to specify the maximum number of levels to be reported for a variable. If the DMDB node reaches the specified maximum number of levels cutoff value for a given variable, DMDB reports the maximum value, and then begins processing the next variable. The Maximum Number of Levels Cutoff property accepts positive integer values, and the default setting is 25.

DMDB Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

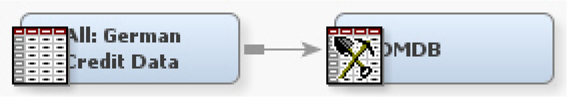
DMDB Node Results

You can open the Results window of the DMDB node after a successful run by selecting the Results button in the Run Status — Run completed window, or by right-clicking the node in the Diagram Workspace and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the Results window main menu to view the following information in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the DMDB node properties panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the DMDB node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the DMDB node. The table includes columns to see the variable name, use, report, role, and level.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the DMDB node run.
  - **Output** — the SAS output of the DMDB node run.
  - **Flow Code** — Flow Code is not available for the DMDB node.
- **Scoring**
  - **SAS Code** — the DMDB node does not generate SAS Code. The SAS Code menu item is dimmed and unavailable in the DMDB Results window.
  - **PMML Code** — the DMDB node does not generate PMML code.
- **Summary Statistics**
  - **Interval Variables** — creates a tabular report that contains the following values for each interval variable: Missing, N, Minimum, Maximum, Mean, Std. Deviation, Skewness, and Kurtosis.
  - **Class Variables** — creates a tabular report that contains the following values for each class variable: Type, Number of Levels, Missing.

DMDB Node Example



The following is a simple example that uses the DMDB node to generate typical summary statistics:

1. Use the Create Data Source Wizard to add the example German Credit data source SAMPSIO.DMAGECR to the Data Sources folder of your Enterprise Miner Project Tree.

   - Choose **SAS Table** as your metadata source.
   - Type in the SAS Table name `SAMPSIO.DMAGECR`.
   - Select the **Advanced Metadata Advisor**.
   - Use the Columns Metadata table to assign the binary variable good_bad the role of **Target**.
   - Select **No** when prompts ask whether you want to perform Decision Processing.
   - Save the Data Source in the **Raw** or **Train** role (user's choice) and exit the Create Data Source Wizard.

2. If necessary, use the Create Diagram function to create and open a diagram to use for the example process flow.
3. Drag the new All: German Credit Data data source from the Data Sources folder of your Enterprise Miner Project Tree onto the Diagram Workspace.
4. Drag a DMDB icon from the Explore tab of the node tools bar onto the Diagram Workspace. Connect the All: German Credit Data data source to the DMDB node.
5. The DMDB node is configured by default to generate summary statistics for both class and interval variables. Leave the DMDB node in its default configuration.
6. Right-click the DMDB node and choose **Run** from the pop-up menu.
7. When the run completes, select the **Results** button in the Run Status window.
8. The Output section of the DMDB Results window displays the DMDB summary reports for the class and interval variables at that point in the process flow diagram.

Chapter 32
Graph Explore Node

Graph Explore Node

Overview of the Graph Explore Node

The Graph Explore node is on the Explore tab of the Enterprise Miner tools bar. The Graph Explore node is an advanced visualization tool that enables you to explore large volumes of data graphically to uncover patterns and trends and to reveal extreme values in the database. This node is a partial replacement for SAS Insight and Distribution Explorer nodes from Enterprise Miner 4.3. The node creates a runtime sample of the input data source. You use the Graph Explore node to interactively explore and analyze your data using graphs. Your exploratory graphs are persisted when the Graph Explore Results window is closed. When you re-open the Graph Explore Results window the persisted graphs are recreated.

You can analyze univariate distributions, investigate multivariate distributions, create scatter and box plots, constellation and 3D charts, and so on. If the Graph Explore node follows a node that exports a data set in the process flow, then it uses either a sample (default) or the entire data set as input. The resulting plot is fully interactive — you can rotate a chart to different angles and move it anywhere on the screen to obtain different perspectives on the data. You can also probe the data by positioning the cursor over a particular bar within the chart. A text window displays the values that correspond to that bar. You may also want to use the node downstream in the process flow to perform tasks, such as creating a chart of the predicted values from a model developed with one of the modeling nodes.

The Graph Explore node must have at least one preceding node that exports one or more data sets, such as an Input Data Source, Sampling, Neural Network, or Score node.

Graph Explore Node Properties

Graph Explore Node General Properties

The following general properties are associated with the Graph Explore node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Graph Explore node that is added to a diagram will have a Node ID of GrfExpl. The second Graph Explore node added to a diagram will have a Node ID of GrfExpl2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Graph Explore window. The Imported Data — Graph Explore window contains a list of the ports that provide data sources to the Graph Explore node. Select the ... button to the right of the Imported Data property to open a table of the imported data

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Graph Explore window. The Exported Data — Graph Explore window contains a list of the output data ports that the Graph Explore node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Graph Explore Node Train Properties

The following train properties are associated with the Graph Explore node:

- **Variables** — Select the ... button to open the Variables — Graph Explore table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use, Report, and Sample Role variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

The following buttons and check boxes provide additional options to view and modify variable metadata:

- **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
- **Reset** — Changes metadata back to its state before use of the Apply button.
- **Label** — Adds a column for a label for each variable.
- **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
- **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
- **Statistics** — Adds statistics metadata for each variable.
- **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

Graph Explore Node Train Properties: Sample

- **Method** — Use the Method property to specify the sample method for graphical displays.
  - **Default** — Specifies the default graphical display.
  - **First N** — Use the First N property to choose the first N observations from your input data source. First N is the default method.
  - **Random** — Use the Random property to specify that sample observations are chosen at random. Each observation in the data set (population) has the same probability of being selected for the sample, independently of the other observations that happen to fall into the sample.
  - **Stratify** — Use the Stratify property to generate a stratified sample. If there are class target variables or segment variables, they will be used to generate the stratified sample.
- **Size** — Use the Size property to specify the number of observations for graphical displays.
  - **Default** — Use the Default property to load a sample consisting of 2000 observations.
  - **Max** — Use the Max property to specify the maximum number of observations that can be downloaded from your input data.
- **Random Seed** — Use the Random Seed property to specify the value that is used to generate the sample.

Graph Explore Node Report Properties

The following properties are associated with the Graph Explore node report:

- **Target** — Use the Target property to specify whether you want to generate a distribution graph of the target variables.
- **Group by Target** — Use the Group by Target property to specify whether to generate plots of the grouping variables by the target variable or plots of the segment variables by the target variable.

Graph Explore Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Graph Explore Node Results

You can open the Results window of the Graph Explore Node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Graph Explore Node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Graph Explore Node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Graph Explore Node. The table includes columns to see the variable Name, Use, Report, Sample Role, Role, and Level.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Graph Explore Node run.
  - **Output** — the SAS output of the Graph Explore Node run.
  - **Flow Code** — the SAS code used to produce the output that the Graph Explore Node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the Graph Explore Node does not generate SAS Code. The SAS Code menu item is dimmed and unavailable in the Graph Explore Results window.
  - **PMML Code** — the Graph Explore Node does not generate PMML code.
- **Data**
  - **Sample Table** — a table of the sample from the input data source.

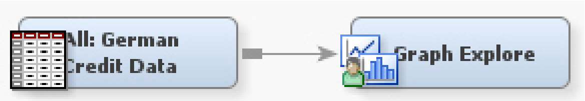
- **Target** — opens a bar chart of frequencies of the target variable.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Select a Chart Type wizard to modify an existing Results plot or create a Results plot of your own. For more information about the available plots, see the Creating Graphs section in the Enterprise Miner User Interface help.

Graph Explore Node Example

Perform the following steps to create the data source All: German Credit Data for use in this example:

1. From the main menu, select: **File** ⇨ **New** ⇨ **Data Source**.
2. Select SAS Table as the metadata source and click **Next**.
3. When prompted to Select a SAS Table, enter **`SAMPSIO.DMAGECR`** in the input field and click **Next**.
4. In the Table Properties window, click **Next**.
5. In the Metadata Advisor Options window, select the Advanced advisor. Click **Next**.
6. In the Column Metadata window, set the role of the variable good_bad to **Target**. Click **Next**.
7. If the Decision Processing window opens, select **No** and click **Next**.
8. In the Data Source Attributes window, leave the data source role as **Raw** and click **Next**.
9. In the Summary window, click **Finish**.

Create a new process flow diagram, then drag the All: German Credit Data data source onto the diagram workspace. Drag a Graph Explore node from the Explore tab of the node toolbar and connect the All: German Credit Data data source node to the Graph Explore node.



Right-click the Graph Explore node, then select **Run** to run the process flow diagram. When the Run Status window indicates that the run is completed, select the **Results** button to open the Graph Explore Results window.

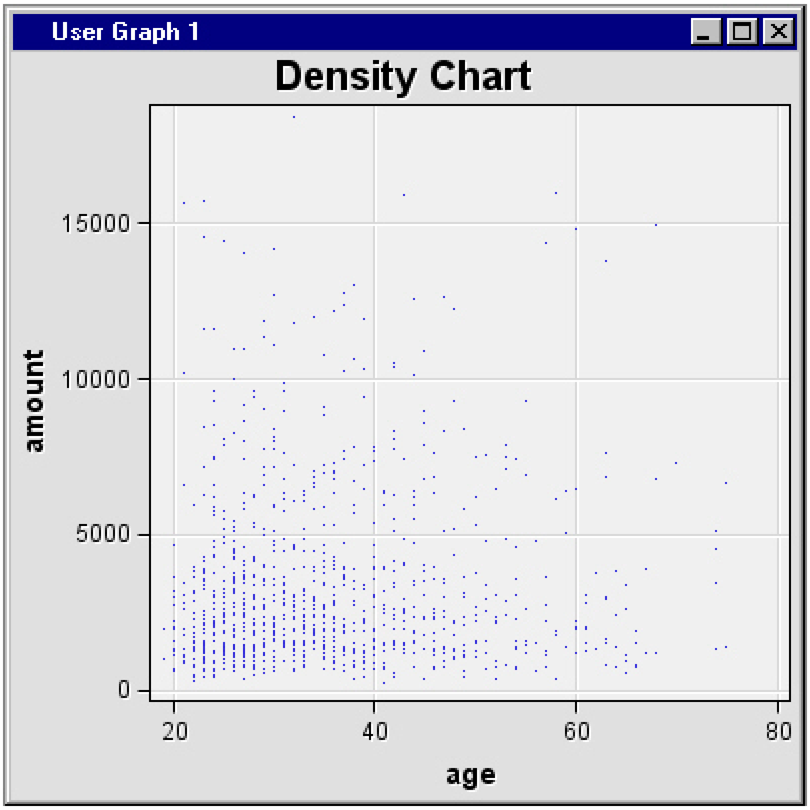
Select the Sample Table and click **View** ⇨ **Plot** from the Results menu. The Select a Chart Type window opens.

You can choose from any of the following chart types:

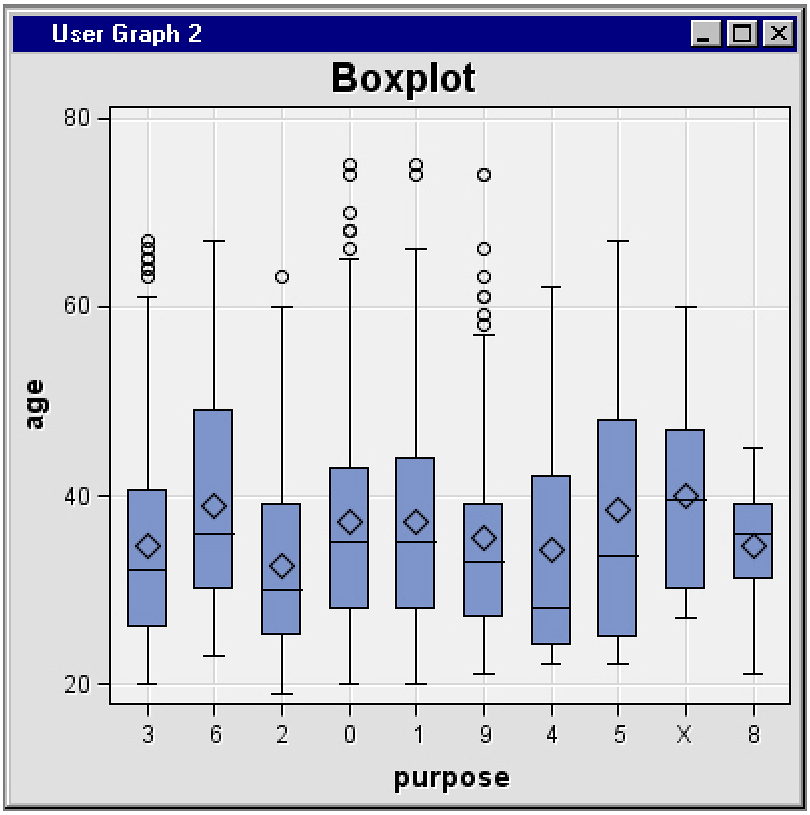
- Scatter
- Line
- Histogram
- Density
- Box

- Tables
- Matrix
- Lattice
- Parallel Axis
- Constellation
- 3D Charts
- Contour
- Bar
- Pie
- Needle
- Vector
- Band

Select **Density** from the Select a Chart Type window. Click the last icon that shows the distribution of two variables plotted against each other. Click **Next**. On the Select Chart Roles window, click the **Use default assignments** button. The variable Age is assigned a role of X and Amount is assigned a role of Y. Click **Next**. Do not specify anything in the Data Where Clause window. Click **Next**. On the Chart Titles window, type Density Chart on the Title field. Click **Next**. Click **Finish**.

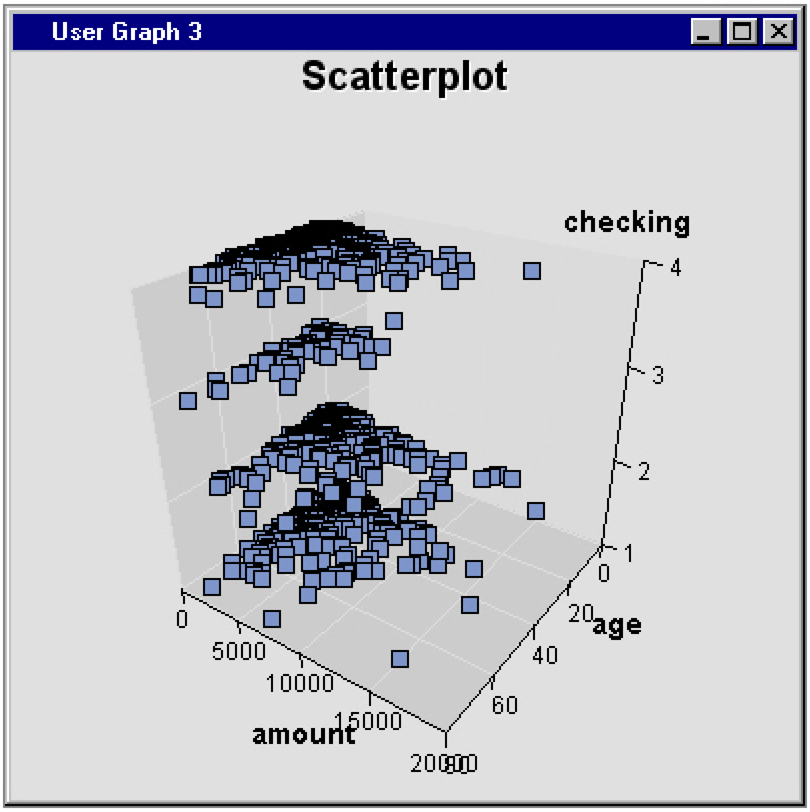


The following is another chart that shows a box plot of the variables Age vs Purpose. Select the box corresponding to Purpose = 1. The corresponding job, minimum whisker, first quartile, median, third quartile, maximum whisker and mean values are displayed on the hover help.

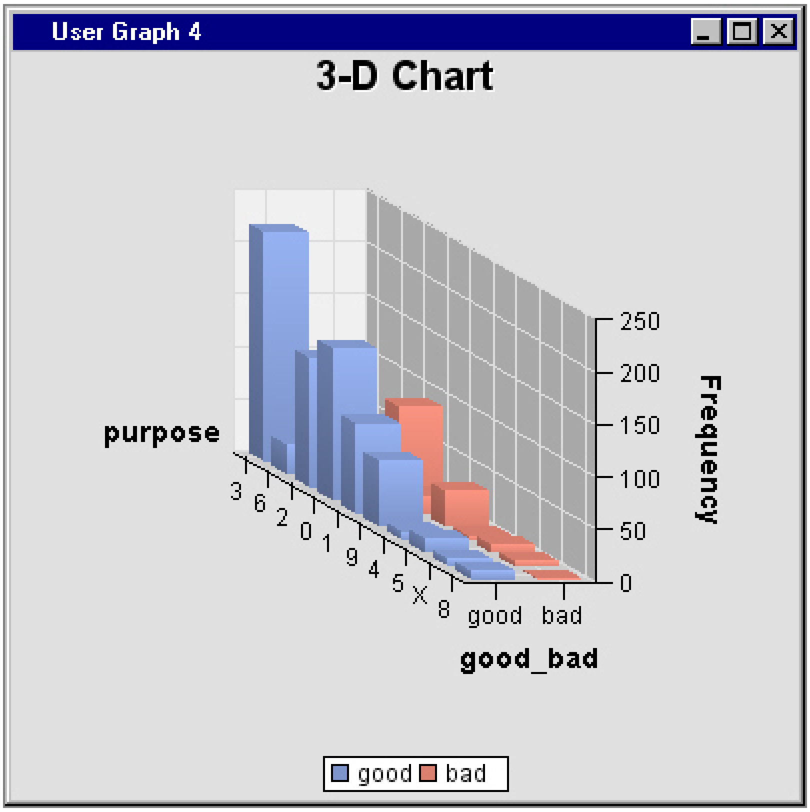


Select the Sample Table from the Graph Explore Results window and click **View** ⇨ **Plot** from the Results menu. The Select a Chart Type opens. Select Scatter and Scatter plot in 3 dimensions. Click **Next**. The Select Chart Roles window opens. Click **Use default assignments** to set the Age, Amount and Checking variables to roles X, Y and Z respectively. Click **Next**. The Data Where Clause window opens.

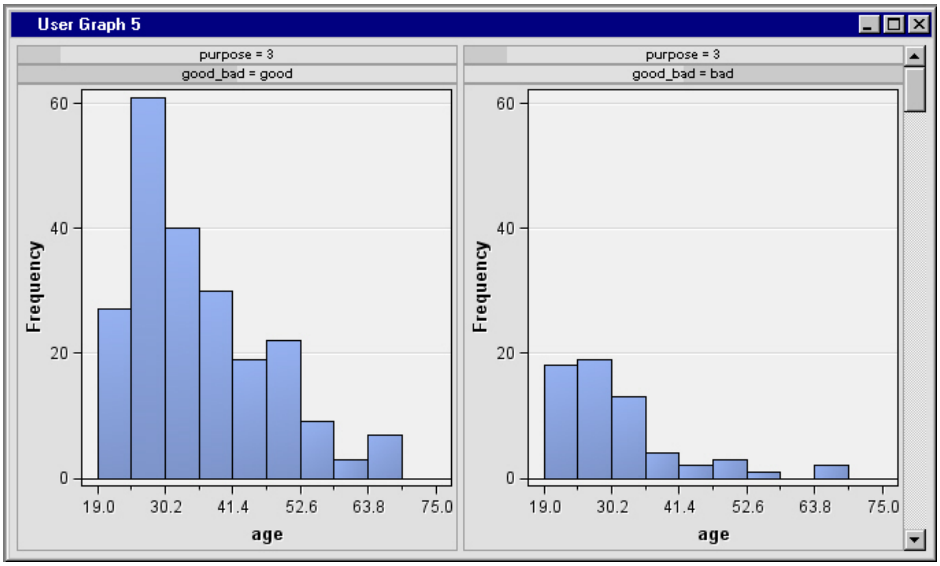
Click **Next**. The Chart Titles window opens. Specify any Title, Footnote or X and Y Axis Labels. Click **Next**. The Chart Legends window opens. Click **Finish**. The 3-D scatter plot opens. Select any point on the graph and note that the corresponding row in the Sample Table is highlighted.



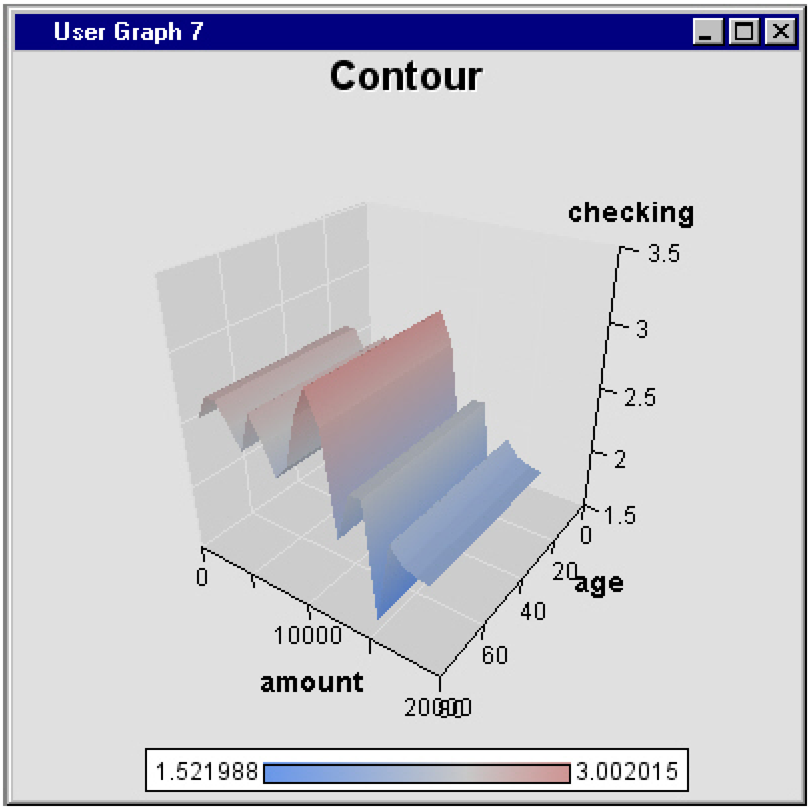
Use a 3-D Chart, and bar chart with both a category and series variable. Use default role assignments to set the GOOD_BAD variable's role to Category and PURPOSE variable to Series . Right-click inside the 3-D graph and select Graph Properties. The Properties window opens. Select Bar on the left pane and select the Color by radio button then select Category from the drop-down list. Click **OK**. Right-click inside the 3-D graph and select **Action Mode** ⇨ **Rotate**. Drag the graph to rotate.



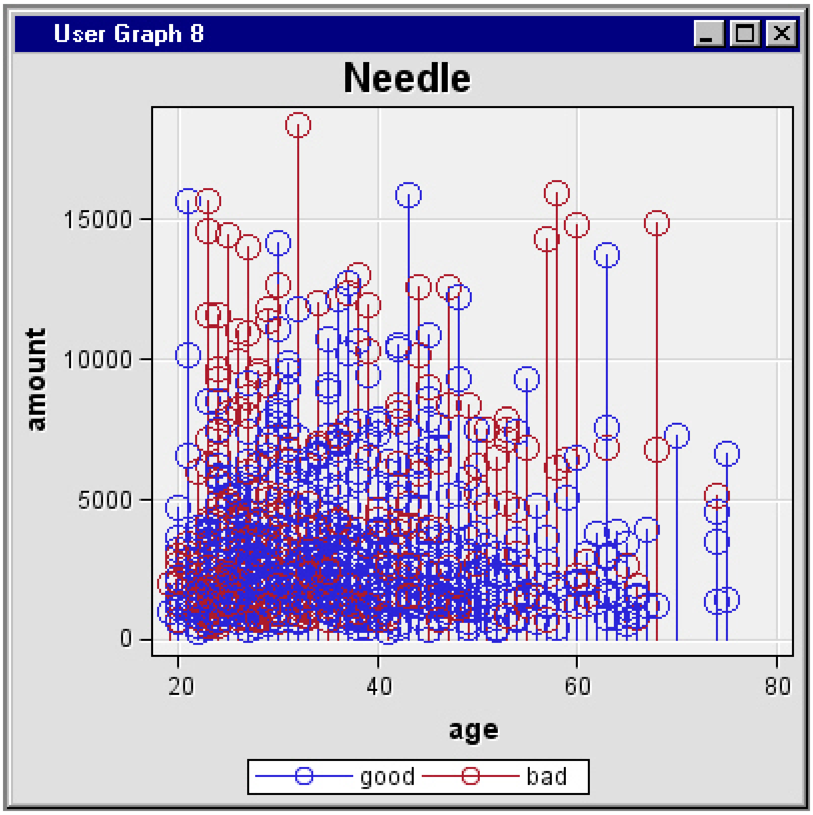
From the Results window, use a lattice histogram with default variable roles to display a plot similar to the following:



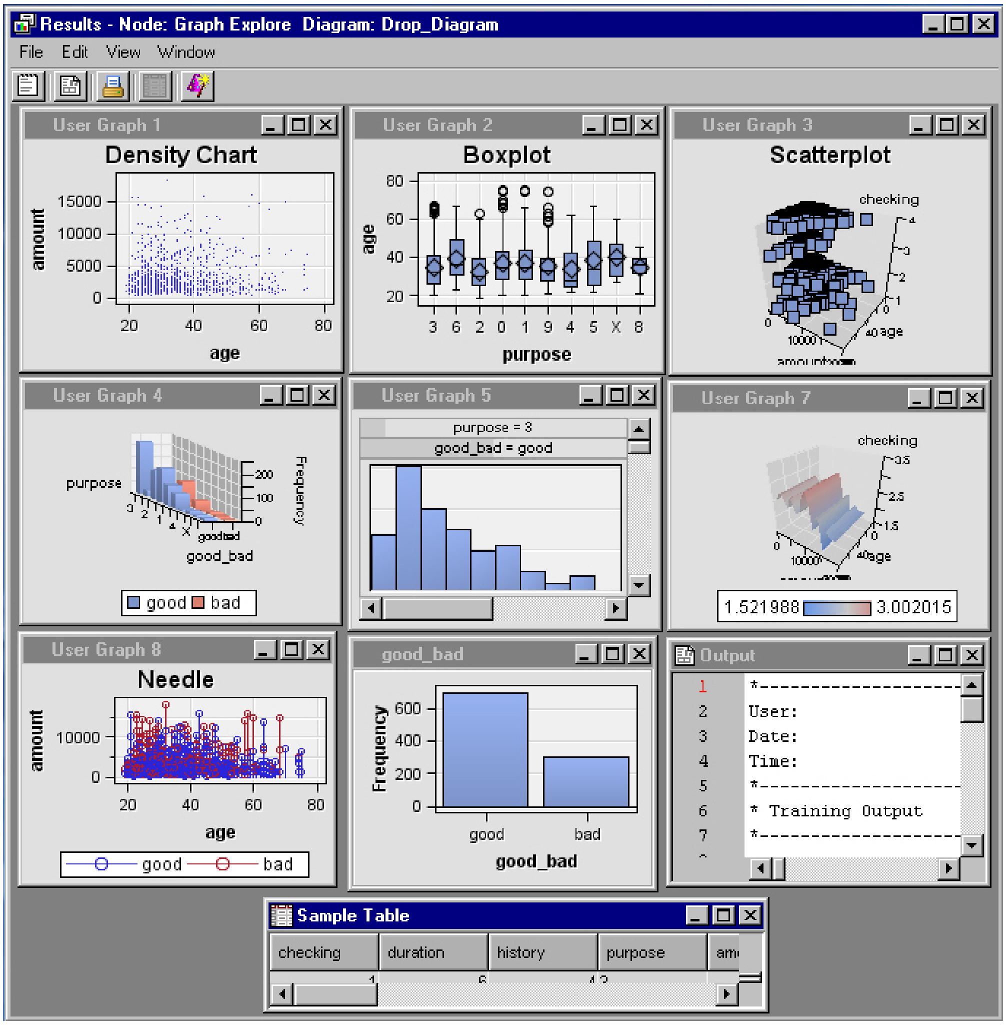
Use the 3D Response Surface Plot setting (the leftmost icon of the 3D charts tools) to create a 3D response surface plot with default variable roles to display a plot similar to the following:



Use a needle with default role assignments. Specify the GOOD_BAD variable with a role of Group.



Close the Results window. Any open graphs are persisted. When you open the Results window again, your exploratory graphs are recreated.



```
*-------------------
User:
Date:
Time:
*-------------------
* Training Output
*-------------------
```

checking	duration	history	purpose	am

Chapter 33

Link Analysis Node

Link Analysis Node

Overview of the Link Analysis Node

Link analysis is the process of discovering and examining the connections between items in a complex system. Analysts typically consider both tabular and graphical representations of the associations, sequences, and networks that exist within a system. They try to discover patterns of activity that can be used to derive useful conclusions. Some applications include forms of fraud detection, criminal network conspiracies, telephone traffic patterns, website structure and usage, database visualization, and social network analysis.

In SAS Enterprise Miner, the **Link Analysis** node transforms data from different sources into a data model that can be graphed. The node provides centrality measures derived from the graph, and performs item-cluster detection for certain types of data. The item-cluster detection information is used for segmentation for scoring purpose. The node also provides recommendation tables for transactional input data.

Input Data Requirements for the Link Analysis Node

The **Link Analysis** node accepts both transactional data and training or raw data. Set the data set **Role** to **Transaction** to specify that the input data is transactional. To specify training or raw data, use either **Train** or **Raw** as the variable role.

When the input data set is transactional, two item association rules or sequence rules are generated, based on the existence of the sequence variable. The two item association rules can be treated as either directed links or undirected links between the two items. For undirected link graphs, the links are analyzed using centrality measures to detect item-clusters or similar items. You can export either a recommendation table or segmentation information based on your preference and the presence of a sequence variable.

A transactional input data set requires one ID variable and one target variable. A sequence variable and a frequency variable are optional for transactional data.

If the input data set is training or raw, the **Link Analysis** node converts it to transactional data first. Each interval variable is binned, and each bin is treated as an item. Each level of a nominal variable is treated as an item. Each observation is considered a basket that contains all of the items. The same analysis that is performed on transactional data is performed on the transformed data. However, no sequence variable is applied to the transformed data.

A training or raw input data set must contain at least two variables. One of these variables is a target variable.

Note that item-cluster detection cannot be applied to a data set that has a sequence variable. The graph that is generated from the sequential transaction data is directed, but item-cluster detection requires undirected graphs.

Link Analysis Node Properties

Link Analysis Node General Properties

The following general properties are associated with the **Link Analysis** node:

- **Node ID** — displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Link Analysis** node that is added to a diagram will have a Node ID of LinkAnalysis. The second **Link Analysis** node that is added to the diagram will have a Node ID of LinkAnalysis2.
- **Imported Data** — provides access to the Imported Data — Link Analysis window. The Imported Data — Link Analysis window contains a list of the ports that provide data sources to the **Link Analysis** node. Click the ... button to the right of the **Imported Data** property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and do the following:

  - Click **Browse** to open a window where you can browse the data set.
  - Click **Explore** to open the Explore window where you can sample and plot the data.
  - Click **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — provides access to the Exported Data — Link Analysis window. The Exported Data — Link Analysis window contains a list of the output data ports that the **Link Analysis** node creates data for when it runs. Click the ... button to the right of the **Exported Data** property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and do the following:

  - Click **Browse** to open a window where you can browse the data set.

  - Click **Explore** to open the Explore window where you can sample and plot the data.
  - Click **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.
- **Notes** — Click the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Link Analysis Node Train Properties

The following train properties are associated with the **Link Analysis** node:

- **Variables** — specifies the status for individual variables that are imported into the **Link Analysis** node. Click the ... button to open a window that contains the variables table. You can set the variable status to either **Use** or **Don't Use** in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Minimum Confidence (%)** — specifies the minimum confidence level that a rule must meet to be included in the analysis. Valid values are real numbers between 0 and 1.
- **Association Rule Support Type** — specifies the type of support that is generated for association analysis. You can specify either **Support Count** or **Support Percentage**.
- **Association Support Percentage** — specifies the minimum transaction frequency, as a percentage of all transactions, that is required to support an association.
- **Association Support Count** — specifies the minimum transaction frequency, as a count, that is required to support an association.
- **Sequence Property Settings** — Click the ... button to the right of **Sequence Property Settings** to open the Sequence Property Settings window. The Sequence Property Settings window enables you to set the following properties:
  - **Sequence Support Type** — specifies the type of support that is generated for a sequence analysis. You can specify either a minimum percentage of observations or a minimum number of observations.
  - **Support Percentage** — specifies the minimum transaction frequency, as a percentage of all transactions, that is required to support an association.
  - **Support Count** — specifies the minimum transaction frequency, as a count, that is required to support an association.
  - **Consolidate Time** — specifies whether consecutive visits to a location over a given interval are consolidated into a single visit during the analysis. This option also enables multiple visits on the same day to be consolidated into a single visit. The value specified here must be less than the value of the **Maximum Transaction Duration** property.
  - **Maximum Transaction Duration** — specifies the maximum transaction window length. By default, all possible sequences are identified. The value specified here must be less than the value specified in the **Consolidate Time** property.

Link Analysis Node Train Properties: Binning Properties

- **Binning Method** — specifies the binning method that is used to bin the interval variables. You can specify either **Bucket** or **Quantile**.
- **Number of Bins** — specifies the number of bins that are used for bucket binning of interval variables.

Link Analysis Node Train Properties: Centrality Measures Properties

- **Centrality Measures Settings** — Click the ... button to the right of **Settings** to open the Settings window. The Settings window enables you to set the following properties:
  - **Clustering Coefficient** — specifies whether the clustering coefficient is calculated. The clustering coefficient for a node is the number of links between the nodes within its neighborhood divided by the number of links that could possibly exist between those nodes.
  - **Influence Centrality** — specifies whether influence centrality is calculated. Influence centrality is a generalization of degree centrality that considers the link and node weights of adjacent nodes (C1) in addition to the link weights of nodes that are adjacent to the adjacent nodes (C2). The metric C1 is referred to as the first-order influence centrality, and C2 is referred to as the second-order influence centrality.
  - **Closeness Centrality** — specifies whether closeness centrality is calculated. Closeness centrality is the reciprocal of the average of the shortest paths (geodesic distances) to all other nodes. Closeness can be thought of as a measure of how long it would take information to spread from a given node to other nodes in the network.
  - **Betweenness Centrality** — specifies whether betweenness centrality is calculated. Betweenness centrality counts the number of times a particular node or link occurs on the shortest paths between other nodes. Betweenness centrality can be thought of as a measure of the control that a node or link has over the communication flow in the rest of the network. In this sense, the nodes or links with high betweenness centrality are the gatekeepers of information because of their relative location in the network.
  - **Eigenvector Centrality** — specifies whether eigenvector centrality is calculated. Eigenvector centrality is an extension of degree centrality, in which centrality points are awarded for each neighbor. However, not all neighbors are equally important. Intuitively, a connection to an important node should contribute more to the centrality score than a connection to a less important node. This is the basic idea behind eigenvector centrality. Eigenvector centrality of a node is defined to be proportional to the sum of the scores of all nodes that are connected to it.
  - **Eigenvector Algorithm** — specifies the algorithm that is used to calculate the eigenvector centrality. The algorithms available to compute eigenvector centrality are **AUTOMATIC**, **JACOBI_DAVIDSON**, and **POWER**. **AUTOMATIC**, which is the default, uses the OPTGRAPH procedure to determine which algorithm is used. The **JACOBI_DAVIDSON** option uses a variant of the Jacobi-Davidson algorithm to solve eigen systems. The **POWER** option uses the power method to calculate the eigenvectors. The **POWER** option does not support eigenvector centrality.

  - **Maximum Iterations** — specifies the maximum number of iterations to use for eigenvector calculations. This property limits the amount of computation time spent when convergence is slow. The default value is 5,000.
  - **Hub Centrality** — Hub centrality (Kleinberg, 1998) was originally developed to rank the importance of web pages. Certain web pages are important in the sense that they point to many important pages. Such pages are called hubs. A good hub node is one that points to many good authorities. Hub centrality applies only to directed graphs, so it is calculated only if input data is transactional data with a sequence variable.
  - **Authority Centrality** — Authority centrality (Kleinberg, 1998) was originally developed to rank the importance of web pages. Some web pages are important because they are linked by many important pages. Such pages are referred to as authorities. A good authority node is one that is pointed to by many good hub nodes. Authority centrality applies only to directed graphs, so it is calculated only if input data is transactional data with a sequence variable.

Link Analysis Node Train Properties: Item-cluster Properties

- **Resolution** — specifies the resolution parameter that is used to merge two item-clusters. A larger resolution value produces more item-clusters. Each item-cluster contains a smaller number of nodes. Suppose the resolution value is R. Two item-clusters are merged if the sum of the weights of intercluster links is at least R times the expected value of the same sum if the graph is configured randomly.
- **Item-cluster Detection Settings** — Click the ... button to the right of **Item-cluster Detection Settings** to open the Item-cluster Detection Settings window. The Item-cluster Detection Settings window enables you to set the following properties:
  - **Algorithm** — specifies the algorithm that is used for item-cluster detection. You can select either the **Louvain** algorithm, which is the default, or the **Label Propagation** algorithm.
  - **Link Removal Ratio** — specifies the percentage of small-weight links that are removed around each node neighborhood. A small-weight link is one that is relatively smaller than the weights of the neighboring links. This option can dramatically improve the time required to compute large graphs. Valid values are real numbers between 0 and 100, inclusive. The default value is 10.
  - **Max Iterations** — specifies the maximum number of iterations that are allowed in the algorithm. The default is 20.
  - **Item-cluster Iteration Tolerance** — specifies the tolerance value that is used to stop iterations. The algorithm stops iterations after the percentage modularity gain between two consecutive iterations falls below the specified tolerance value. Valid values are real numbers between 0 and 1, inclusive. The default value is 0.0010.

Link Analysis Node Score Properties: Recommendation Properties

The following score properties are associated with the **Link Analysis** node:

- **Generate Recommendation Table** — specifies whether the Recommendation Table is generated in the Results window. The Recommendation Table recommends items that meet the filtering criteria. This property applies only when the input data is transactional data.

  Note that if no recommendation table is generated, segmentation information is provided in the exported data if that input is not transactional data with a sequence variable. If the Recommendation Table is generated, segmentation information is not provided.

- **Top N** — specifies the number of recommendations to show for each customer. The default value is 1, which implies that the Recommendation Table provides the next best offer.
- **Minimum Confidence** — specifies the minimum confidence level that is required in order for an item to be included in the Recommendation Table.
- **Criterion Relation** — specifies if just one of the filtering criteria or both of the filtering criteria are required for an item to be included in the Recommendation Table. Specify **AND** to require both and **OR** to require either one.
- **Export Recommendation Table** — specifies whether the full recommendation table or the filtered recommendation table is exported.

Link Analysis Node Report Properties: Rules Properties

- **Sort By** — specifies which rules statistic is used to sort the rules table. You can specify **Transaction Count**, **Support**, **Confidence**, **Lift/Pseudolift**, or **Expected Confidence**.
- **Top N Rules** — specifies the number of rules that are shown in the rules statistics bar chart.

Link Analysis Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Link Analysis Node Exported Data

The type of data that is exported by the **Link Analysis** node is determined by both the input data type and the **Generate Recommendation Table** option. The following table summarizes the data that is exported:

Generate Recommendation Table Setting	Training or Raw Data	Transactional Data
Yes	No data is exported.	The recommendation table is exported with a role of **Train**, and the input table is sorted by the ID variable and exported with a role of **Transaction**.
No	The import table is combined with the segmentation information and is exported with a role of **Train**.	If a sequence variable exists, then the input table is sorted by the ID variable and exported with a role of **Transaction**. No score code is generated. Otherwise, the input table is combined with the segmentation information and is exported with a role of either **Train** or **Transaction**.

The Recommendation Table contains rules that are generated by a market basket analysis. The segmentation information is based on item-cluster overlap intensity statistics.

Link Analysis Node Results

You can open the Results window of the **Link Analysis** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration settings for the **Link Analysis** node. The information was captured when the node was last run. Use the **Show Advanced Properties** check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the **Link Analysis** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information about the data set that was submitted to the **Link Analysis** node. The table includes columns for the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **Link Analysis** node run.
  - **Output** — the SAS output of the **Link Analysis** node run.

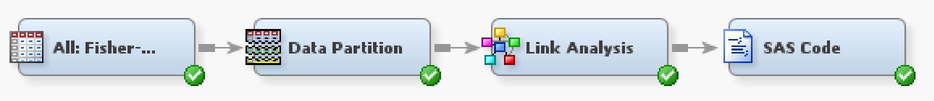
  - **Flow Code** — the SAS code used to produce the output that the **Link Analysis** node passes on to the next node in the process flow diagram.

- **Scoring**
  - **SAS Code** — the SAS code that was created by the **Link Analysis** node. The SAS code can be modified for use outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **Link Analysis** node does not generate PMML code.
- **Plots**
  - **Item-cluster Community Constellation** — displays the connection between each item-cluster to every other item-cluster. The thickness of the link indicates the relative similarity of two item-clusters.
  - **Item-cluster Overlap Plot** — displays the item-clusters as red dots and the nodes as blue dots. The thickness of the link between a node and an item-cluster indicates the relative importance of that node to an item-cluster.
  - **Node Frequency Histogram (but Item-cluster)** — a histogram is created for each item-cluster that enables you to explore the node weight distribution within each item-cluster.
  - **Item-cluster Size Pie Chart** — provides a pie chart that displays the relative sizes of each item-cluster.
  - **Node Frequency Histogram** — displays the weight assigned to each node.
  - **Link Frequency Histogram** — displays the frequency of link transaction counts.
  - **Unweighted Centrality Measure Histogram** — plots the unweighted centrality measures.
  - **Weighted Centrality Measure Histogram** — plots the weighted centrality measures.
  - **Rules Statistics by Rule ID** — plots the rule statistics for each rule.
  - **Exploratory Plot for Transactional Data** — a constellation plot that enables you to view the links between the ID variable and the target or sequence variables.
  - **Items Constellation Plot** — explores the relationships between nodes. The color of each node indicates the item-cluster that that node belongs to. The width of the line indicates the relative strength between two nodes.
- **Recommendations**
  - **Recommendation Table** — displays the full recommendation table.
  - **Filtered Recommendation Table** — displays the filtered recommendation table. Results are filtered based on the criteria specified in the score properties.
- **Table** — displays a table that contains the underlying data used to produce a chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — You can use the Graph wizard to modify an existing Results plot or to create a Results plot of your own.

Link Analysis Node Examples

Example 1: Clustering and Scoring a Raw Data Set

In this example, you use the **Link Analysis** node to cluster a non-transactional input data set. You then score the results of the **Link Analysis** node with a SAS Code node. When finished, your process flow diagram will resemble the image below.



This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

This example uses the DMAIRIS data set. To add the DMAIRIS data set to your diagram, follow these steps:

1. In the Project Panel, right-click the **Data Sources** folder and click **Create Data Source**. This opens the Data Source Wizard — Metadata Source window.

2. In the Data Source Wizard — Metadata Source window, ensure that the **Source** is set to **SAS Table**. Click **Next**.

3. In the Data Source Wizard — Select a SAS Table window, enter `SAMPSIO.DMAIRIS` in the **Table** field. Click **Next**.

4. In the Data Source Wizard — Table Information window, click **Next**.

5. In the Data Source Wizard — Metadata Advisor Options window, click **Next**.

6. In the Data Source Wizard — Column Metadata window, set the **Role** of the variable SPECIES to **Target**. Click **Next**.

7. Click **Next** in the next three windows to get to the Data Source Wizard — Summary window. Click **Finish**.

   The **All: Fisher-Anderson Iris Data** data set should now be in your Project Panel.

8. From the Project Panel, drag **All: Fisher-Anderson Iris Data** to your diagram.

Before you can perform an analysis of the DMAIRIS data set, you need to partition the data set.

1. From the **Sample** tab, drag a **Data Partition** node to your diagram workspace.

2. In the **Data Set Allocations** properties subgroup, enter the following values:

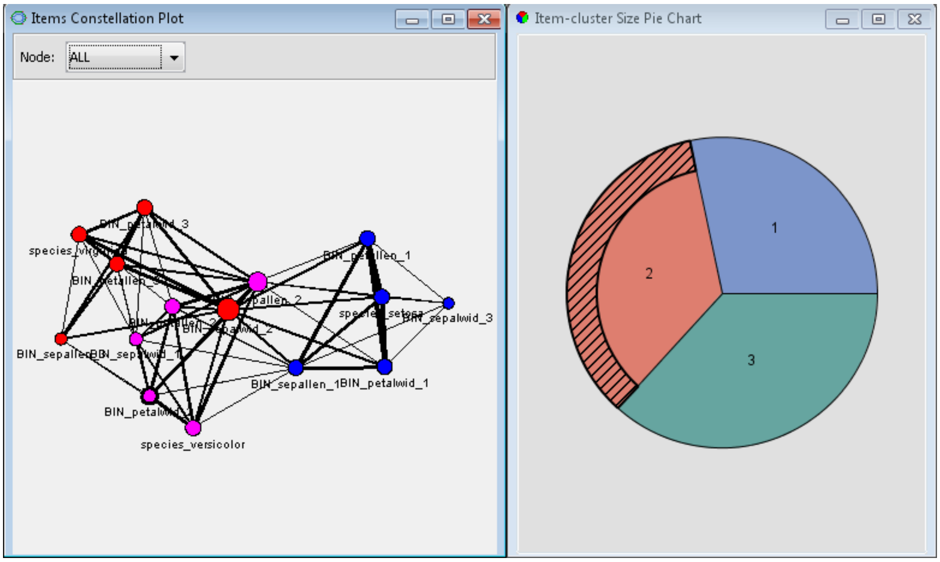
   - For the **Training** property, enter `66`.
   - For the **Validation** property, enter `34`.
   - For the **Test** property, enter `0`.

3. Connect the **All: Fisher-Anderson Iris Data** node to the **Data Partition** node.

Now, you are ready to add the **Link Analysis** node to your process flow diagram. From the **Explore** tab, drag a **Link Analysis** node to your diagram workspace. In the **Binning Properties** subgroup, set the **Number of Bins** property to `3`. Connect the **Data Partition** node to the **Link Analysis** node.

Right-click the **Link Analysis** node and click **Run**. In the Confirmation window, click **Yes**. After the **Link Analysis** node has successfully run, click **Results** in the Run Status window.

View the Item-Cluster Size Pie Chart. This graph displays the relative size of each item-cluster in the DMAIRIS data set. Note that three item-clusters were created. These three item-clusters are represented in the Items Constellation Plot with three distinct node colors. These graphs are linked, which means that a selection in one graph will translate to a selection in the other.



To view the Items Constellation Plot with directed links, right-click on the constellation plot and click **Graph Properties**. In the Properties — Constellation window, click **Links** in the properties list. Click the **Show Directed Links** property so that the box next to the property is checked. Click **OK**. The Items Constellation Plot now displays directed links.

Close the Results window.

The last step in this example is to score the output data set. First, you need to determine the two-level table name for the training and validation data sets that are exported by the **Link Analysis** node. To do this, select the **Link Analysis** node. Click the ellipsis button in the **Exported Data** property. In this example, the **Link Analysis** node exports training data and validation data. These tables are named according to the convention <diagramReference>.Link_TRAIN and <diagramReference>.Link_Validate. Your value for <diagramReference> will vary based on the number of diagrams that you have open. Record the value given here because you will need it in the next step.

Note: The exported table names are also based on the Node ID. So if you have more than one Link Analysis node in your diagram, your table name might differ from what is given above. If this is the case, remember to adjust for this situation in the code below.

From the **Utility** tab, drag a **SAS Code** node to your diagram workspace. Connect the **Link Analysis** node to the **SAS Code** node. Select the **SAS Code** node and click the ellipsis button in the **Code Editor** property. Enter the following code into the **Training Code** field:

```
proc freq data=<diagramReference>.Link_TRAIN; tables _SEGMENT_*species / nopercent ;
title 'Frequency table for Training Data';
proc freq data=<diagramReference>.Link_VALIDATE; tables _SEGMENT_*species / nopercent ;
title 'Frequency table for Validation Data';
```

```
run;
title;
```

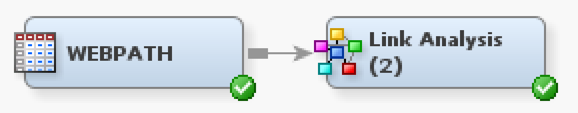
Ensure that you replace **`<diagramReference>`** with the value that you recorded earlier. You might encounter difficulty copying the quotation marks and brackets into the **Training Code** field.

Click the icon in the Training Code window toolbar. Click the icon in the window toolbar. Click the icon in the toolbar. In the Confirmation window, click **Yes**. In the Run Status window, click **Results**.

In the Output window, scroll down to the **Frequency table for Training Data** to view the classification rates for each item-cluster. Notice that in the training data, only two observations from item-cluster two were misclassified into item-cluster three. The validation data was perfectly classified. Close the Results window.

Example 2: Examining Web Log Data

This example uses the WEBPATH data set to explore the relationships that exist in a web log. When finished, your process flow diagram will resemble the image below.



This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

To add the WEBPATH data set to your diagram, follow these steps:

1. In the Project Panel, right-click the **Data Sources** folder and click **Create Data Source**. This opens the Data Source Wizard — Metadata Source.

2. In the Data Source Wizard — Metadata Source window, ensure that the **Source** is set to **SAS Table**. Click **Next**.

3. In the Data Source Wizard — Select a SAS Table window, enter **`SAMPSIO.WEBPATH`** in the **Table** field. Click **Next**.

4. In the Data Source Wizard — Table Information window, click **Next**.

5. In the Data Source Wizard — Metadata Advisor Options window, click **Next**.

6. In the Data Source Wizard — Column Metadata window, set the following variable roles:

   - Set the value of REQUESTED_FILE to **Target**.

   - Set the value of SESSION_SEQUENCE to **Sequence**.

   - Set the value of REFERRER to **Input**.

   - Set the value of SESSION_ID to **ID**.

   Click **Next**.

7. Click **Next** in the next two windows to get to the Data Source Wizard — Data Source Attributes window. Set the value of the **Role** property to **Transaction**. Click **Next**.

8. On the Data Source Wizard — Summary page, click **Finish**. The **WEBPATH** data set should now be in your Project Panel.
9. From the Project Panel, drag the **WEBPATH** to your diagram.

You are now ready to add a Link Analysis node to your process flow diagram. Because the input data set is transactional, the **Link Analysis** node will perform a directional analysis. In a directional analysis, the link for moving from Page A to Page B is different from the link for moving from Page B to Page A. The standard link analysis considers these two actions identical.

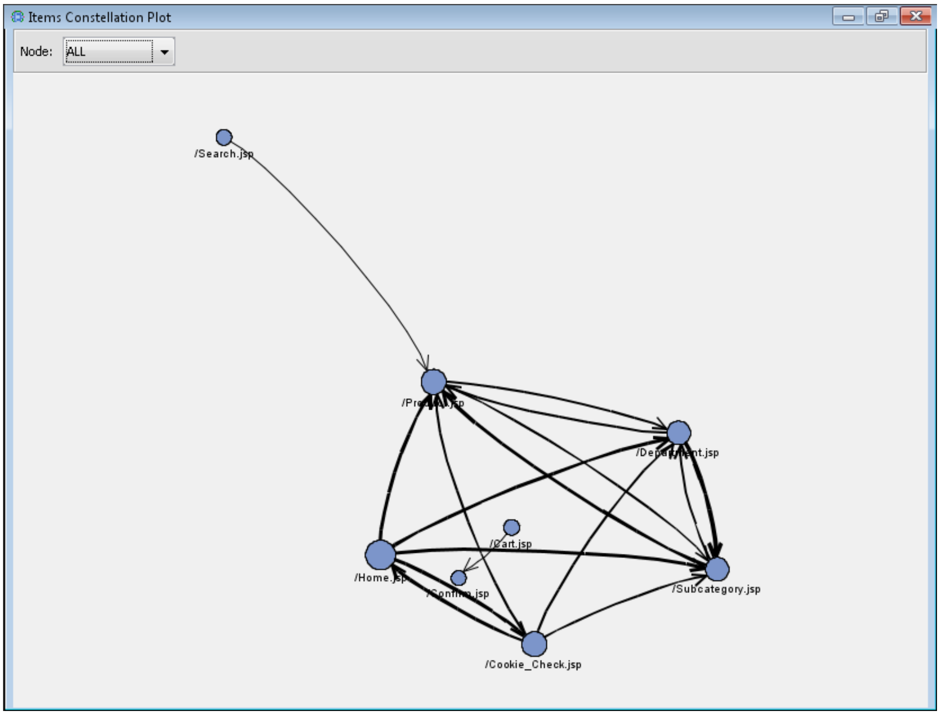
From the **Explore** tab, drag a **Link Analysis** node to your diagram workspace.

Make the following changes to the **Link Analysis** node properties:

- Set the value of the **Minimum Confidence (%)** property to `50`.
- Set the value of the **Association Rule Support Type** property to **Count**.
- Set the value of the **Support Count** property to `50`.
- Set the value of the **Generate Recommendation Table** property to **No**.

Right-click the **Link Analysis** node and select **Run**. In the Confirmation window, click **Yes**. In the Run Status window, click **Results**.

Maximize the Items Constellation Plot window.



Items Constellation Plot

Node: ALL

Notice that the links between nodes contain directed arrows that indicate the referring and the target page. In this example, everybody that started on /Search.jsp traveled to /Product.jsp. However, people also visited /Product.jsp from /Home.jsp, /Cookie_Check.jsp, /Subcategory.jsp, and /Department.jsp. Nobody left /Product.jsp for /Home.jsp.

Close the Results window.

References

Kleinberg, Jon M. 1998. “Authoritative Sources in a Hyperlinked Environment.” *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, California, 668–677.

Chapter 34
Market Basket Node

Market Basket Node

Overview of the Market Basket Node

The Market Basket node is on the Explore tab of the Enterprise Miner tools bar. The Market Basket node performs association rule mining over transaction data in conjunction with item taxonomy. Transaction data contain sales transaction records with details about items bought by customers. Market basket analysis uses the information from the transaction data to give you insight about which products tend to be purchased together. This information or patterns can be used to change store layouts, which products to put on sale or when to issue coupons or some other profitable course of action.

The Market Basket node is useful in retail marketing scenarios involving thousands of distinct items where the items are grouped into subcategories, categories, departments, and so on, called item taxonomy. The Market Basket node uses the taxonomy data and generates rules at multiple levels in the taxonomy. Generalized association rule mining solves many of the problems in simple market basket analysis as explained below.

One of the problems with simple market basket analysis performed over the transaction or point-of-sale data is that significant and potentially useful associations often go undetected. For example, consider a supermarket selling different types of breads and a wide selection of wines. Further, assume that a large number of customers buy some type of bread with some type of wine. However, market basket analysis performed using the Association node may not find any rules linking bread and wine. The Association node computes the support for the combination of specific types of bread and specific types of wine. However, none of these supports may be large enough to generate a rule.

On the other hand, by combining the item taxonomy with the transaction data, the Market Basket node computes the support for the combination of any type of bread and any type of wine in addition to specific types of bread and specific types of wine. Note that reducing the minimum support is often not a solution to this problem since it may lead to the generation of a large number of associations, many of them potentially insignificant.

Another problem with simple association rule mining is that often a large number of obvious and uninteresting rules are generated along with the useful ones. In practice, this is considered one of the major drawbacks of association rule mining: if the support is set high, fewer rules are generated but most of them may be obvious and hence useless (e.g., "Cereal => Milk"). On the other hand if the support is set low, too many rules are generated and the domain experts have to evaluate the generated rules and identify the ones that are useful. The market basket analysis node computes a measure called support lift to identify more interesting rules based on the deviation of a rule's support from its expected support derived from the support of parents of the items in the rule. From an objective perspective, the higher the deviation the more "surprise" the rule holds, consequently the rule is likely to be more interesting. To give a different perspective, item taxonomy represents a limited form of domain knowledge which is used by the Market Basket node to generate more interesting rules.

The market basket analysis is not limited to retail marketing domain. The analysis frame work can be abstracted to other areas such as word co-occurrence relationships in text documents.

Using the Market Basket Node

The Market Basket node requires an input data source that contains one or more ID variables and a target variable. The ID variables are used to group the target into baskets. The Target variable is a single nominal variable. Finally, the input data source must have a role of transaction.

You can use a single data set to specify the hierarchy of items by setting the Dimension property. See the example for details on how to do so.

The following are not supported in this release:

- Multiple parents are not supported in the item hierarchy. If multiple parents are specified for a child (at any level), all except the last one are ignored.
- Rugged hierarchy is not supported. That is, at any level if a parent exits for an item, it must immediately appear in the next level.

Market Basket Node Properties

Market Basket Node General Properties

The following general properties are associated with the Market Basket Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Market Basket node added to a diagram will have a Node ID of MRKBSKT. The second Market Basket node added to a diagram will have a Node ID of MRKBSKT2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Market Basket window. The Imported Data — Market Basket window contains a

list of the ports that provide data sources to the Market Basket node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data - Market Basket window. The Exported Data - Market Basket window contains a list of the output data ports that the Market Basket node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Market Basket Node Train Properties

- **Variables** — Select the ... button to open the Variables — Market Basket table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use and Report variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Normalize** — Use to specify whether to normalize class variables or not. This setting will result in target variable normalization where character string values are left justified and upper cased. The default is no normalization.

Market Basket Node Train Properties: Constraints

- **Maximum Items** — Use to specify the maximum number of items in a set to be considered in an association. Specify a number between 1 and 100. The default is 2. For example, the default of 2 indicates that up to 2-way associations are performed.
- **Minimum Confidence Level** — Use to specify the minimum confidence level that is required to generate a rule. Specify a real number between 0 and 100. The default is 50.0%.
- **Minimum Lift** — Use to specify the minimum lift for the rules.
- **Minimum Support Lift** — Use to specify the minimum support lift for the rules. This option is valid when a hierarchy dimension data set property is specified.
- **Support Type** — Use to specify the type of support that is generated to claim that items are associated.
  - **Count** — Use to express minimum transaction frequency as count.
  - **Percent** — Use to express minimum transaction frequency as percentage. This is the default.
- **Support Count** — Use an integer value to specify the minimum transaction frequency to support. The frequency is expressed as count. This option is available when Support Type property is set to Count.
- **Support Percentage** — Use to specify the minimum transaction frequency to support. The frequency is expressed as percentage. This option is available when Support Type property is set to Percent. The default is 2.0%.

Market Basket Node Train Properties: Hierarchy

- **Dimension Data Set** — Use to specify the data source that defines the hierarchy of items. Select the ... button to open the Select a SAS Table window.

  The data set must contain the following variables:

  - a nominal variable taking the child role
  - a nominal variable taking the parent role
  - a numeric variable indicating the level of the child in the hierarchy

  Note: If no hierarchy is specified, the Market Basket node will perform simple association analysis with the input data without the hierarchy. Each distinct item that appears in the input (transactional) data set must have a parent in the lowest level of the hierarchy.

- **Mapping** — Use to open an editor that enables you to specify the parent and children variables that define the hierarchy. Select the ... button to open the Mapping window. This property is grayed out if the Dimension Data Set property is not specified.

Market Basket Node Train Properties: Basket Size Options

- **Minimum Size** — Use to specify the minimum size of the basket to be considered valid in the training data set. A basket size is a grouping of all target items by the customer id. Baskets that have smaller sizes that this setting are rejected. The default is 1.

- **Maximum Size** — Use to specify the maximum size of the basket to be considered valid in the training data set. The default is 1000.

Market Basket Node Train Properties: Rules

The following rules properties are associated with the Market Basket Node:

- **Maximum Number of Rules** — Use to specify the maximum number of generated rules.
- **Number to Keep** — Use to specify the maximum number of rules to keep for the results. The default is to output all the rules. If the number of generated rules is smaller than the specified number, this setting will have no effect.
- **Sort Criterion** — Use to specify the statistic used to sort the generated rules. By default, the rules are sorted by Support.

  Specify any of the following values:

  - **Confidence**
  - **Lift**
  - **Size**
  - **Size of Left Hand Side**
  - **Size of Right Hand Side**
  - **Support**
  - **Support Lift**

Market Basket Node Score Properties

The following score properties are associated with the Market Basket Node:

- **Rules**
  - **Export Rule by ID** — determine whether the rule by ID data set is exported as a training data set. When the data set is exported it consists of rule variables that identifies the rules that are associated with each customer ID.
  - **Transpose Selection** — specifies how to select the rules that are transposed. When you select **User Defined**, the **Rules** property permits you to determine the rules that are transposed. When you select **Automatic**, the first N rules are transposed, where N is determined by the **Number to Transpose** property. Additionally, all previous user defined selections are reset when you choose **Automatic**.

    When the node runs for the first time, it will always act as if **Automatic** is selected. This ensures that there are rules to use in the score code. If you want to use **User Defined** rule selection, you must run the node once, use the **Rules** editor to select the rules to transpose, and then run the node a second time.
  - **Number to Transpose** — specifies the maximum number of rules that will be transposed and used to create the training data set (that is, the rule by ID data set).
  - **Rules** — Click the ... button to launch an editor that enables you to select rules interactively. The **Rules** property is available only when the **Transpose Selection** is set to **User Defined**.
  - **Recommendation** — specifies that the score is done only for recommendations. You should set this property to **No** if you are trying to determine which rules are met for a given ID.

Market Basket Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Market Basket Node Results

You can open the Results window of the Market Basket Node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

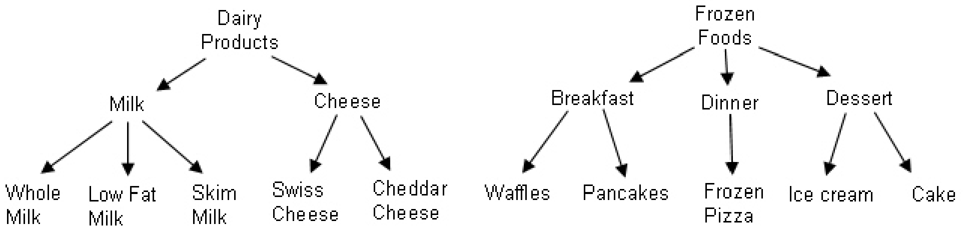
- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Market Basket Node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Market Basket Node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Market Basket Node . The table includes columns to see the variable name, the variable role, the variable level, and the model used. (Name, Role, Level, and Use).
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Market Basket Node run.
  - **Output** — the SAS output of the Market Basket Node run.
  - **Flow Code** — the SAS code used to produce the output that the Market Basket Node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Market Basket Node does not generate PMML code.
- **Rules**

- **Item Statistics** — opens an Item Statistics window containing a table with columns for the Item Name, Transaction Count, and Support(%).
- **Rules Table** — opens a Rules Table window containing a table with information about any rules that have been specified.
- **Statistics Plot** — opens a Statistics Plot window containing a graph of Relations with Confidence(%) on the x-axis and Support(%) on the y-axis.
- **Statistics Line Plot** — opens a Statistics Line Plot window with a graph of Rule ID plotted against Lift.

- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard that you can use to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Market Basket Node Example

The following example taken from a grocery store context will serve to clarify the concepts of generalized association rule mining. For simplicity, assume that the lowest level in the taxonomy represents the actual items in the customer baskets. The item taxonomy for this example is shown below.



The transaction data is shown in the table below. Note that in reality the market baskets contain SKUs or some other unique identification code for specific products on sale indicating things such as the brand, size, type, and so on (e.g., "Kraft low fat Swiss cheese, sliced, 8Oz").

Customer	Item
Anne	Low Fat Milk
Anne	Cheddar Cheese
Anne	Cake
Anne	Frozen Pizza
Anne	Ice Cream
Anne	Pancakes

Customer	Item
Bob	Low Fat Milk
Bob	Swiss Cheese
Bob	Frozen Pizza
Bob	Ice Cream
Chris	Skim Milk
Chris	Swiss Cheese
Chris	Ice Cream
Chris	Cake

The taxonomy data consist of a flattened data set. A row represents the child-parent relationship and the specific level in the taxonomy. A child cannot have multiple parents. The table below shows the corresponding to the taxonomy shown in the first figure.

Product	Parent	Level
Whole Milk	Milk	1
Low Fat Milk	Milk	1
Skim Milk	Milk	1
Swiss Cheese	Cheese	1
Cheddar Cheese	Cheese	1
Waffles	Breakfast	1
Pancakes	Breakfast	1
Frozen Pizza	Dinner	1
Ice Cream	Dessert	1
Cake	Dessert	1
Milk	Dairy Products	2
Cheese	Dairy Products	2
Breakfast	Frozen Foods	2
Dinner	Frozen Foods	2

Product	**Parent**	**Level**
Dessert	Frozen Foods	2

1. Create a new project and define the start code.

   - From the Enterprise Miner window, select **File** ⇨ **New Project**. The Create New Project window opens.
   - In the **Name** box, type a name for the project, such as **`Market Basket Example`**. Finish creating the new project.
   - After you have finished creating your new project, click the top of the tree of the project in the tree view. Once you have selected Market Basket Example, or the name of your project, click the ... button to the right of the Project Start Code property in the properties panel. Enter the following statement in the editor in the Project Start Code window:

     ```
     libname mba '<path-to-your-example-library>';
     ```

     Note: Note: You must replace **`<path-to-your-example-library>`** with the path specification that points to an existing folder on your client machine.
   - Click **Run Now**. Click **OK** to close the Project Start Code window.
   - Select **View** ⇨ **Explorer** and verify that Mba appears as a SAS library on the Explorer window. Note: You may need to refresh the screen by clicking the Show Project Data check box.

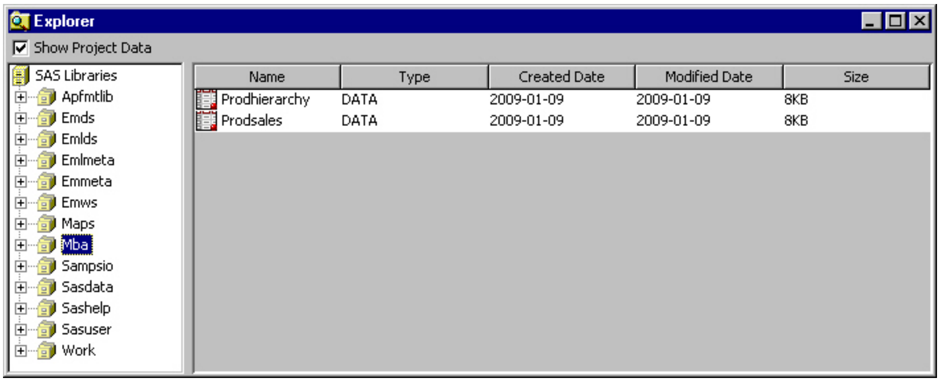
2. From the Enterprise Miner window, select **View** ⇨ **Program Editor**. Copy and run the SAS code below:

```
data mba.prodhierarchy;
  length Product $20 ParentProd $20;
   Product='Whole Milk';     ParentProd='Milk';           level=1; output;
   Product='Low Fat Milk';   ParentProd='Milk';           level=1; output;
   Product='Skim Milk';      ParentProd='Milk';           level=1; output;
   Product='Swiss Cheese';   ParentProd='Cheese';         level=1; output;
   Product='Cheddar Cheese'; ParentProd='Cheese';         level=1; output;
   Product='Waffles';        ParentProd='Breakfast';      level=1; output;
   Product='Pancakes';       ParentProd='Breakfast';      level=1; output;
   Product='Frozen Pizza';   ParentProd='Dinner';         level=1; output;
   Product='Ice Cream';      ParentProd='Dessert';        level=1; output;
   Product='Cake';           ParentProd='Dessert';        level=1; output;
   Product='Milk';           ParentProd='Dairy Products'; level=2; output;
   Product='Cheese';         ParentProd='Dairy Products'; level=2; output;
   Product='Breakfast';      ParentProd='Frozen Foods';   level=2; output;
   Product='Dinner';         ParentProd='Frozen Foods';   level=2; output;
   Product='Dessert';        ParentProd='Frozen Foods';   level=2; output;
 run;

 data mba.prodsales;
   length Customer $ 10 Item $ 20 ;
   retain Customer;
   customer='Anne';
    Item='Low Fat Milk';   output;
    Item='Cheddar Cheese'; output;
    Item='Cake';           output;
    Item='Frozen Pizza';   output;
```

```
     Item='Ice Cream';      output;
     Item='Pancakes';       output;
    customer='Bob';
     Item='Low Fat Milk';   output;
     Item='Swiss Cheese';   output;
     Item='Frozen Pizza';   output;
     Item='Ice Cream';      output;
    customer='Chris';
     Item='Skim Milk';      output;
     Item='Swiss Cheese';   output;
     Item='Ice Cream';      output;
     Item='Cake';           output;
  run;
```

3. Select **View** ⇨ **Explorer** to open the Explorer window.

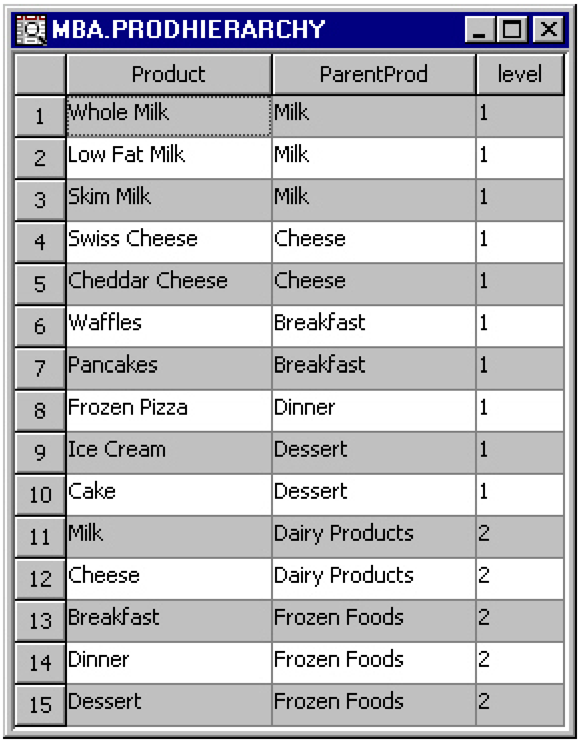
   - On the Explorer window, select the Mba library and verify that Prodhierarchy and Prodsales data sets are generated.



SAS Libraries
- Apfmtlib
- Emds
- Emlds
- Emlmeta
- Emmeta
- Emws
- Maps
- Mba
- Sampsio
- Sasdata
- Sashelp
- Sasuser
- Work

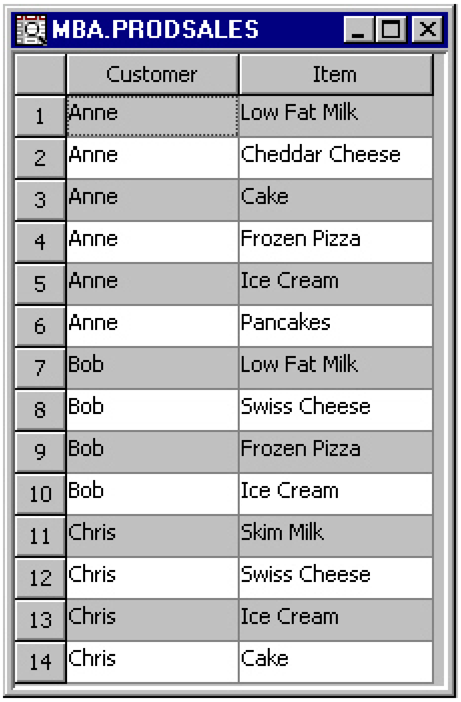
Name	Type	Created Date	Modified Date	Size
Prodhierarchy	DATA	2009-01-09	2009-01-09	8KB
Prodsales	DATA	2009-01-09	2009-01-09	8KB

   - Double-click the Prodhierarchy data set to open the table:



	Product	ParentProd	level
1	Whole Milk	Milk	1
2	Low Fat Milk	Milk	1
3	Skim Milk	Milk	1
4	Swiss Cheese	Cheese	1
5	Cheddar Cheese	Cheese	1
6	Waffles	Breakfast	1
7	Pancakes	Breakfast	1
8	Frozen Pizza	Dinner	1
9	Ice Cream	Dessert	1
10	Cake	Dessert	1
11	Milk	Dairy Products	2
12	Cheese	Dairy Products	2
13	Breakfast	Frozen Foods	2
14	Dinner	Frozen Foods	2
15	Dessert	Frozen Foods	2

   - Double-click the Prodsales data set to open the table:



	Customer	Item
1	Anne	Low Fat Milk
2	Anne	Cheddar Cheese
3	Anne	Cake
4	Anne	Frozen Pizza
5	Anne	Ice Cream
6	Anne	Pancakes
7	Bob	Low Fat Milk
8	Bob	Swiss Cheese
9	Bob	Frozen Pizza
10	Bob	Ice Cream
11	Chris	Skim Milk
12	Chris	Swiss Cheese
13	Chris	Ice Cream
14	Chris	Cake

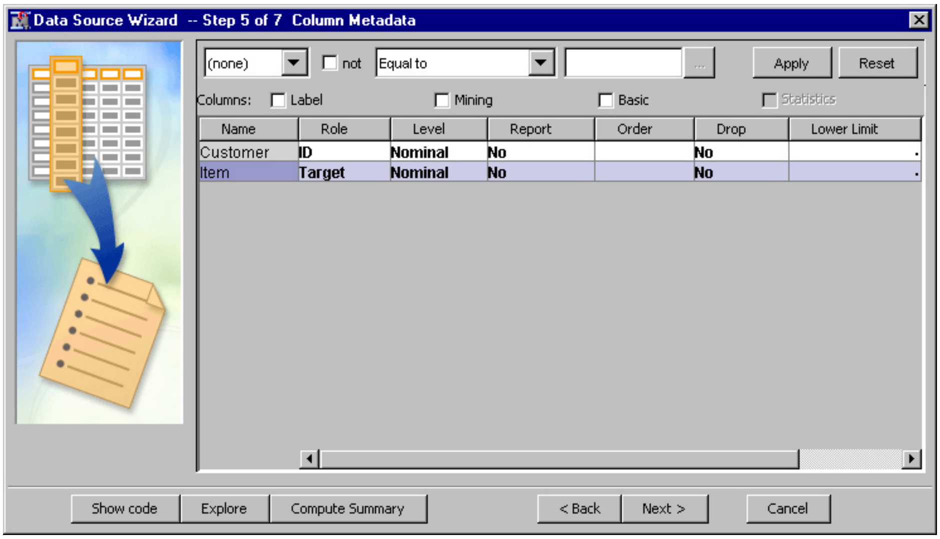
4. Right-click the Data Sources folder on the Project panel and select Create Data Source. The Data Source Wizard Step 1 window opens.

- Select **SAS Table** as the source and click **Next**. The Data Source Wizard Step 2 window opens.

- Click **Browse** to open the Select a SAS Table window. Select the Mba library and Prodsales data set then click **OK**. Mba.Prodsales appear on the Table field.

- Click **Next**. The Data Source Wizard Step 3 Table Information window opens. Click **Next**. The Data Source Wizard Step 4 Metadata Advisor Options window opens. Click **Next**. The Data Source Wizard Step 5 window opens.

- Change the role of Customer variable to ID. Change the role of Item variable to Target.

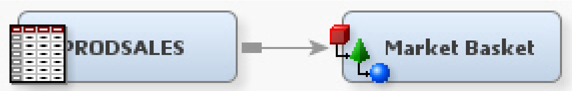

Data Source Wizard -- Step 5 of 7 Column Metadata
(none)
□ not
Equal to
Apply
Reset
Columns: □ Label □ Mining □ Basic □ Statistics
Show code
Explore
Compute Summary
< Back
Next >
Cancel

Name	Role	Level	Report	Order	Drop	Lower Limit
Customer	ID	Nominal	No		No	.
Item	Target	Nominal	No		No	.

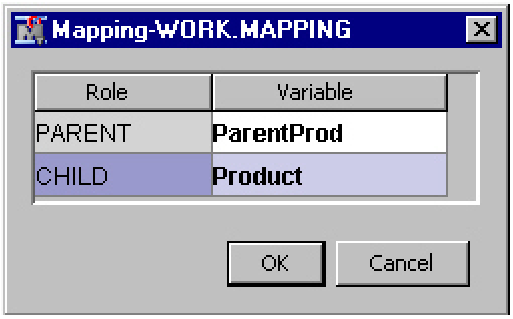
- Click **Next**. The Data Source Wizard Decision Configuration window opens. Click **Next**.

- The Data Source Wizard Data Source Attributes window opens. Set the Role of the Prodsales data set to Transaction. Click **Next**.
- The Summary window opens. Click **Finish**. The Prodsales data set appears under the Data Sources folder on the Project Panel.

5. Right-click the Diagrams folder on the Project panel and select Create Diagram. Type a diagram name on the Create a Diagram window and click **OK**.
6. Drag the PRODSALES data source onto the diagram workspace. Drag a Market Basket node from the Explore tab of the node toolbar and connect this node to the PRODSALES data source node.

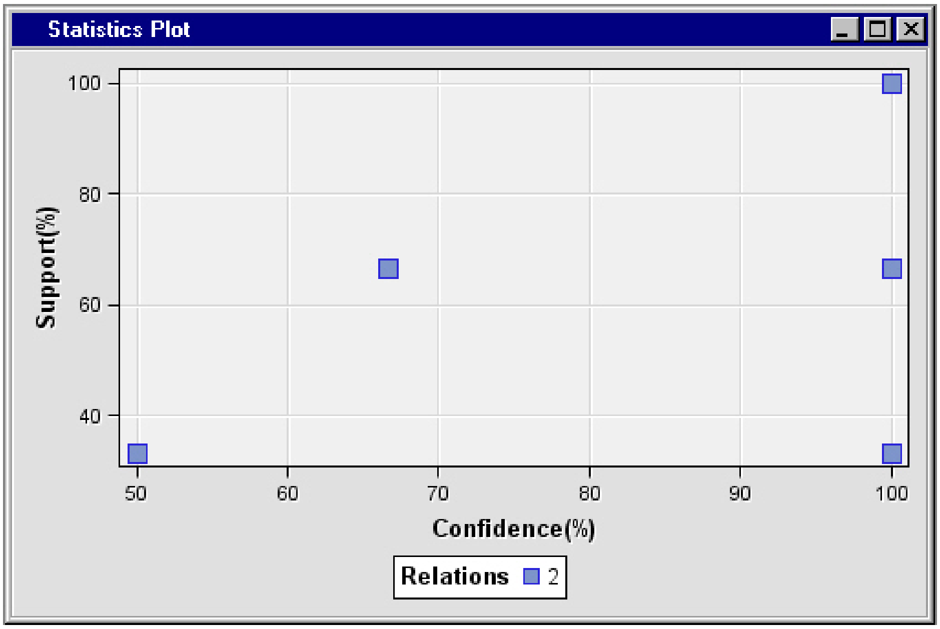


7. Select the Market Basket node on the diagram and click the ... button to the right of the Dimension Data Set property to open the Select a SAS Table window. Select the Mba.Prodhierarchy data set and click OK. The Mba.Prodhierarchy data source appears on the Dimension Data Set property panel.
8. Select the Market Basket node on the diagram and click the ... button to the right of the Mapping property to open the Mapping window. Set the ParentProd variable with a PARENT role and the Product variable with a CHILD role. Click **OK**.

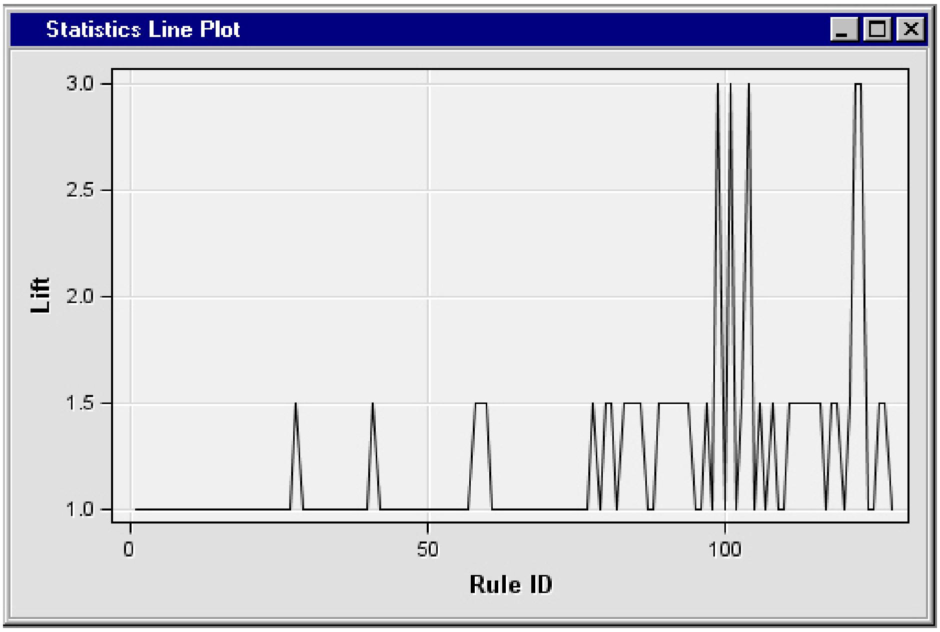

Mapping-WORK.MAPPING

Role	Variable
PARENT	ParentProd
CHILD	Product

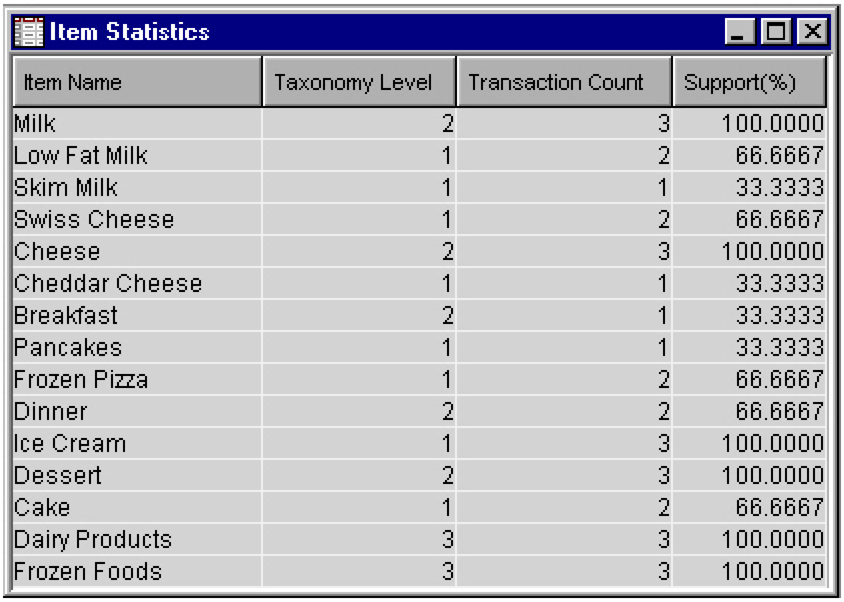
9. Right-click the Market Basket node on the process flow diagram and select Run. Click Results.
   - Examine the Statistics plot.



- Examine the Statistics Line Plot.



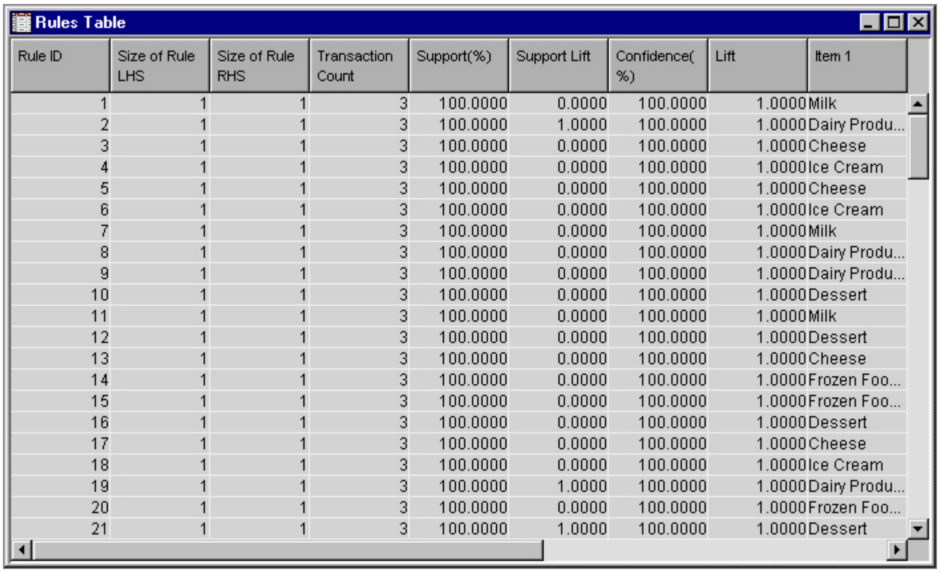
- Examine the Item Statistics Table.



Item Statistics

Item Name	Taxonomy Level	Transaction Count	Support(%)
Milk	2	3	100.0000
Low Fat Milk	1	2	66.6667
Skim Milk	1	1	33.3333
Swiss Cheese	1	2	66.6667
Cheese	2	3	100.0000
Cheddar Cheese	1	1	33.3333
Breakfast	2	1	33.3333
Pancakes	1	1	33.3333
Frozen Pizza	1	2	66.6667
Dinner	2	2	66.6667
Ice Cream	1	3	100.0000
Dessert	2	3	100.0000
Cake	1	2	66.6667
Dairy Products	3	3	100.0000
Frozen Foods	3	3	100.0000

- From the Results Window, select **View** ⇨ **Rules** ⇨ **Rules Table**.



Rules Table

Rule ID	Size of Rule LHS	Size of Rule RHS	Transaction Count	Support(%)	Support Lift	Confidence(%)	Lift	Item 1
1	1	1	3	100.0000	0.0000	100.0000	1.0000	Milk
2	1	1	3	100.0000	1.0000	100.0000	1.0000	Dairy Produ...
3	1	1	3	100.0000	0.0000	100.0000	1.0000	Cheese
4	1	1	3	100.0000	0.0000	100.0000	1.0000	Ice Cream
5	1	1	3	100.0000	0.0000	100.0000	1.0000	Cheese
6	1	1	3	100.0000	0.0000	100.0000	1.0000	Ice Cream
7	1	1	3	100.0000	0.0000	100.0000	1.0000	Milk
8	1	1	3	100.0000	0.0000	100.0000	1.0000	Dairy Produ...
9	1	1	3	100.0000	0.0000	100.0000	1.0000	Dairy Produ...
10	1	1	3	100.0000	0.0000	100.0000	1.0000	Dessert
11	1	1	3	100.0000	0.0000	100.0000	1.0000	Milk
12	1	1	3	100.0000	0.0000	100.0000	1.0000	Dessert
13	1	1	3	100.0000	0.0000	100.0000	1.0000	Cheese
14	1	1	3	100.0000	0.0000	100.0000	1.0000	Frozen Foo...
15	1	1	3	100.0000	0.0000	100.0000	1.0000	Frozen Foo...
16	1	1	3	100.0000	0.0000	100.0000	1.0000	Dessert
17	1	1	3	100.0000	0.0000	100.0000	1.0000	Cheese
18	1	1	3	100.0000	0.0000	100.0000	1.0000	Ice Cream
19	1	1	3	100.0000	1.0000	100.0000	1.0000	Dairy Produ...
20	1	1	3	100.0000	0.0000	100.0000	1.0000	Frozen Foo...
21	1	1	3	100.0000	1.0000	100.0000	1.0000	Dessert

References

Berry, Michael JA and Linoff, Gordon 1997. *Data Mining Techniques for Marketing, Sales, and Customer Support*. New York, NY: John Wiley and Sons.

Chapter 35
MultiPlot Node

MultiPlot Node

Overview of the MultiPlot Node

The MultiPlot node is a tool that you can use to visualize your data from a wide range of perspectives. With MultiPlot you can graphically explore large volumes of data, observe data distributions, and to examine relationships among the variables. The node is used primarily in the Explore phase of the SEMMA (Sample, Explore, Model, Modify, Assess) methodology. MultiPlot reveals extreme values in the data and helps you discover patterns and trends.

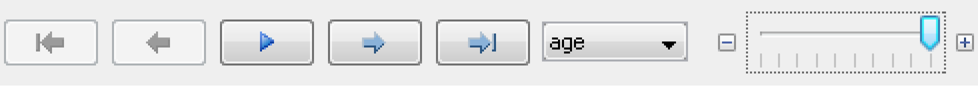
MultiPlot node configuration is simple and straightforward. In addition, MultiPlot generates code that you can use to create graphs in a batch environment.

The MultiPlot node creates the following types of charts:

- Bar Charts:
  - Histogram of each input and target.
  - Bar chart of each input versus each class target.
  - Bar chart of each input grouped by each interval target.
- Scatter Plots:
  - Plot of each interval input versus the target.
  - Plot of each class input versus the target.

You can annotate the interval input by interval target scatter plots with a regression line and 90% confidence intervals for the mean.

The Results window contains several tool icons for managing the graphs, shown below. You can select a graph from a list of all graphs, manually navigate through all the graphs, or automatically scroll through all the graphs. The drop down menu, displaying **age** in the image below, enables you to select the variable that you want to graph.



From left to right, the tool icons perform the following actions:

- ⇤ — returns you to the first graph in the list.
- ⇦ — returns you to the previous graph in the list.
- ▶ — starts an automatic slide show of all the plots. The speed of the slide show is determined by the slider bar to the right of variable selection drop down menu. When the slider is to the far right, as shown above, the graph will change every second. When the slide is the far left, the graph will change every 10 seconds.
- ⇨ — moves you to the next graph in the list.
- ⇥ — moves you to the final graph in the list.

The MultiPlot node must be connected to a predecessor node that exports one or more data sets, such as the Input Data, Data Partition, or Regression nodes.

MultiPlot Node Properties

MultiPlot Node General Properties

The following are the general node properties that MultiPlot uses:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first MultiPlot node that is added to a diagram will have a Node ID of Plot. The second MultiPlot node added to a diagram will have a Node ID of Plot2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — MultiPlot window. The Imported Data — MultiPlot window contains a list of the ports that provide data sources to the MultiPlot node. Select the ... button to open the Imported Data — MultiPlot window, which contains a table displaying the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — MultiPlot window. The Exported Data — MultiPlot window contains a list of the

output data ports that the MultiPlot node creates data for when it runs. Select the ... button to open the Exported Data — MultiPlot window, which contains a table displaying the exported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

MultiPlot Node Train Properties

The following train properties are associated with the MultiPlot node:

- **Variables** — Use the Variables property to specify the properties for each variable in your data source. Select the ... button to view a variables table.
- **Type of Charts** — Use the Type of Charts property to specify the type of charts you want to generate with your data. If at least one class target variable is defined, bar charts will be generated for each variable with the class target as a subgroup. If there is no class targets defined, the bar charts are univariate charts. If an interval target is defined, additional bar charts will be generated according to the value specified in the Interval Target Charts option.
  - **Bar Charts** (default setting)
  - **Scatter Plots**
  - **Both**

MultiPlot Node Train Properties: Bar Chart Options

- **Graph Orientation** — Use the Orientation property to specify the visual orientation that you want on the graphs that are generated. You can choose between **Vertical** (default setting) or **Horizontal**.
- **Include Missing Values** — When set to **No**, the Include Missing property does not use missing values in plots. The default setting for the Include Missing Values property is **Yes**.
- **Interval Target Charts** — Use the Interval Target Chart property to indicate the type of chart that you want to use with interval targets.

  The choices are as follows:

  - **Mean** — (default setting) interval means are plotted.
  - **Sum** — interval sums are plotted.
  - **None** — no charts are produced for interval targets
- **Show Values** — When the Show Values property is set to **No**, data values are suppressed from displaying on the generated plots. The default setting for the Show Values property is **Yes**.
- **Statistic** — Use the Statistic property to specify the type of Y-axis statistic that you want to appear on plots.

  - **FREQ** — the Y-axis displays the frequency variable.
  - **PERCENT** — The Y-axis displays %Y.
- **Numeric Threshold** — Use the Numeric Threshold property to specify the bin threshold that you want to use. The bin threshold determines whether binning takes place when you create a plot. If the number of unique levels in your data set exceeds the threshold value, binning occurs and midpoints are displayed on bar charts instead of actual levels. To prevent binning, set the threshold higher than the number of unique levels in the data set. Permissible values are nonnegative integers. The default value for the Numeric Threshold property is 20.

MultiPlot Node Train Properties: Scatter Options

- **Confidence Interval** — When set to **No**, the Confidence Interval property suppresses the confidence intervals that you used in your analysis from appearing on your plots. The default setting for the Confidence Interval Property is **Yes**.
- **Regression Equation** — When set to **Yes**, the Display Regression Equation property displays the regression equations that you used on your scatter plots. The default setting for the Regression Equation property is **No**.
- **Regression Type** — Use the Regression Type setting to choose the type of regression that you want to perform on interval targets.

  The choices are as follows:

  - **Linear** — (default setting) linear regression
  - **Quadratic** — quadratic regression
  - **Cubic** — cubic regression
  - **None** — no regression is performed on interval targets

MultiPlot Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

MultiPlot Node Bar Chart Notes

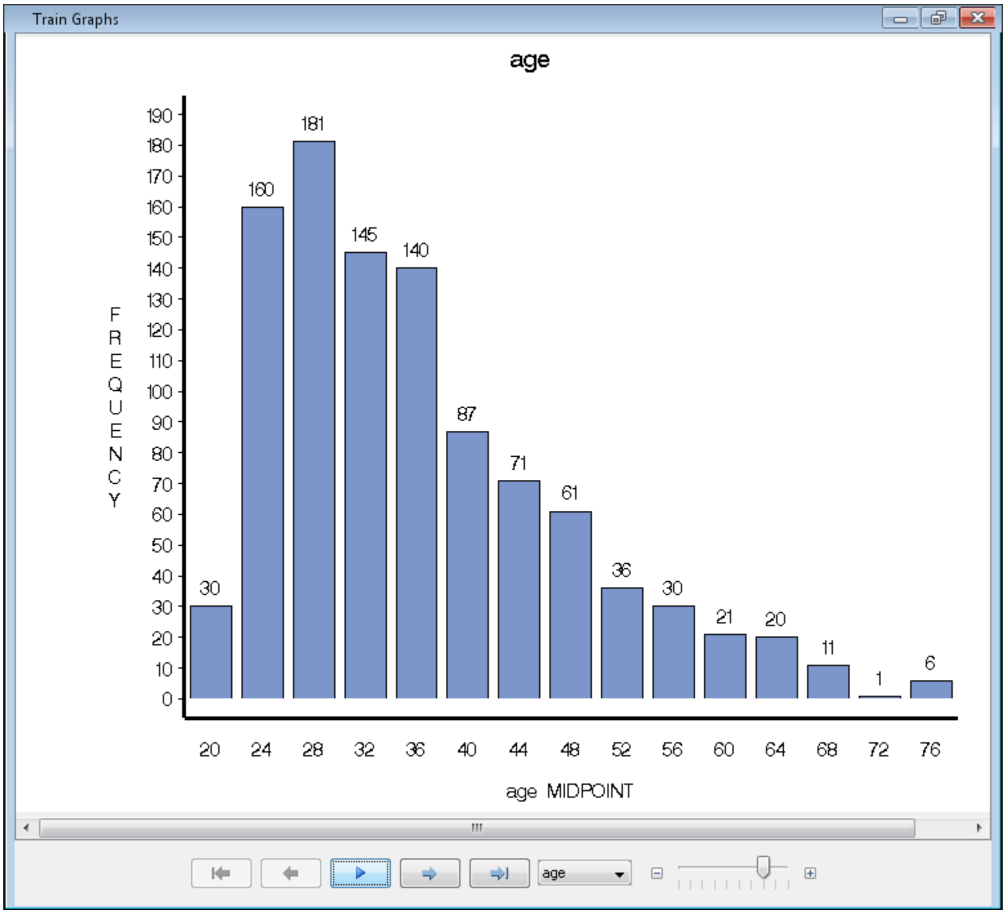
Introduction

The Numeric Threshold property is used to determine whether binning takes place when creating a plot. This option does not configure the number of bins used. If the number of unique levels in your data set is greater than the Numeric Threshold value, binning will occur and midpoints will be displayed on your bar charts instead of actual levels. If you do not want binning to occur, the Numeric Threshold must be greater than the number of unique levels in the data set. The default value is 20.

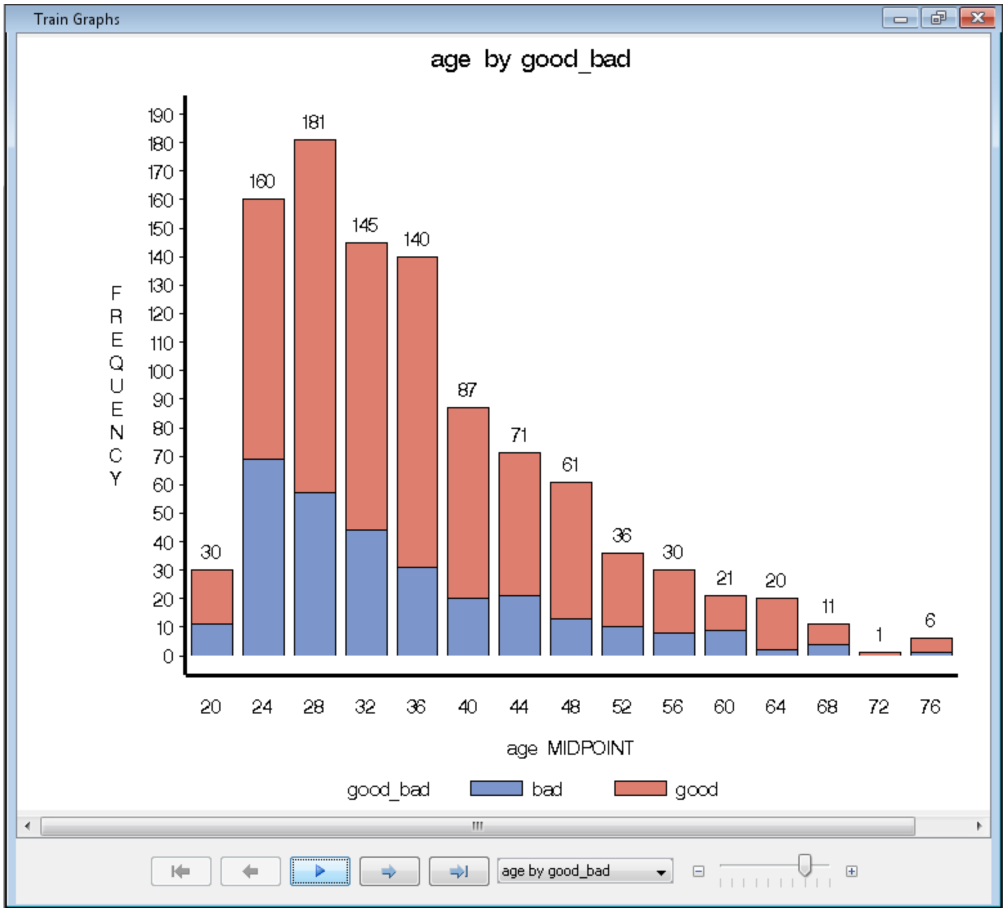
For example, the data set SAMPSIO.DMAGECR contains an interval variable, DURATION, which has 28 unique levels. If you run this data set in MultiPlot with the default settings of Numeric Threshold=20, the plot for DURATION will display the binned midpoints across the x-axis. However, if you increase the Numeric variable threshold to any number greater than 28 and re-run the node, the plot will display all 28 unique levels across the x-axis.

"Univariate Histogram" on page 511 display values for a single variable. "Bivariate Histogram" on page 512 plot the specified variable by target value behavior. Some examples of univariate and bivariate histograms can be found below.

Univariate Histogram



Bivariate Histogram



Note: Bivariate histograms require a target variable. If no target variable is declared in the input data, only univariate histograms can be produced.

At the bottom of the Graphs window the following buttons are displayed:

- **First** — displays the first graph.
- **Previous** — displays the previous graph.
- **Next** — displays the next graph.
- **Last** — displays the last graph.
- Click the drop-down list to display a specific graph.

Graphics Storage Locations

The graphs that are produced by the MultiPlot node are stored in a directory in your project folder. For example, suppose that you have a project named Phoenix, and the diagram ID is EMWS1. The MultiPlot graphs are stored in the following directory: **`../Projects/Phoenix/Workspaces/Emws1/Plot/Graph`**.

Graphs are stored as GIF images. They are saved under the same names that are displayed in the Graphs window when you view the MultiPlot results.

When you create a report from the MultiPlot node, graphs that are produced by the MultiPlot node are stored in the report SPK file.

Chapter 36
Path Analysis Node

Path Analysis Node

Overview of the Path Analysis Node

In the SAS SEMMA data mining methodology, the Path Analysis node belongs to the Explore group. The core functionality of the Path Analysis node is to analyze Web log data to determine the most frequent paths taken by Web site visitors. The Path Analysis node can be used as a research and marketing tool to analyze preprocessed Web log data. The Path Analysis node can also analyze collected sequence data to reveal frequent consecutive sub-sequences.

Path Analysis Node Features

Path Scoring

The Path Analysis node supports path scoring. Path scoring maps visitor IDs to path sequence rules. Path Analysis node scoring can be classified into two types.

- Basic scoring matches entire sequence rules with visitor sessions. If the visitor session meets the requirements of the sequence rule, the user ID and sequence rule variables <VISITOR_ID, RULE_ID> are included in the Path Analysis node output.

This is the default scoring behavior, and it can be used to identify visitors who follow the most "popular" paths.

- "Left-hand side only" scoring identifies visitors who follow the most "popular" paths except for the last page or click.

A separate <Visitor_ID, Rule_ID> observation is output for each rule that the visitor sequence satisfies. If no rules are satisfied during a visitor sequence, a single observation is output with a missing value entry in the Rule_ID field.

The output for left-hand side only scoring contains two additional variables that are not included in the output for basic scoring. The new output variables are RULE_RHS and TARGET. RULE_RHS contains the right-hand side of the rule, and TARGET displays the location that the visitor clicked on. If the visitor leaves or drops off the Web site, the TARGET value will be missing.

Left-hand side only scoring generates output only if the following criteria are met:

- the visitor's sequence satisfies the left-hand side of the sequence rule
- either the visitor's session has a click other than the right-hand side of the path rule, or there are no more other requests or clicks.

The Path Analysis node needs a data set of sequence rules in order to score visitors. The rules input can come from a number of sources.

- use the top n rules generated during the current Path Analysis node run where the Training Mode property is set to Train (no input data set is specified).
- use the complete path rules data set (or a subset of the path rules data set) that was generated during a previous Path Analysis node run where the Training Mode property is set to Train.
- use a manually created sequence pattern.

Recount Paths

You can submit a set of paths that were uncovered by previous Path Analysis node runs in training mode, and use them to generate results metrics for the paths, such as confidence and support, in the current Web site log data set. The recount paths feature can be used to indicate how user patterns change over time, a key analytical perspective for tracking frequent paths. Sequence rules for recount paths come from the same sources as sequence rules for scoring.

- output from a previous Path Analysis node run where the Training Mode property is set to Train
- manually created sequence patterns.

If the recount path input data set is output from a previous Path Analysis node run, both previous and updated performance metrics are included in the node output. For manually created recount path sequences, only current metrics are included in the output.

Transition Probabilities

Transition probabilities are also known as entry-exit probabilities. Transition probabilities attempt to quantify the probability that a visitor will move (or transition) from one given location to another given location. To calculate transition probabilities in the Path Analysis node, users submit a list of locations (or "nodes") that they are interested in.

For each target node, two parameters are calculated and sent to the Path Analysis output:

- the top k nodes from which users visited the target node

- the top k nodes users requested after visiting the target node.

The input parameters are submitted to the Path Analysis node in two forms:

- a manually generated list of target values or most requested pages
- the top n pages accessed during a Path Analysis node run where the Training Mode property is set to Train (no input data set is specified).

The transition probabilities output consists of top k entry pages, with the counts and percentages as a fraction of all entry page counts, and the top k exit pages with the counts and percentages as a fraction of all exit page counts.

The example table below shows the output for two target pages, LL and MM with top k = 2:

Target	Count	From1	From Count 1	From Pct 2	From2	From Count 2	From Pct 2	Tol	To Count 1	To Pct 1	To 2	To Count 2	To Pct 2
LL	100	AA	40	50.0	BB	20	25.0	PP	45	50.0	QQ	30	33.3
MM	200	CC	150	75.0	DD	50	25.0	RR	25	100.0	.	.	.

Note: The Target count and the sum of From / To counts may not be equal. The difference represents the visitors who "dropped off" to / from the target, or visitors who transitioned to / from other pages that are not displayed in the chart.

Funnel Counts

Funnel counts show the drop-off in the number of visitors along a particular path of interest. It can be useful to see how visitor attrition occurs along a path, indicating points of interest such as where the biggest drop-off points are. For example, on a given sequence path of **A** ⇨ **B** ⇨ **C** ⇨ **D**, the funnel count output may look like <1002, 899, 300, 40>, which means that 1002 visitors came to A, 899 visitors went on to B, 300 of which continued on to C, and so on. The input parameters for funnel counts can be submitted to the node in either of the two forms.

- a manually generated sequence pattern.
- the path rules output from a previous Path Analysis node run where the Training Mode property is set to Train.

Restrict Patterns

Instances may arise where Path Analysis node users are not interested in all frequent paths. Instead, they may be interested only in certain specific paths. For example, an analyst may only be interested in visitors who viewed the "ReturnPolicy.htm" page on an e-commerce Web site, then subsequently abandoned their shopping cart.

The Path Analysis node offers restrict patterns that are based on a parameter set of four values: <BEGIN, END, MINLENGTH, MAXLENGTH>. Only one of the BEGIN or END values can be left unspecified. One or both of the MINLENGTH and MAXLENGTH values can be omitted. If the EXACT property is specified, the beginning and / or the ending page in the input restrict pattern must exactly match the first and / or last page requests in the user sequence.

Use the Restricted Paths property to configure whether you want to use the Restrict Patterns feature of the Path Analysis node. The restrict patterns data set must be manually constructed and saved to a location where Enterprise Miner can access it.

Path Analysis Node Data Set Requirements

Path Analysis Transaction Data Sets

Your data set must contain the following:

- one Target variable.
- one or more ID variables.
- one Sequence variable.

Optionally, you can define a single Referrer variable. If a visitor is on a target page, the referrer page is the page immediately preceding the current page in a sequence. The data set that you analyze with the Path Analysis node must have a data set role of **Transaction**. When possible, the input data set should be sorted by the Target, ID, Sequence, and Referrer variables.

Path Analysis Rule Data Sets

The Path Analysis node functions Recount Paths, Score, and Funnel Count require input data sets that contain sequence patterns. These input data sets must come from one of three sources.

- The output path rules data set produced by a prior run of the Path Analysis node configured with the Train Mode property set to Train.
- A manually created SAS data set which exactly duplicates the content and structure of a Path Analysis node output sequence pattern data set. The manually created data set must use the same variable names that would have been used in an output Path Analysis data set.
- A manually created SAS data set which contains only the target items and no other variables (such as RULEID, CONF, and so on). The order of target items in the rule data set must follow the order of the items in the rule. For example, the created data set for the rule **Homepage.htm** ⇨ **Books.htm** ⇨ **Classics.html** would require the following structure:

REQ1	REQ2	REQ3
Homepage.htm	Books.htm	Classics.htm

Path Analysis Restrict Pattern Data Sets

The Path Analysis node Restrict Pattern function requires an input data set that specifies specific sequence patterns to be analyzed.

The restrict pattern input data set must have the following format:

- A data set with the variable names, "BGN_ITEM", "END_ITEM", "MIN_LEN", and "MAX_LEN". When these variable names are used, the variable data columns can appear in any order.
- The Path Analysis node will also accept a restrict pattern data set that does not use the exact variable names specified above, but the variable columns must be in a specific order. Any variable names can be used, but the restrict pattern data set must

be formatted in the following order: <MINLENGTH, MAXLENGTH, BEGIN, END>. Any additional variables will be ignored.

Path Analysis Path Completion

The algorithm underlying the Enterprise Miner Path Analysis node uses heuristics to infer user requests which are absent from a Web site's server log due to client-side browser page caching. Path completion is necessary to generate results that accurately conform to the analyzed Web site's structure. The following example illustrates the effect of path completion on other dependent analytical results.

Consider the following sequence from a Web server log:

VISITOR ID	Requested File	Referrer
1001	Homepage.html	—
1001	USSales.html	Homepage.html
1001	Books.html	USSales.html
1001	ScienceFiction.html	Books.html
1001	Mystery.html	Books.html
1001	Videos.html	USSales.html
1001	ClassicSF.html	Videos.html
1001	Search.html	Homepage.html
1001	ScienceFiction.html	Search.html

Path Analysis users can turn off Path Completion by omitting the referrer specification. This can be done by editing the referrer specification variable out of input data sets or by not declaring a referrer variable when creating an input data set for the Path Analysis node using the Enterprise Miner Create Data Source Wizard.

Path Analysis Maximal Paths

The Path Analysis node can optionally infer maximal paths from a visitor session. Maximal paths refer to the process of path completion followed by considering the subset of requests that only treat forward references as a separate path. You can enable maximal path generation in the Path Analysis node by configuring the Maximal Paths property setting in the Path Analysis properties panel to Yes.

For example, if the Maximal Paths option is enabled, the final path completion table example above is interpreted as four separate paths, as follows:

VISITOR ID	Requested Page
10001	Homepage.htm

VISITOR ID	Requested Page
10001	USSales.htm
10001	Books.htm
10001	ScienceFiction.htm

VISITOR ID	Requested Page
10001	Homepage.htm
10001	USSales.htm
10001	Books.htm
10001	Mystery.htm

VISITOR ID	Requested Page
10001	Homepage.htm
10001	USSales.htm
10001	Videos.htm
10001	ClassicSF.htm

VISITOR ID	Requested Page
10001	Homepage.htm
10001	Search.htm
10001	ScienceFiction.htm

Path Analysis Path Breaks

When performing Path Analysis with Web user logs, path completion does not always succeed. Path completion may fail, for example, if a visitor leaves the site and subsequently returns within the timeout period established by the session algorithm. Path completion may also fail if a visitor enters or travels within a Web site using browser bookmarked links instead of navigation links in the current Web page.

When path completion does not succeed, it is reported as a break in the path. Path break detection also depends on the Backup Limit property specified in the Path Analysis properties panel, since path completion depends on scanning the referring page and requested backward file sequence moves. When a visitor navigates to a page after a series of browser back button clicks, it is considered a page break if the number of back button clicks exceeds the value specified in the Path Analysis Backup Limit property. The default Backup Limit property value is 5.

Keeping Page Reload Requests in Path Analysis

Clicking the browser page reload button is a common operation during Web browsing. Clicking the browser page reload button generates a new request at the Web server. By default, the Path Analysis node ignores page reload requests in the input data. In the absence of a referring page field, if there are consecutive requests for the same page, all requests after the first one are treated as page reload requests. However, if a referrer field is specified, consecutive requests for the same page are only treated as reload requests if the referrer page is the same as the requested page. The Path Analysis node Keep Reload property, when set to Yes, keeps the reload requests for path mining. The default setting for the Keep Reload property is No.

Path Analysis Node Properties

Path Analysis Node General Properties

The following general properties are associated with the Path Analysis node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Path Analysis node added to a diagram will have a Node ID of Path. The second Path Analysis node added to a diagram will have a Node ID of Path2, and so on.
- **Imported Data** — accesses the Imported Data — Path Analysis window. The Imported Data — Path Analysis window contains a list of the ports that provide data sources to the Path Analysis node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — Path Analysis window. The Exported Data — Path Analysis window contains a list of the output data ports that the Path Analysis node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Path Analysis Node Train Properties

The following train properties are associated with the Path Analysis node:

- **Variables** — Use the Variables property of the Path Analysis node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, as well as select which variables you want included in reports.
- **Training Mode** — Use the Training Mode property to choose between Train and Score modes when the Path Analysis node runs. When the mode is set to **Train**, the node discovers a set of rules. When the mode is set to **Score**, a previously found set of rules is applied to the transaction data set. You must specify a scoring rules data set when the Training Mode property is set to **Score**. The default setting for the Training Mode property is **Train**.
- **Data Sorted** — Use the Data Sorted property to specify whether the Path Analysis node should sort the data. When path analysis is performed, the input data set must be sorted by the Target, ID, Sequence, and Referrer variables. If the input data set is already sorted, then this property should be set to **No**. Otherwise, the Data Sorted property should be set to **Yes**. The default setting for the Data Sorted property is **No**.
- **Maximal Paths** — Use the Maximal Paths property to specify whether you want to save one simplified path summary or more than one path in order to summarize complicated paths, which include backward linking. In the path **A ⇨ B ⇨ C ⇨ B ⇨ D**, both C and D are accessed from node B. If Maximal Paths is set to **No**, one path is saved as **A ⇨ B ⇨ C ⇨ B ⇨ D**. If the property is set to **Yes**, two paths are saved: **A ⇨ B ⇨ C** and **A ⇨ B ⇨ D**. The default setting for the Maximal Paths property is **No**.
- **Longest Only** — Use the Longest Only property to specify if you want to keep only the longest paths or all of the paths that are found. When the Longest Only property is set to **Yes**, if there are paths **A ⇨ B ⇨ C** and **A ⇨ B ⇨ C ⇨ D**, then only the longer path is kept. Otherwise, both paths are kept. The default setting of the Longest Only property is **No**.
- **Maximum Number of Items** — Use the Maximum Number of Items property to specify the maximum number of items that you want to consider in a path. Permissible values are nonnegative integers. The default value of the Maximum Items property is 4.
- **Backup Limit** — Use the Backup Limit property to specify the maximum number of page-back operations that you want to allow before a referrer path is declared broken. Permissible values are nonnegative integers. The default value of the Backup Limit property is 5.
- **Keep Reload** — Use the Keep Reload property to specify whether page reloads are recorded as visits. When set to Yes, Keep Reload treats page reloads as visits. The default setting for the Keep Reload property is No.
- **Maximum Visits** — Use the Maximum Visits property to specify the maximum number of page visits that you want to allow within a single path. Permissible values are nonnegative integers. The default value for the Maximum Visits property is 1000.
- **Minimum Visits** — Use the Minimum Visits property to specify the minimum number of page visits that you require to constitute a single path. Permissible values

are nonnegative integers. The default value for the Minimum Visits property is 0 (no minimum applied).

- **Maximum Number of Items to Process** — Use the Maximum Number of Items to Process property to specify the maximum number of items that you want to process during the analysis. Permissible values are nonnegative integers. The default value for the Maximum Number of Items to Process property is 100,000.

Path Analysis Node Train Properties: Support

- **Support Type** — Use the Support Type property to specify whether the path analysis should use the count support or the percentage support. The default support setting is the percentage support.
- **Count Support** — Use the Count Support property to specify the minimum transaction frequency that is required to support path analysis, expressed as a count. Permissible values are integers greater than or equal to 1. If no value is entered, the default value for the Count Support property is "." (missing value). If the Count Support property is in its default state, it is ignored and the Percentage Support option is used to determine the transaction frequency as a percentage value.
- **Percentage Support** — Use the Percentage Support property to specify the minimum transaction frequency that is required to support path analysis. The value is expressed as a percentage. Permissible values are real numbers between 1 and 100. The default value for the Percentage Support property is 5.0.

Path Analysis Node Train Properties: Rules

- **Number to Keep** — Use the Number to Keep property to define the maximum number of rules that you want to keep for the results. The default setting is 200 rules. Choices include settings for 50, 100, 200, 500, 1000, 2000, 5000, and 10000 rules.
- **Sort Criterion** — Use the Sort Criterion property to specify the statistic that you want to use to sort the generated rules. The available statistic settings are **Confidence**, **Count**, and **Support**. The default sort criterion is **Support**.
- **Pattern Match** — Use the Pattern Match property to specify whether the beginning and / or ending page in the user sequence must exactly match the beginning and / or ending page that is specified in the pattern table. A pattern table must be specified in order for the Pattern Match property to take effect. The default setting for the Pattern Match property is **No**.
- **Match LHS** — Use the Match LHS property to specify if scoring should identify visitors that meet only the left-hand side of the submitted sequence pattern. In order to meet this criterion, the visitors should exactly follow the left-hand side of the sequence, with the exception of the very last click. A scoring rules data set must be specified in order for the Match LHS property to take effect. The default setting for the Match LHS property is **No**.
- **Number to Transpose** — Use the Number to Transpose property to control the number of rules that will be transposed and used to create the train data set (Rule By ID data set). The default setting is 200 rules. Choices include settings for 50, 100, 200, 500, 1000, 2000, and 5000 rules.
- **Export Rule by ID** — Set the Export Rule by ID property to **Yes** if you want to export the Rule by ID data set as a training data set. The Rule by ID data set consists of rule variables that map the sequence rules associated with each customer ID. The default setting for the Export Rule by ID property is **No**.

Path Analysis Node Train Properties: Data Sets

- **Restricted Paths** — Select the … button to open the Select a SAS Table window to select a SAS data set to use as a Restricted Paths pattern table. See "Restrict Patterns" on page 515 for information on Restricted Paths table creation and formatting. The default setting for the Restricted Paths property is no table specified.
- **Transition Probabilities** — Select the … button to open the Select a SAS Table window to select a SAS data set to use as a Transition Probabilities table. See "Transition Probabilities" on page 514 for information on Transition Probabilities pattern table creation and formatting. The default setting for the Transition Probabilities property is no table specified.
- **Funnel Counts** — Select the … button to open the Select a SAS Table window to select a SAS data set to use as a Funnel Counts pattern table. See "Funnel Counts" on page 515 for information on Funnel Counts pattern table creation and formatting. The default setting for the Funnel Counts property is no table specified.
- **Recount Paths** — Select the … button to open the Select a SAS Table window to select a SAS data set to use as a Recount Paths pattern table. See "Recount Paths" on page 514 for information on Recount Paths pattern table creation and formatting. The default setting for the Recount Paths property is no table specified.
- **Scoring Rules** — Select the … button to open the Select a SAS Table window to select a SAS data set to use as a Scoring Rules pattern table. See "Path Scoring" on page 513 for information on Scoring Rules table creation and formatting. The default setting for the Scoring Rules property is no table specified.

Path Analysis Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Path Analysis Node Results

You can open the Results window of the Path Analysis node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

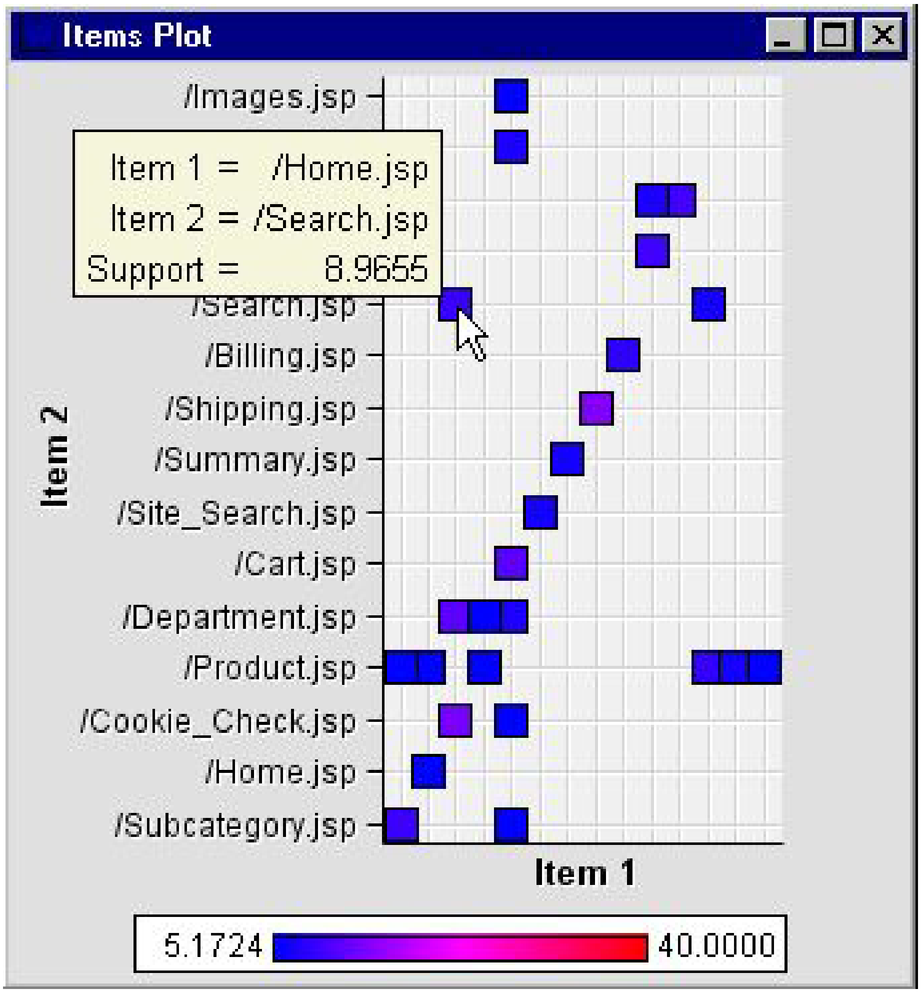
Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Path Analysis node properties configuration when the node was last run.

  - **Run Status** — indicates the status of the Path Analysis node run. The run start time, run duration, and completion status is displayed in this window.
  - **Variables** — The Variables setting in the menu opens a window containing a table of the variables submitted to the Path Analysis node. The variables table displays the role, level, and use settings for each variable.
  - **Train Code** — the code that Enterprise Miner uses to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — the SAS log of the Path Analysis node run.
  - **Output** — the SAS output of the Path Analysis run. For the Path Analysis node, SAS output includes a variable summary, an items report, a path report, funnel counts, and rule statistics from PROC MEANS and PROC FREQ. More information is available for the "Output Window" on page 528 .
  - **Flow Code** — the Path Analysis node does not produce SAS flow code.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the Path Analysis node. The Path Analysis node SAS code can be edited and used as input data sets for subsequent Path Analysis node runs or in external applications outside the Enterprise Miner environment. The data in the Publish Code window can be used for Recount Paths, Score, Funnel Count, and Restrict Patterns input data sets.
  - **PMML Code** — the Path Analysis node does not generate PMML code.
- **Rules** — the rules tables and charts associated with a Path Analysis node run. To see the data table that SAS uses to produce a chart, select the chart and from the Results window main menu, select **View** ⇨ **Table**. The column headings of the rules table display variable labels by default. To display the variable names used in SAS code as column headings, right-click in any table cell and select **Show Variable Names**. To view labels in the column headings again, right-click in any table cell and select **Show Variable Labels**.
  - **Rule Description** — displays text descriptions of the rules.
  - **Rules Table** — displays the Rules table from the Path Analysis node run. The Rules table contains the columns below.
    - **Rule Identifier** — an integer value that uniquely identifies a sequence rule.
    - **Chain Size** — the number of items in the sequence.
    - **Count** — the number of times that the sequence occurs in the data set.
    - **Support** — a measure of how frequently the sequence occurs in the data set. Support is the number of times the sequence occurs in the data set, divided by the total number of sequences.
    - **Confidence** — the confidence rule measure. In the sequence **A** ⇨ **B**, the confidence rule measure is the percentage of times that event B occurs after event A occurs.
    - **ITEM1** — the first item in the rule.
    - **ITEM2** — the second item in the rule.
    - **ITEM3** — the third item in the rule.
    - **ITEM4** — the fourth item in the rule.

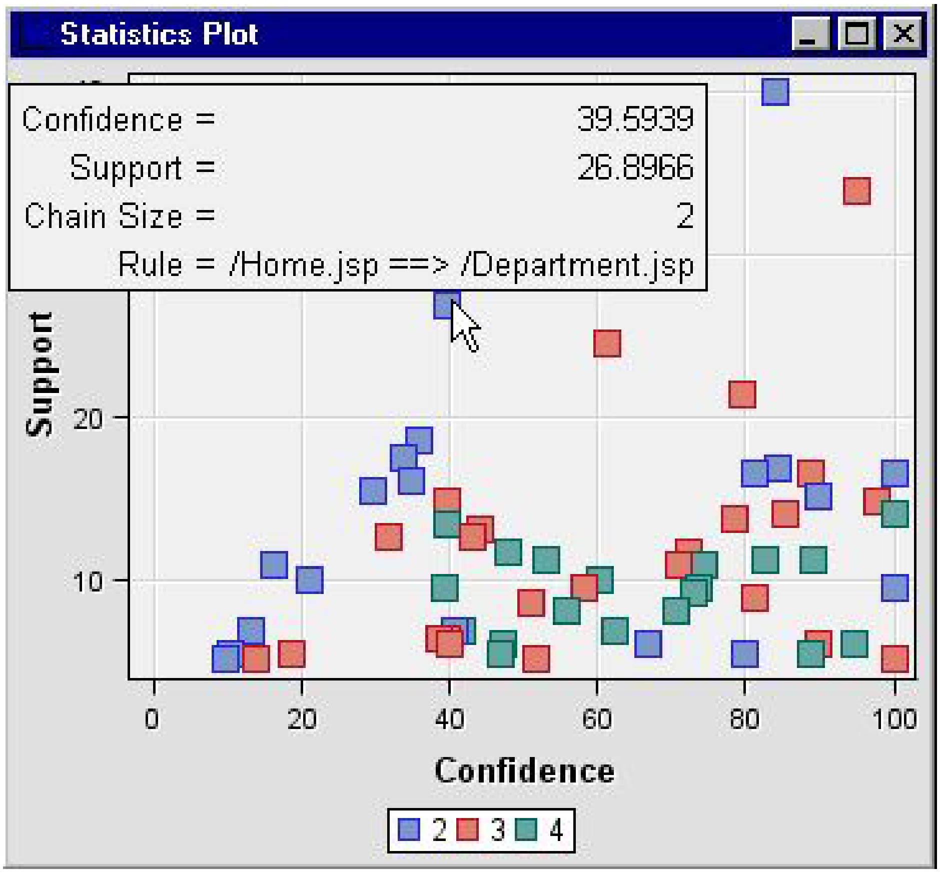
  - **Rule** — the sequence rule.

- **Items Plot** — Displays a scatter plot (statistics histogram) of the items in a rule. The plot is shaded by the rule support level. Hovering your pointer over a marker on the plot displays the items in the rule and the support level.



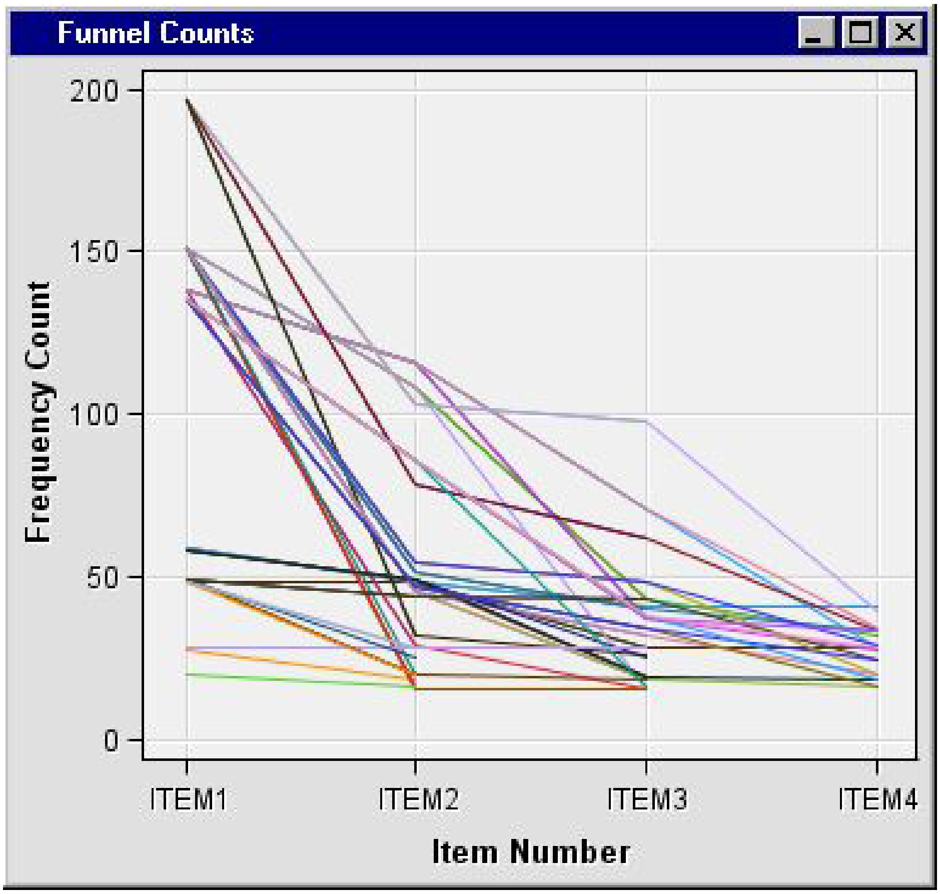
Item 1 = /Home.jsp
Item 2 = /Search.jsp
Support = 8.9655

- **Statistics Plot** — displays a scatter plot of confidence and support levels in generated rules. Hovering your pointer over a marker in the plot displays the rule and its confidence and support levels.

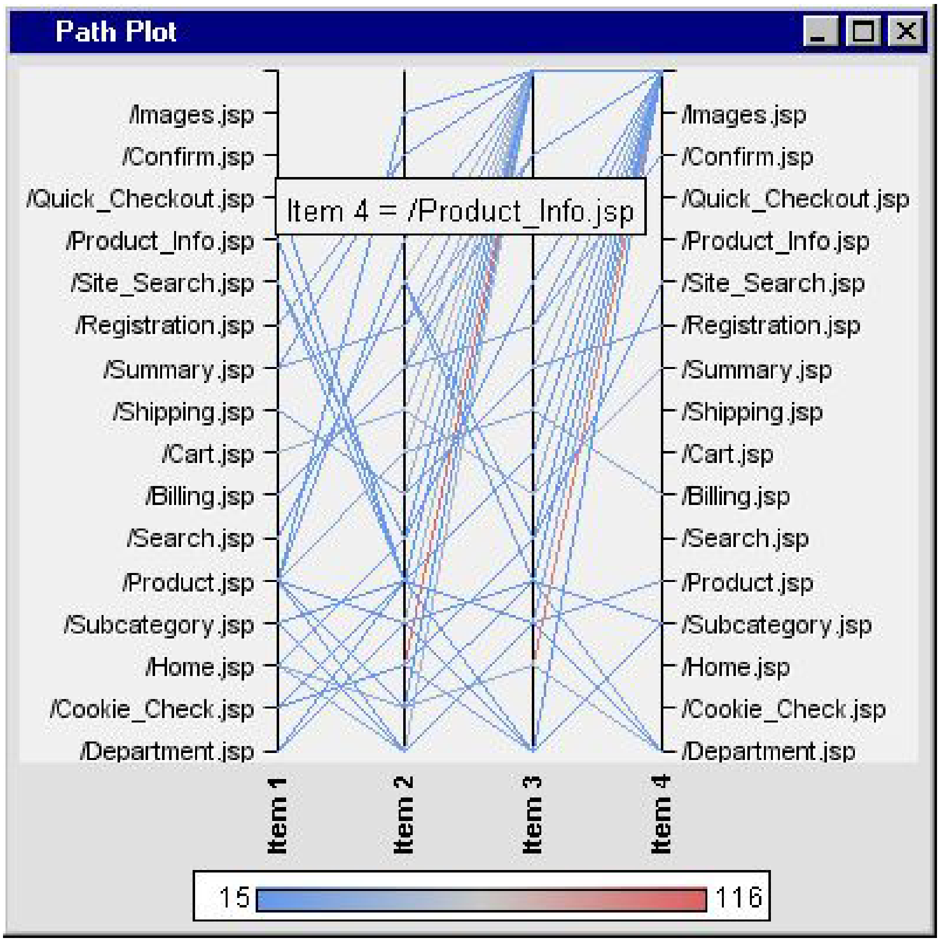


Confidence = 39.5939
Support = 26.8966
Chain Size = 2
Rule = /Home.jsp ==> /Department.jsp

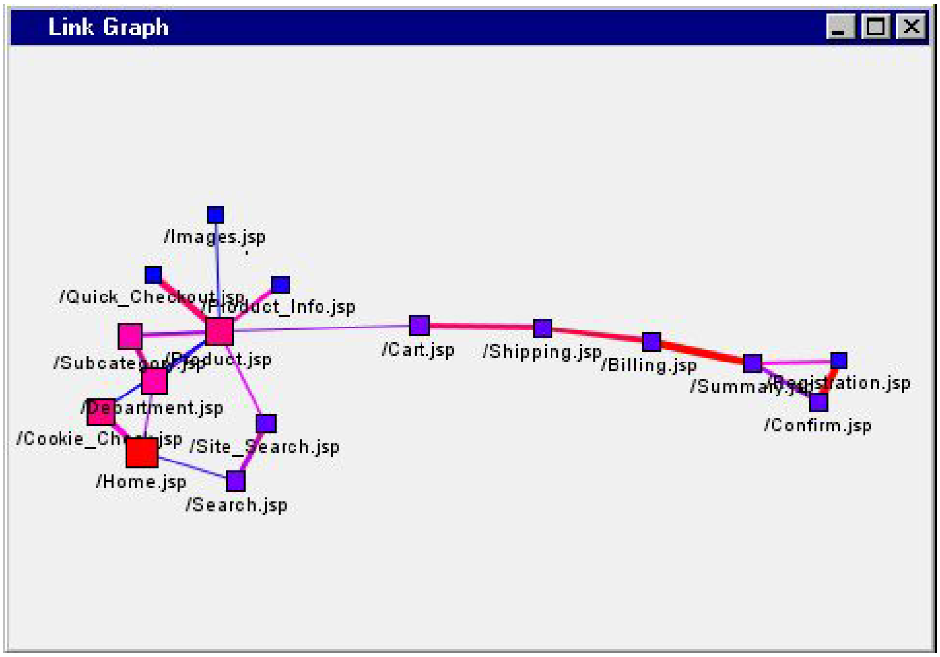
- **Funnel Counts** — displays a line plot of the funnel counts by item number from the Path Analysis run.



- **Path Plot** — displays a line plot depicting the visitor transition vector frequencies between different locations. The data set that underlies the path plot is the Rules table.



- **Link Graph** — displays a link graph of the Path Analysis results. Two data tables are used to produce the link graph: a node data table and a link data table. To view the link graph, select **View** ⇨ **Rules** ⇨ **Link Graph** from the Path Analysis Results window main menu. A *link graph* is a graphical representation of the data in the Rules data set. Here is an example of a link graph:



A link graph has the following features:

- Node size and color — determined by the count of the item in the data set. The count measures how frequently the item is requested.
- Link color and thickness — determined by the confidence value. The higher the confidence, the thicker the link is between nodes. The link represents a connection between two items in a rule. Links do not have direction in the link graph.

- **Table** — enables you to open a table of the data that is associated with the graph that you have open. Data table column headings display variable labels by default. To display the variable names used in SAS code as column headings, right-click in any table cell and select Show Variable Names. To view labels in the column headings again, right-click in any table cell and select Show Variable Labels.
- **Plot** — enables you to create or modify a graph of the data in the table that you have open.

Output Window

The Output window displays the results of the PATH procedure.

The window contains the following sections:

- **Variable Summary** — a listing of the roles and levels of the input variables.
- **Items Report** — a table of the items that were determined by the procedure.

  The table contains the following columns:

  - **Transaction Item** — the filename of the item.
  - **Transaction Count** — the number of times the item occurred in the training data set.
  - **Transaction Support (%)** — the number of visitor sessions exhibiting a pattern divided by the total number of visitor sessions.
- **Path Report** — a table of the paths that were determined by the procedure.

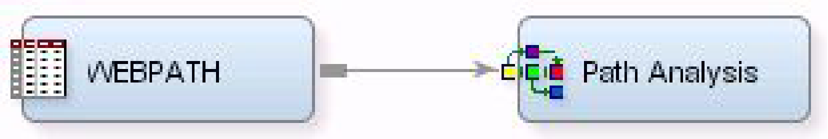
  The table contains the following columns:

  - **Chain Size** — the number of items in the path (rule).
  - **Count** — the frequency of the observed chain.
  - **Support** — the number of times the rule occurred in the data set, divided by the total number of rules in the data set.
  - **Confidence** — the confidence of the rule. In the rule **A** ⇨ **B**, confidence is the percentage of times that event B occurs after event A occurs.
  - **Rule** — the path.
- **Funnel Counts** — reports a rule identifier, the rule associated with the identifier, and under the headings Item1, Item2, Item3 and Item4, the number of sessions during which a visitor progressed to that point in the path.
- **Rule Statistics** — the results of the MEAN procedure for the variables Size, Count, Support, and Conf, and the results of the FREQ procedure that is run on the Size variable.

Example: Analyzing Web Log Data

Introduction

The following example uses the Path Analysis node to analyze a fictional, pre-processed Web log. The data in the Web log represents the activity of customers at an online store. Follow the steps below to create the process flow diagram.



- "Define the Data Source" on page 529
- "Add the Nodes to the Diagram Workspace" on page 529
- "Modify the Path Analysis Node Settings and Run the Node" on page 529
- "View the Results" on page 530

Define the Data Source

1. From the main menu, select **File** ⇨ **New** ⇨ **Data Source**.
2. In the Metadata Source window, select **SAS Table** as the **Source**, and click **Next**.
3. In the Derive Metadata From SAS Table window, type `SAMPSIO.WEBPATH` in the **Table** field. Click **Next**.
4. Click **Next** in the Table Metadata window.
5. In the Metadata Configuration window, select the **Advanced** button. Click **Next**.
6. Set the variable roles in the Columns Metadata window.
   1. Set the Role of referrer to Referrer.
   2. Set the Role of requested_file to Target.
   3. Set the Role of session_id to ID.
   4. Set the Role of session_sequence to Sequence.
   5. Click Next.
7. Select No in the Decision Processing window, and click **Next**.
8. In the Data Source Attributes window, set the data set Role to **Transaction**. Click **Finish**.

Add the Nodes to the Diagram Workspace

1. In the Project Panel, expand the Data Sources folder. Drag the WEBPATH data set to the Diagram Workspace.
2. Add a Path Analysis node to the Diagram Workspace.
3. Connect the Input Data node to the Path Analysis node.

Modify the Path Analysis Node Settings and Run the Node

1. Click the Path Analysis node to select it.

2. Set the Export Rule by ID property to **Yes**.
3. Right-click the Path Analysis node and select **Run**.

View the Results

1. Right-click the Path Analysis node and select **Results**.
2. In the Output window, scroll down to the Items Report section.

```
Items Report

                         Transaction    Transaction
Target Item                    Count    Support(%)

/Billing.jsp                      48      16.5517
/Cart.jsp                         59      20.3448
/Catalog_Order.jsp                15       5.1724
/Catalog_Request.jsp              22       7.5862
/Confirm.jsp                      50      17.2414
/Cookie_Check.jsp                151      52.0690
/Department.jsp                  138      47.5862
/Home.jsp                        197      67.9310
/Images.jsp                       16       5.5172
/Member_Home.jsp                  16       5.5172
/Product.jsp                     151      52.0690
/Product_Info.jsp                 27       9.3103
/Quick_Checkout.jsp               20       6.8966
/Registration.jsp                 28       9.6552
/Retail_Store.jsp                 22       7.5862
/Search.jsp                       58      20.0000
/Shipping.jsp                     49      16.8966
/Site_Search.jsp                  49      16.8966
/Subcategory.jsp                 135      46.5517
/Summary.jsp                      48      16.5517
```

3. In the Items Report section, notice that the Item with the highest count is Home.jsp. Home.jsp is usually the first page that a visitor will view. In most cases the home page of the Web site will have the most hits.
4. Scroll down to the Path Report section.

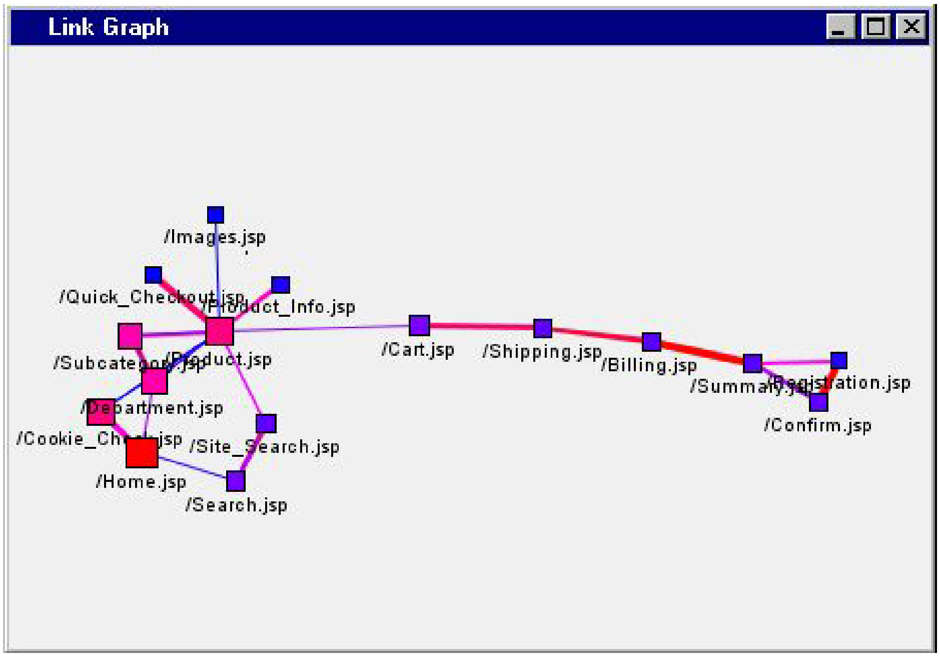
```
Path Report

Chain
Size     Count     Support    Confidence    Rule

   2       116     40.0000      84.0580     /Department.jsp ==> /Subcategory.jsp
   2       108     37.2414      71.5232     /Cookie_Check.jsp ==> /Home.jsp
   2       103     35.5172      52.2843     /Home.jsp ==> /Cookie_Check.jsp
   3        98     33.7931      95.1456     /Home.jsp ==> /Cookie_Check.jsp ==> /Home.js
   2        86     29.6552      63.7037     /Subcategory.jsp ==> /Product.jsp
   2        78     26.8966      39.5939     /Home.jsp ==> /Department.jsp
```

The path (rule) that has the highest count was Department.jsp to Subcategory.jsp. Note that this rule also had high confidence, which indicates that there is an 84% chance that when a visitor clicks /Department.jsp, they will then click /Subcategory.jsp.

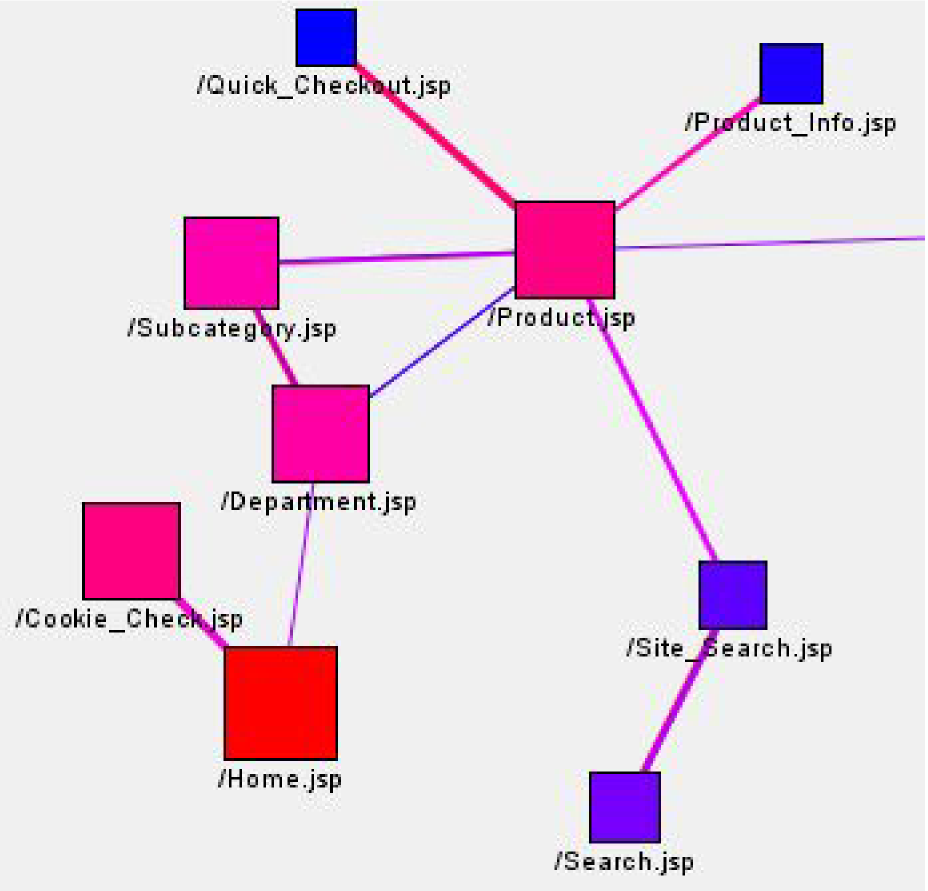
5. Close the Output window.

6. From the Results window main menu, select **View** ⇨ **Rules** ⇨ **Link Graph**.

7. View the link graph in the Link Graph window.



Right-click the background of the Link Graph window, and select **Action Mode** ⇨ **Viewport**. Use the magnifying glass icon to enlarge the cluster of links on the left side of the link graph so you can see the connections in more detail.



The objects in the link graph use color to indicate higher values. As in a heat-map, the hues ranges from colder shades of blue and violet to warmer shades of magenta, pink, and red. The values that are associated with the nodes and link lines increase with the heat map color warmth. Blues indicate smaller values and reds indicate larger values. To find interesting path predictors, look for larger-sized nodes with "warm" shades of pink and red that are connected by thick lines in shades of pink and red. These will be the links between nodes with the highest combinations of count, support, and confidence.

Three links invite closer inspection, based on heat map color, proximity, node size, and link line thickness: /Department.jsp to /Subcategory.jsp, /Home.jsp to /Cookie_Check.jsp, and /Department.jsp to /Home.jsp.

Working together with the Rules Table of the Path Analysis Results window, we observe that the path /Department.jsp to /Home.jsp has 26.9% support and 39.6% confidence. The two strongest paths are /Department.jsp to /Subcategory.jsp with 40.0% support and 84.1% confidence, and /Home.jsp to /Cookie_check.jsp, with 37.2% support and 71.5% confidence.

8. Some links in the graph are not displayed because by default, the confidence threshold is set at 20%. You can change this value by right-clicking the graph and selecting **Graph Properties**. In the Properties — Links window, you can select the Show Threshold Slider check box in the Links properties and click **Apply** to add a slider bar to the bottom of the link graph. When you move the slider to the left, it increases the confidence threshold for displayed links, and only the links that meet or exceed the confidence threshold level that is indicated by the slider are displayed in the plot.

Chapter 37
SOM/Kohonen Node

SOM/Kohonen Node

Overview of the SOM/Kohonen Node

Introduction

The SOM/Kohonen node belongs to the Explore category of the SAS SEMMA (Sample, Explore, Modify, Model, Assess) data mining process. You use the SOM/Kohonen node to perform unsupervised learning by using Kohonen vector quantization (VQ), Kohonen self-organizing maps (SOMs), or batch SOMs with Nadaraya-Watson or local-linear smoothing. Kohonen VQ is a clustering method, whereas SOMs are primarily dimension-reduction methods. For cluster analysis, the Clustering node is recommended instead of Kohonen VQ or SOMs.

Note: The term *step* applies to the computations that are done while reading a single case and updating the cluster seeds. The term *iteration* applies to the computations that are done while reading the entire data set once and updating the cluster seeds.

CAUTION:

Frequency and weight variables should be used with care when you are doing Kohonen training.

If an observation has frequency F and weight W, then a step taken with learning rate L is equivalent to a step for an observation with frequency 1.0 and weight 1.0 that uses a learning rate of 1-(1-L)FW. This is equivalent to applying the Kohonen update FW times.

Kohonen Vector Quantization

Vector quantization networks are competitive networks that can be viewed as unsupervised density estimators or autoassociators (Kohonen, 1995/1997; Hecht-Nielsen 1990), closely related to k-means cluster analysis (MacQueen, 1967; Anderberg, 1973; use the Clustering node for k-means analyses). Each competitive unit corresponds to a cluster, the center of which is called a codebook vector or cluster seed. Kohonen's learning law is an online algorithm that finds the cluster seed closest to each training case and moves that "winning" seed closer to the training case. The seed is moved some proportion of the distance between it and the training case; the proportion is specified by the learning rate. Let C_j^s be the seed for the j^th^ cluster on the s^th^ step, X_i be the input vector for the i^th^ training case, and L_s be the learning rate for the s^th^ step. On each step, a training case X_i is selected, and the index n of the winning cluster is determined by

$$n = \arg\min_j \left\| C_j^s - X_i \right\|$$

The Kohonen update formula is

$$C_n^{s+1} = C_n^s (1 - L^s) + X_i L^s$$

For all nonwinning clusters, $C_j^s + 1 = C_j^s$. Numerous similar algorithms have been developed in the neural net and machine learning literature; see Hecht-Nielsen (1990) for a brief historical overview and Kosko (1992) for a more technical overview of competitive learning.

MacQueen's online k-means algorithm is essentially the same as Kohonen's learning law, except that in MacQueen's algorithm the learning rate is the reciprocal of the number of cases that have been assigned to the winning cluster. Suppose that when processing a given training case, N_n cases have been previously assigned to the winning seed. Then, the seed is updated as

$$C_n^{s+1} = C_n^s \frac{N_n}{N_n + 1} + X_i \frac{1}{N_n + 1}$$

This reduction of the learning rate makes each seed the mean of all cases that are assigned to its cluster, and it guarantees convergence of the algorithm to an optimum value of the error function (the sum of squared Euclidean distances between cases and seeds) as the number of training cases goes to infinity. Kohonen's learning law with a fixed learning rate does not converge. As is well known from stochastic approximation theory, convergence requires the sum of the infinite sequence of learning rates to be infinite, while the sum of squared learning rates must be finite (Kohonen, 1995, p. 34). These requirements are satisfied by MacQueen's k-means algorithm. To perform MacQueen's k-means algorithm, use the Clustering node and set the Seed Initialization Method to MacQueen.

Kohonen VQ is often used for offline learning (as in SAS Enterprise Miner), in which case the training data is stored and Kohonen's learning law is applied to each case in turn, cycling over the data set many times (incremental training). Convergence to a local optimum can be obtained as the training time goes to infinity if the learning rate is reduced in a suitable manner as described above. However, there are offline k-means algorithms, both batch and incremental, that converge in a finite number of iterations

(Anderberg, 1973; Hartigan, 1975; Hartigan and Wong, 1979). The batch algorithms such as Forgy's algorithm (1965) and Anderberg (1973) have the advantage for large data sets, because the incremental methods require you either to store the cluster membership of each case or to do two nearest-cluster computations as each case is processed.

Forgy's algorithm, the default method in the Clustering node, is a simple alternating least-squares algorithm that consists of the following steps:

1. Initialize the seeds.
2. Repeat the following two steps until convergence:
   - Read the data and assign each case to the nearest (using Euclidean distance) seed.
   - Replace each seed with the mean of the cases that were assigned to it.

Fastest training is usually obtained if MacQueen's online algorithm is used for the first pass and offline k-means algorithms are applied on subsequent passes. However, these training methods do not necessarily converge to a global optimum of the error function. The chance of finding a global optimum can be improved by using rational initialization (SAS Institute Inc., 1989, pp. 824-825), multiple random initializations, or various time-consuming training methods that are intended for global optimization (Ismail and Kamel, 1989; Zeger, Vaisy, and Gersho, 1992).

Vector quantization (VQ) has been a popular topic in the signal processing literature, which has been largely separate from the literature on Kohonen networks and from the cluster analysis literature in statistics and taxonomy. In signal processing, online methods such as Kohonen's and MacQueen's are called adaptive vector quantization (AVQ), while offline k-means methods go by the names of Lloyd-Max (Lloyd, 1982; Max, 1960) and LBG (Linde, Buzo, and Gray, 1980). A textbook on VQ by Gersho and Gray (1992) summarizes these algorithms as information compression methods.

Kohonen's learning law is an approximation to the k-means model, which is an approximation to normal mixture estimation by maximum likelihood, assuming that the mixture components (clusters) all have spherical covariance matrices and equal sampling probabilities. Therefore, if the population contains clusters that are not equiprobable, k-means will tend to produce sample clusters that are more nearly equiprobable than the population clusters. Corrections for this bias can be obtained by maximizing the likelihood without the assumption of equal sampling probabilities (Symons, 1981).

In cluster analysis, the purpose is not to compress information but to recover the true cluster memberships. k-means algorithms differ from mixture models: in k-means, the cluster membership for each case is considered a separate parameter to be estimated while mixture models estimate a posterior probability for each case that is based on the means, covariances, and sampling probabilities of each cluster. Balakrishnan, Cooper, Jacob, and Lewis (1994) found that k-means algorithms recovered cluster membership more accurately than Kohonen VQ.

Self-Organizing Maps

Self-organizing maps (SOMs) are competitive networks that provide a topological mapping from the input space to the clusters (Kohonen, 1995). Kohonen (1995, p. vii) says that SOMs are intended for clustering, visualization, and abstraction. The SOM was inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that spatial or other relations among stimuli correspond to spatial relations among the neurons. In a SOM, the neurons (clusters) are organized into a grid, usually two-dimensional, but sometimes one-dimensional or (rarely) three-dimensional or more. The grid exists in a space that is separate from the input space; any number of inputs can be used, as long as the number of inputs is greater than the dimensionality of the grid space. A SOM tries to find clusters such that any two clusters that are close to each other in the grid space have seeds close to each other in the input space. But the converse does not hold: seeds that are close to each other in the input space do not necessarily correspond to clusters that are close to each other in the grid. Another way to look at this is that a SOM tries to embed the grid in the input space such that every training case is close to some seed, but the grid is bent or stretched as little as possible. Yet another way to look at it is that a SOM is a (discretely) smooth mapping between regions in the input space and points in the grid space. The best way to understand this is to look at the pictures in Kohonen (1995) or various other neural network textbooks.

The Kohonen algorithm for SOMs is very similar to the Kohonen algorithm for AVQ. The Kohonen SOM algorithm requires a kernel function $K^s(j,n)$, where $K^s(j,j)=1$ and each $K^s(j,n)$ is usually a non-increasing function of the distance (in any metric) between seeds j and n in the grid space (not the input space). Usually, $K^s(j,n)=0$ for seeds that are far apart in the grid space. As each training case is processed, all the seeds are updated as

$$C_n^{s+1} = C_n^s(1 - K^s(j,n)L^s) + X_i K^s(j,n)L^s$$

As indicated by the superscript s, the kernel function typically changes during training. The neighborhood of a given seed is the set of seeds for which $K^s(j,n)>0$. To avoid poor results (akin to local minima), it is usually advisable to start with a large neighborhood and to let the neighborhood gradually shrink during training. If $K^s(j,n)=0$ for j not equal to n, then the SOM update formula reduces to the formula for Kohonen vector quantization. In other words, if the neighborhood size (for example, the radius of the support of the kernel function) is zero, then the SOM algorithm degenerates into simple VQ. Therefore, it is important not to let the neighborhood size shrink all the way to zero during training if you want topological mapping. Indeed, the choice of the final neighborhood size is the most important tuning parameter for SOM training.

A SOM works by smoothing the seeds in a manner similar to kernel estimation methods, but the smoothing is done in neighborhoods in the grid space rather than in the input space (Mulier and Cherkassky, 1995). This is easier to see in a batch algorithm for SOMs, which is similar to Forgy's algorithm for batch k-means, but incorporates an extra smoothing process.

1. Initialize the seeds.
2. Repeat the following two steps until convergence or boredom:
   - Read the data, assigning each case to the nearest (using Euclidean distance) seed. While you are doing this, keep track of the mean and the number of cases for each cluster.
   - Do a nonparametric regression using $K^s(j,n)$ as a kernel function, with the grid points as inputs, the cluster means as target values, and the number of cases in each cluster as a case weight. Replace each seed with the output of the nonparametric regression function evaluated at its grid point.

If the nonparametric regression method is Nadaraya-Watson kernel regression (Wand and Jones, 1995), then the batch SOM algorithm produces essentially the same results as Kohonen's algorithm, barring local minima. The main differences are that the batch algorithm often converges and requires no learning rate to be specified. Mulier and Cherkassky (1995) note that other nonparametric regression methods can be used to provide superior SOM algorithms. In particular, local-linear smoothing eliminates the notorious border effect, whereby the seeds near the border of the grid are compressed in the input space. The border effect is especially problematic when you try to use a high degree of smoothing in a Kohonen SOM, because all the seeds will collapse into the center of the input space. The SOM border effect has the same mathematical cause as the boundary effect in kernel regression, which causes the estimated regression function to flatten out near the edges of the regression input space. There are various cures for the edge effect in nonparametric regression; local-linear smoothing is the simplest (Wand and Jones, 1995). Hence, local-linear smoothing also cures the border effect in SOMs. Furthermore, local-linear smoothing is much more general and reliable than the heuristic weighting rule that was proposed by Kohonen (1995, p. 129).

Because nonparametric regression is used in the batch SOM algorithm, various properties of nonparametric regression extend to SOMs. In particular, it is well known that the shape of the kernel function is not a crucial matter in nonparametric regression; hence, it is not crucial in SOMs. On the other hand, the amount of smoothing used for nonparametric regression is crucial, so the choice of the final neighborhood size in a SOM is crucial. Unfortunately, there do not seem to be any systematic studies of methods for choosing the final neighborhood size.

The batch SOM algorithm is very similar to the principal curve and surface algorithm that was proposed by Hastie and Stuetzle (1989), as pointed out by Ritter, Martinetz, and Schulten (1992) and Mulier and Cherkassky (1995). A principal curve is a nonlinear generalization of a principal component.

Given the probability distribution of a population, a principal curve is defined by the following self-consistency condition:

1. If you choose any point on a principal curve,
2. then find the set of all the points in the input space that are closer to the chosen point than any other point on the curve,
3. and compute the expected value (mean) of that set of points with respect to the probability distribution, then
4. the result is the same point on the curve that you chose originally.

In a multivariate normal distribution, the principal curves are the same as the principal components. A principal surface is the obvious generalization from a curve to a surface. In a multivariate normal distribution, the principal surfaces are the subspaces that are spanned by any two principal components.

A one-dimensional local-linear SOM can be used to estimate a principal curve by letting the number of seeds approach infinity while choosing a kernel function of appropriate smoothness. A two-dimensional local-linear SOM can be used to estimate a principal surface by letting the number of both rows and columns in the grid approach infinity. This connection between SOMs and principal curves and surfaces shows that the choice of the number of seeds is not crucial, provided that this number is fairly large.

In a Kohonen SOM, as in VQ, it is necessary to reduce the learning rate during training to obtain convergence. Another advantage of batch SOMs is that there is no learning rate.

The one advantage of Kohonen SOMs over batch SOMs is that Kohonen SOMs are less likely to get stuck in bad configurations (akin to local optima) for highly nonlinear data. Nadaraya-Watson SOMs are also less prone to bad configurations than local-linear SOMs.

For many realistic data sets, however, the principal curves differ little from the principal components. As a result, it is possible to avoid bad configurations by initializing the seeds to regularly spaced points along the first principal component (for a 1-D SOM) or in the plane of the first two principal components (for a 2-D SOM). It is still advisable to use a large initial neighborhood size and shrink it during training.

Hence, if you train a non-Kohonen SOM, then the default behavior of the SOM/Kohonen node is as follows:

- The seeds are initialized to an evenly spaced grid in the plane of the first two principal components.
- Batch training is used, first with Nadaraya-Watson smoothing, then with local-linear smoothing.

To get good topological ordering, it is advisable to specify a final neighborhood size greater than one. Determining a good neighborhood size usually requires trial and error.

For highly nonlinear data, use a Kohonen SOM, which by default behaves as follows:

- The initial seeds are randomly selected cases.
- The initial neighborhood size is set to half the size of the SOM. The neighborhood size is gradually reduced to zero during the first thousand training steps.
- Incremental training is used. The learning rate is initialized to 0.9 and linearly reduced to 0.02 during the first thousand training steps.

You can usually get better convergence by specifying, in addition to Kohonen training, one or both of the batch training options for Nadaraya-Watson smoothing or local-linear smoothing. Batch training often converges but sometimes does not. You can specify any combination of the Kohonen, Nadaraya-Watson, and local-linear training, which are always applied in that order.

SOM/Kohonen Node Data Set Requirements

The input variables for the SOM/Kohonen node can be interval, binary, nominal, and ordinal. However, if you have many categorical variables (binary, nominal, or ordinal), it takes longer to run the node.

SOM/Kohonen Node Properties

SOM/Kohonen Node General Properties

The following general properties are associated with the SOM/Kohonen node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first SOM/Kohonen node added to a diagram has a Node ID of SOM. The second SOM/Kohonen node added to a diagram has a Node ID of SOM2.
- **Imported Data** — accesses the Imported Data — SOM/Kohonen window. The Imported Data — SOM/Kohonen window contains a lists of the ports that provide data sources to the SOM/Kohonen node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Exported Data** — accesses the Exported Data — SOM/Kohonen window. The Exported Data — SOM/Kohonen window contains a list of the output data ports that the SOM/Kohonen node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

SOM/Kohonen Node Train Properties

The following train properties are associated with the SOM/Kohonen node:

- **Variables** — Use the Variables property to specify the properties of the variables that you want to use from the data source. Select the ... button to open a variables table.
- **Method** — Use the Method property to specify one of the following network methods.

  - **Batch SOM** — For the Batch SOM method, larger maps are usually better, as long as most clusters have at least five or ten cases. However, the larger the map, the longer it will take to train. You may specify the number of rows and columns in the topological map using the Row and Column properties, and you specify the final neighborhood size using the Final Size property in the Neighborhood Options table. The final neighborhood size should usually be set in proportion to the map size. For example, if you double the number of rows and columns in the map, you should double the final neighborhood size. Choosing the map size and final neighborhood size generally requires trial and error. Batch SOM is the default setting for the Method property.
  - **Kohonen SOM** — For the Kohonen SOM method, larger maps are usually better, as long as most clusters have at least five or ten cases. However, the larger the map, the longer it will take to train. You may specify the number of rows and columns in the topological map using the Row and Column properties, the final neighborhood size using the Final Size property in the Neighborhood Options table, and the learning rate parameters by using the Learning Rate, Initial Rate, Final Rate, and Number of Steps properties in the Kohonen Options table. The final neighborhood size should usually be set in proportion to the map size. For example, if you double the number of rows and columns in the map, you should double the final neighborhood size. Choosing the map size and final neighborhood size generally requires trial and error. The learning rate changes during training. If the initial seeds are random, then it is important to start with a high learning rate, such as 0.9. If the initial seeds are obtained by using principal components or some preliminary analysis, then the initial learning rate should be much lower.
  - **Kohonen VQ** — For the Kohonen VQ method, you may specify the number of clusters by using the Maximum Number of Clusters property. Choosing the number of clusters generally requires trial and error. You may specify the

learning rate by using the Learning Rate property in the Kohonen Options table. The learning rate changes during training. If the initial seeds are random, then it is important to start with a high learning rate, such as 0.5. If the initial seeds are obtained through some preliminary analysis, then the initial learning rate should be much lower.

- **Internal Standardization** — Use this property to specify one of the following internal standardization methods.
  - **None** — specifies that the variables are not standardized prior to clustering.
  - **Range** — specifies that the minimum value be subtracted from the variable values and then that the variable values be divided by the range.
  - **Standardization** — specifies that the mean be subtracted from the variable values and then that the variable values be divided by (standard deviation) * SQRT(dimension). Dimension refers to the number of unique categories. For nominal variables, the default dimension is C, the number of categories. The default dimension for ordinal variables is 1. This is the default setting for the **Internal Standardization** property.

SOM/Kohonen Node Train Properties: Segment

Finding a good map size generally requires trial and error. If the map size is too small, then it will not accurately reflect nonlinearity in the data. If the map size is too large, the analysis will take a lot of computer time and memory, and empty clusters might produce misleading results. If you change the map size, then you might need to change the neighborhood size in proportion to the map size, because the amount of smoothing depends on the ratio of the neighborhood size to the map size.

- **Row** — Use the Row property to specify the number of rows in the topological map. The default value is 10. A value of 1 creates a one-dimensional map.
- **Column** — Use the Column property to specify the number of columns in the topological map. The default value is 10. A value of 1 creates a one-dimensional map.

SOM/Kohonen Node Train Properties: Seed Options

- **Initial Method** — Use the Initial Method property to specify the seed initialization method.

  The following initialization methods are available:

  - **Default** — uses the Princomp method to initialize seeds if the network method is set to BatchSOM, and the Outlier method if the network method is set to KohonenSOM or KohonenVQ.
  - **Outlier** — selects initial seeds that are well separated using the full replacement algorithm.
  - **First** — selects the first complete cases as initial seeds. The next complete case that is separated from the first seed by at least the specified radius becomes the second seed. Subsequent cases are selected as new seeds if, and only if, they are separated from all previous seeds by at least the radius, and if the maximum number of seeds is not exceeded.
  - **Separate** — selects initial seeds that are well separated using a partial replacement algorithm. This method selects the first complete case as the first seed. The next complete case that is separated from the first seed by at least the specified radius becomes the second seed. Subsequent cases are selected as new seeds if they are separated from all previous seeds by at least the radius as long

as the maximum number of seeds is not exceeded. If a case is complete but fails to qualify as a new seed, this case is then considered to replace one of the old seeds. An old seed is replaced if the distance between the case and the closest seed is greater than the minimum distance between seeds. The seed that is replaced is selected from the two seeds that are closest to each other. The seed that is replaced is the one of these two seeds that has the shortest distance to the closest of the remaining seeds when the other seed is replaced by the current observation.

  - **Principal Components** — initializes seeds on an evenly spaced grid in the plane of the first two principal components. If the number of rows is less than or equal to the number of columns, then the first principal component is placed to vary with the column number, and the second principal component is placed to vary with the row number. If the number of rows is greater than the number of columns, then the first principal component is placed to vary with the row number, and the second principal component is placed to vary with the column number.

- **Radius** — Use the Radius property to specify the minimum distance between cluster seeds. The Radius property accepts real numbers. The default setting is 0.0.

SOM/Kohonen Node Train Properties: Batch SOM Training

When the Method property is set to **Batch SOM** or **Kohonen SOM**, use the following Batch SOM Training Properties to specify advanced batch training options.

- **Use Defaults** — Use this property to specify whether to use the default batch SOM training options to run the node. The default batch SOM training options depend on the type of network that you specified in the Method property.

  - If the Method property is set to **Batch SOM**, SOM training is always performed, and both the local-linear and Nadaraya-Watson smoothing methods are applied.

  - If the Method property is set to **Kohonen SOM**, SOM training is not performed, and neither the local-linear nor Nadaraya-Watson smoothing method is applied.

- **Local-Linear Smoothing** — Use the Local-Linear Smoothing property to specify whether to perform local-linear training. If the property is set to **Yes**, you can use the Local-Linear Options properties to modify the smoothing options.

- **Nadaraya-Watson Smoothing** — Use the Nadaraya-Watson Smoothing property to specify whether to perform Nadaraya-Watson training. If the property is set to **Yes**, you can use the Nadaraya-Watson Options properties to modify the smoothing options.

SOM/Kohonen Node Train Properties: Local-Linear Options

- **Convergence Criterion** — Use the Convergence Criterion property to specify the convergence criterion for the local-linear smoothing method. The Convergence Criterion property is a real number greater than or equal to 0, and the default value is 1.0E-4 (0.0001).

- **Max Iterations** — Use the Max Iterations property to specify the maximum number of iterations for the local-linear smoothing method. The Max Iterations property is an integer greater than or equal to 1, and the default value is 10.

The **Use Defaults** property must be set to **No** before you can change either of the properties in the Local-Linear Options.

SOM/Kohonen Node Train Properties: Nadaraya-Watson Options

- **Convergence Criterion** — Use the Convergence Criterion property to specify the convergence criterion for the Nadaraya-Watson smoothing method. The Convergence Criterion property is a real number greater than or equal to 0, and the default value is 1.0E-4 (0.0001).
- **Max Iterations** — Use the Max Iterations property to specify the maximum number of iterations for the Nadaraya-Watson smoothing method. The Max Iterations property is an integer greater than or equal to 1, and the default value is 10.

The **Use Defaults** property must be set to **No** before you can change either of the properties in the Nadaraya-Watson Options.

SOM/Kohonen Node Train Properties: Kohonen VQ

- **Maximum Number of Clusters** — If the Method property is set to **Kohonen VQ**, use the Maximum Number of Clusters property to specify the number of clusters that are created when you run the node. The Maximum Number of Clusters is an integer greater than or equal to 1. The default value is 10.

SOM/Kohonen Node Train Properties: Kohonen

- **Batch Training** — Use the Batch Training property to specify whether to perform batch training when the Method property is set to Kohonen SOM.
- **Use Defaults** — Use this property to specify whether to use the default Kohonen training settings.
- **Kohonen Options** — Select the button to open a properties table that allows you to customize the Kohonen Options settings. The Method property must be set to Kohonen SOM or Kohonen VQ and the Use Defaults property must be set to No before you can open the Kohonen Options properties table.
  - **Learning Rate** — Use the Learning Rate property to specify the learning rate for Kohonen training. The default value is 0.9.
  - **Initial Rate** — Use the Initial Rate property to specify the initial learning rate for Kohonen training. The default value is 0.9. Valid values are real numbers between 0 and 1.
  - **Final Rate** — Use the Final Rate property to specify the final learning rate for Kohonen training. The default value is 0.2. Valid values are real numbers between 0 and 1.
  - **Number of Steps** — Use the Number of Steps property to specify the step number at which the learning rate reaches the final learning rate. The default value is 1,000.
  - **Convergence Criterion** — Use the Convergence Criterion property to specify the value of the convergence criterion. The default value is 0.0001.
  - **Max Iterations** — Use the Max Iterations property to specify the maximum number of training iterations that you want to perform. An iteration is the processing that is performed on the entire data set. The default value is 100.
  - **Max Steps** — Use the Max Steps property to specify the maximum number of steps to perform during Kohonen training. A step is the processing that is performed on a single observation. The default value is 1,200.

SOM/Kohonen Node Train Properties: Neighborhood Options

If you selected Batch SOM or Kohonen SOM as the Method property, you can use the following properties to specify Neighborhood Options:

- **Use Defaults** — Use this property to specify whether to use the default values of the following neighborhood options.
- **Neighborhood Options** — Select the button to open a properties table that allows you to customize the Neighborhood Options settings. Note that the Method property must be set to Batch SOM or Kohonen SOM and the Use Defaults property must be set to No before you can open the Neighborhood Options properties table.

  The following properties are available in the Neighborhood Options properties table:

  - **Size** — Use the Size property to specify the neighborhood size. The neighborhood size must be greater than or equal to 0. The default value is Max(5, Max(Rows, Columns)/2).

  - Let Row(j) be the row number of the jth cluster, Col(j) be the column number of the jth cluster, Size be the neighborhood size, k be the kernel shape, and p be the kernel metric of infinity when the metric is max. The kernel function K(j,n) is

    $$K(j,n) = \left\{ 1 - \frac{\left[\left| Row(j) - Row(n) \right|^{p} + \left| Col(j) - Col(n) \right|^{p} \right]^{2/p}}{size^{2}} \right\}^{2k}$$

    Note that for a uniform kernel, K(j,n)=1 when the distance between the two seeds equals the neighborhood size, but for other kernels K(j,n)=0 when the distance between the two seeds equals the size.

  - **Kernel Shape** — Valid values are 0 (uniform), 1 (Epanechnikov), 2 (biweight) and 3 (triweight). The default value is 1.
  - **Kernel Metric** — Valid values are 0 (max), 1 (cityblock), and 2 (Euclidean). The default value is 0.
  - **Initial Size** — Use the Initial Size property to specify the initial size for the neighborhood. The default value is 5.
  - **Final Size** — Use the Final Size property to specify the final size for the neighborhood. The default value is 0.
  - **Number to Reset** — Use the Number to Reset property to specify the number of steps after which neighborhood size and kernel are reset. The default value is 100.
  - **Number of Steps** — Use the Number of Steps property to specify the step number at which the neighborhood size reaches the value of the Final Size property for Kohonen SOM training. The default value is 1,000.
  - **Iteration Number** — Use the Iteration Number property to specify the iteration number at which the neighborhood size reaches the value of the Final Size property for Batch SOM training. The default value is 3.

SOM/Kohonen Node Train Properties: Encoding of Class Variables

To incorporate the class variables into the analysis, use the following properties to specify how the SOM/Kohonen node handles the class variables:

- **Ordinal Encoding** — Use the Ordinal Encoding property to specify the encoding method that you want to use on the ordinal class variables in your data. Each encoding method derives one or more quantitative variables to be used for computing distances. The quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables computed for an encoding is called the *dimension*. The available ordinal encoding methods are Rank, Index, Bathtub, and Thermometer. The default ordinal encoding method is Rank.

  To illustrate the choices below, assume the following data set:

  ```
  DATA;
  INPUT x $ @@;
  CARDS;
  d c b a d c b d c d
  ```

  Each matrix below represents the ordinal class variable encoding method that is applied to the example data set.

  - **Rank Encoding** — (Default Setting) The dimension is one.

    ```
    LEVEL=ORDINAL(RANK)
        X   T_X
        --------
        a   0.05
        b   0.20
        c   0.45
        d   0.80
    ```

  - **Index Encoding** — The dimension is one.

    ```
    LEVEL=ORDINAL(INDEX)
        X       T_X
        ------------
        a         0
        b   0.333333
        c   0.666667
        d         1
    ```

  - **Thermometer Encoding** — The dimension is the number of categories minus one.

    ```
    LEVEL=ORDINAL(THERMOMETER)
        X  Xa  Xb  Xc
        -------------
        a   0   0   0
        b   1   0   0
        c   1   1   0
        d   1   1   1
    ```

  - **Bathtub Encoding** — The dimension is the number of categories minus one.

    ```
    LEVEL=ORDINAL(BATHTUB)
        X       Xa       Xb       Xc
        ----------------------------
        a -0.63246 -0.63246 -0.63246
        b  0.63246 -0.63246 -0.63246
        c  0.63246  0.63246 -0.63246
        d  0.63246  0.63246  0.63246
    ```

- **Nominal Encoding** — Use the Nominal Variable Encoding property to specify the encoding that you want to use on nominal variables. The encoding method specifies how to compute one or more quantitative variables to be used for computing distances. The quantitative variables are similar to the dummy variables that are used in linear models. The number of quantitative variables that are computed for an encoding is called the *dimension*. The available nominal encoding methods are GLM, Deviation, and Reference. GLM is the default nominal class variable encoding method.

  To illustrate the choices below, assume the following data set:

  ```
  DATA;
  INPUT x $ @@;
  CARDS;
  d c b a d c b d c d
  ```

  Each matrix below represents the nominal class variable encoding method that was applied to the example data set.

  - **GLM Encoding** — The dimension is the number of categories.

    ```
    LEVEL=NOMINAL(GLM)
        X Xa Xb Xc Xd
        -------------
        a    1  0  0  0
        b    0  1  0  0
        c    0  0  1  0
        d    0  0  0  1
    ```

  - **Deviation Encoding** — The dimension is the number of categories minus one.

    ```
    LEVEL=NOMINAL(DEVIATION)
        X  Xa  Xb  Xc
        --------------
        a   1   0   0
        b   0   1   0
        c   0   0   1
        d  -1  -1  -1
    ```

  - **Reference Encoding** — The dimension is the number of categories minus one.

    ```
    LEVEL=NOMINAL(REFERENCE)
        X Xa Xb Xc
        -----------
        a  1  0  0
        b  0  1  0
        c  1  1  1
        d  0  0  0
    ```

SOM/Kohonen Node Train Properties: Missing Values

Use the following properties to specify how to handle missing values during cluster initialization and scoring:

- **Interval Variables** — Use the Interval Variables property to specify how to handle missing values for interval variables during cluster initialization. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.

The following methods are available:

- **Default** — If the Scoring Imputation Method property is set to **None**, and the Interval Variables property is set to **Default**, then the Interval Variables property setting is not used. That is, observations that have missing values are omitted during cluster initialization; and all observations are scored, but distances are computed using only the non-missing values. If the Imputation Method property is set to **Seed of Nearest Cluster**, and the Interval Variables property is set to **Default**, then missing values are ignored during cluster initialization, and the seeds of the nearest cluster are used to replace missing values scoring.
- **Ignore** — Missing interval variable values are ignored during cluster initialization. For observations that have ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the non-missing variables divided by the total variance of all non-rejected variables.
- **Mean** — Missing interval variable values are replaced by the variable mean during cluster initialization.
- **Midrange** — Missing interval variable values are replaced by the variable midrange during cluster initialization.
- **Omit** — Any observation that has any missing values for any of the listed interval variables is omitted from the cluster analysis.

- **Nominal Variables** — Use the Nominal Variables property to specify how to handle missing values for nominal variables during cluster initialization. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.

  The following methods are available:

  - **Default** — If the Scoring Imputation Method property of the SOM/Kohonen node is set to **None**, and if the Nominal Variables property setting is **Default**, then the Nominal Variables property setting is not used. That is, observations that have missing values are omitted during cluster initialization; and all observations are scored, but distances are computed using only the non-missing values. If the Scoring Imputation Method property is set to **Seed of Nearest Cluster**, and the Nominal Variables setting is **Default**, then observations that have missing values for nominal variable are ignored during cluster training, and missing values are replaced by the nearest cluster seed during cluster scoring.
  - **Ignore** — Missing nominal variable values are ignored during cluster initialization. For observations that have ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the non-missing variables divided by the total variance of all non-rejected variables.
  - **Mean** — Missing nominal variable values are replaced by the variable mean during cluster initialization.
  - **Mode** — Missing nominal variable values are replaced by the variable mode during cluster initialization. If there is no unique mode, the mean of all modes is used.
  - **Omit** — Any observation that has any missing values for any of the listed nominal variables is omitted from the cluster analysis.

- **Ordinal Variables** — Use the Ordinal Variables property to specify how to handle observations that contain missing values for ordinal variables during cluster initialization. Observations that have missing values cannot be used as cluster seeds. Observations that contain all missing values are excluded from the analysis.

The following methods are available:

  - **Default** — If the Scoring Imputation Method property is set to **None**, and the Ordinal Variables property is set to **Default**, the Ordinal Variables property setting is not used. That is, observations that have missing values are omitted during cluster initialization; and all observations are scored, but distances are computed using only the non-missing values. If the Scoring Imputation Method property is set to **Seed of Nearest Cluster**, and the Ordinal Variables property is set to **Default**, then missing ordinal variables are ignored during cluster initialization, and missing variables are replaced using the nearest cluster seed during scoring.
  - **Ignore** — Missing ordinal variable values are ignored during cluster initialization. For observations with ignored missing values, distance from the cluster seed is computed as the square root of the summed variances ratio. The summed variances ratio is the total variance of the non-missing variables divided by the total variance of all non-rejected variables.
  - **Median** — Missing ordinal variable values are replaced by the variable median during cluster initialization.
  - **Mode** — Missing ordinal variable values are replaced by the variable mode during cluster initialization. If there is no unique mode, the mean of all modes is used.
  - **Omit** — Any observation that has any missing values for any of the listed ordinal variables is omitted from the cluster analysis. Statistics from the DMDB catalog are not adjusted for omitted observations.

- **Scoring Imputation Method** — Use the Scoring Imputation Method property to specify the method that you want to use to impute missing values during scoring.

  You can choose from the following methods:

  - **None** — no imputations are performed. None is the default setting.
  - **Seed of Nearest Cluster** — missing values are replaced with the value of the seed of the nearest cluster.

SOM/Kohonen Node Score Properties

The following score properties are associated with the SOM/Kohonen node:

- **Segment Role** — Use the Segment Role property to specify the model role that is assigned to the cluster variable (segment identifier). The available model roles are

  - **Segment**
  - **ID**
  - **Input**
  - **Target**

  The default model role is **Segment**. Note that the cluster variable retains the selected model role as it is passed to subsequent nodes in the process flow diagram.

- **Exported Variables** — Use the Exported Variables property to indicate which of the variables that the SOM/Kohonen node creates at run time are exported to successor nodes in the process flow diagram.

  - **All** — All created variables are exported to successor nodes. **All** is the default setting.
  - **Dimensions** — Only created dimension variables are exported to successor nodes.
  - **Segment** — Only created segment variables are exported to successor nodes.
- **Hide Original Variables** — Use the Hide Original Variables property to specify whether you want to display the original transformed variables in the node output. The default setting for the Hide Original Variables property is **Yes**.

SOM/Kohonen Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

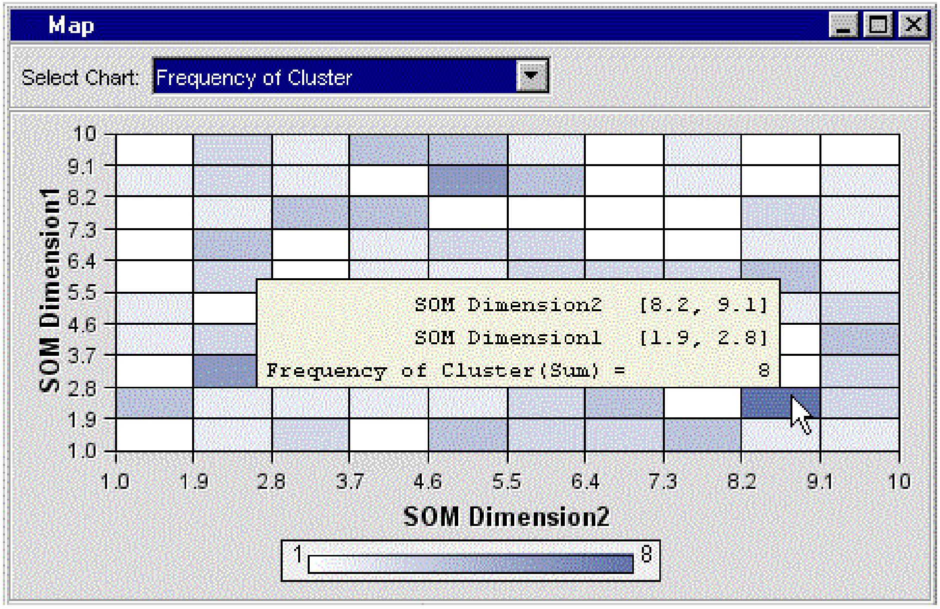
SOM/Kohonen Node Results

Open the Results window by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the SOM/Kohonen node properties configuration when the node was last run.
  - **Run Status** — displays the status of the SOM/Kohonen node run. Information about the run start time, run duration, run ID and completion status are displayed in this window.
  - **Variables** — displays a table of the variables submitted to the SOM/Kohonen node. The variables table displays the Name, Use, Report, Role and Level settings for each variable.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — the SAS log of the SOM / Kohonen node run.
  - **Output** — the SAS output of the SOM / Kohonen node run. The SOM / Kohonen node SAS output contains a variable summary.

  - **Flow Code** — the SAS code used to produce the output that the SOM / Kohonen node passes on to the next node in the process flow diagram.
  - **Train Graphs** — is dimmed and unavailable in the SOM/Kohonen node
  - **Report Graphs** — is dimmed and unavailable in the SOM/Kohonen node
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the SOM / Kohonen node. The SOM / Kohonen node SAS code can be edited and used in external applications outside of the Enterprise Miner environment.
  - **PMML Code** — the SOM/Kohonen node does not generate PMML code.
- **Model**
  - **Mean Statistics** — The Mean Statistics table displays following statistics for each cluster.
    - Clustering Criterion
    - Maximum Relative Change in Cluster Seeds
    - Improvement in Clustering Criterion
    - SOM Segment ID
    - Frequency of Cluster — the number of observations in each cluster.
    - Root-Mean-Square Standard Deviation — the root mean square error across variables of the cluster standard deviations, which is equal to the root mean square distance between observations in the cluster.
    - Maximum Distance from Cluster Seed — the maximum distance from the cluster seed to any observation in the cluster.
    - Nearest Cluster — the number of the cluster that has a mean closest to the mean of the current cluster.
    - Distance to Nearest Cluster — the distance between the centroids (means) of the current cluster and the nearest other cluster.
    - SOM Dimension1
    - SOM Dimension2
    - SOM ID
    - Mean of each input variable.
  - **Map** — if you set the Method property to **Batch SOM** or **Kohonen SOM**, a topological mapping of the input space to the clusters is produced. If you set the Method property to **Kohonen VQ**, a graphical representation of key characteristics of the segments that are generated by the node is produced.



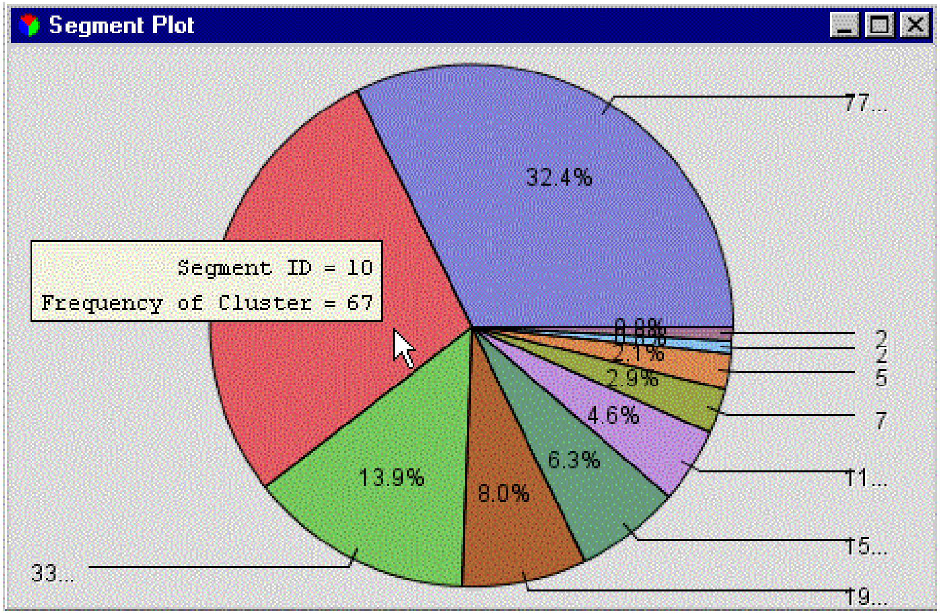
```
          SOM Dimension2   [8.2, 9.1]
          SOM Dimension1   [1.9, 2.8]
Frequency of Cluster(Sum) =           8
```

To examine a cluster in the map, place your mouse pointer over a cluster in the topological map. A text box will display the SOM dimensions and Frequency of Cluster (the number of observations in a cluster).

When you select a cluster, the corresponding row in the Mean Statistic table is highlighted and displays statistics for that cluster. To select multiple clusters, hold your right mouse button down and drag to create a rectangle over the clusters in the map.

By default, the SOM/Kohonen node uses the cluster frequency to color the cluster. Use the Select Chart combo box to select a statistic to color the clusters.

- **Segment Plot** — if you set the Method property to **Kohonen VQ**, the Segment Plot provides a graphical representation of key characteristics of the segments that are generated from the **Kohonen VQ** method.



```
        Segment ID = 10
Frequency of Cluster = 67
```

To examine a segment in the map, place your pointer over a slice of the pie. A text box will display the segment ID.

When you select a slice, the corresponding row in the Mean Statistic table is highlighted and displays statistics for that cluster. To select multiple clusters, press CTRL and click the clusters that you want.

By default, the size of each slice in the two-dimensional pie chart is proportional to the cluster frequency. To use a different variable to create the pie chart, right-click in the map and select **Data Options**, and then assign the **Response** role to that variable.

- **Analysis Statistics** — The Analysis Statistics table displays the following statistics:
  - Type of Observation
  - SOM Segment ID
  - SOM Dimension 1
  - SOM Dimension 2
  - SOM ID
  - Statistic Applying over All Variables.
  - Statistics for each input variable

- **Table** — displays a table of the data that is associated with the graph that you have open. Data table column headings display variable labels by default. To display the variable names used in SAS code as column headings, right-click in any table cell and select **Show Variable Names**. To view labels in the column headings again, right-click in any table cell and select **Show Variable Labels**.
- **Plot** — enables you to create or modify a graph of the data in the table that you have open.

References

Anderberg, M.R 1973. *Cluster Analysis for Applications*. New York, NY: Academic Press, Inc.

Balakrishnan, P.V., Cooper, M.C., Jacob, V.S., and Lewis, P.A “A Study of the Classification Capabilities of Neural Networks Using Unsupervised Learning: A Comparison with k-Means Clustering..” 1994. *Psychometrika* 59: 509–525.

Forgy, E.W. “Cluster Analysis of Multivariate Data: Efficiency versus Interpretability.” 1965. *Abstract in Biometrics* 21: 768.

Gersho, A. and Gray, R.M 1992. *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers.

Hartigan, J.A 1975. *Clustering Algorithms*. New York, NY: Wiley.

Hartigan, J.A. and Wong M.A “Algorithm AS136: A k-Means Clustering Algorithm.” 1979. *Applied Statistics* 28: 100–108.

Jastie, T. and Stuetzle, W. “Principal Curves.” 1989. *Journal of the American Statistical Association* 84: 502–516.

Hecht-Nielsen, R 1990. *Neurocomputing*. Reading, MA: Addison-Wesley.

Ismail, M.A. and Kamel, M.S. “Multidimensional Data Clustering Utilizing Hybrid Search Strategies.” 1989. *Pattern Recognition* 22: 75–89.

Kohonen, T 1984. *Self-Organization and Associative Memory*. Berlin, Germany: Springer-Verlag.

Kohonen, T. “Learning Vector Quantization.” 1988. *Neural Networks* 1 (suppl 1): 303.

Kohonen, T. 1995. *Self-Organizing Maps*. Berlin, Germany: Spring-Verlag.

Kosko, B 1992. *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Linde, Y., Buzo, A., and Gray, R “An Algorithm for Vector Quantizer Design.” 1980. *IEEE Transactions on Comunications* 28: 84–95.

Lloyd, S “Least Squares Quantization in PCM.” 1982. *IEEE Transaction on Information Theory* 28: 129–137.

MacQueen, J.B “Some Methods for Classification and Analysis of Multivariate Observations.” 1967. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1: 281–297.

Max, J. “Quantizing for Minimum Distortion.” 1960. *IEEE Transaction on Information Theory* 6: 7–12.

Mulier, F. and Cherkassky, V “Self-Organization as an Iterative Kernel Smoothing Process.” 1995. *Neural Computation* 7: 1165–1177.

Ripley, B.D 1996. *Pattern Recognition and Neural Networks*. Cambridge, MA: Cambridge University Press.

Ritter, H., Martinetz, T., and Schulten, K 1992. *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, MA: Addison-Wesley.

SAS Institute 2009. *SAS/STAT User's Guide*. Cary, NC: SAS Institute.

Symons, M.J “Clustering Criteria and Multivariate Normal Mixtures.” 1981. *Biometrics* 37: 35–43.

Tibshirani, R. “Principal Curves Revisited.” 1992. *Statistics and Computing* 2: 183–190.

Wand, M.P. and Jones, M.C 1995. *Kernel Smoothing*. London, England: Chapman and Hall.

Zador, P.L “Asymptotic Quantization Error of Continuous Signals and the Quantization Dimension.” 1982. *IEEE Transactions on Information Theory* 28: 139–149.

Zeger, K., Vaisey, J., and Gersho, A “Globally Optimal Vector Qunatizer Design by Stochastic Relaxation.” 1992. *IEEE Transactions on Signal Processing* 40: 310–332.

Chapter 38
StatExplore Node

StatExplore Node

Overview of the StatExplore Node

The StatExplore node is a multipurpose tool that you use to examine variable distributions and statistics in your data sets. The StatExplore node generates summarization statistics.

You can use the StatExplore node to do the following:

- Select variables for analysis, for profiling clusters, and for predictive models.
- Compute standard univariate distribution statistics.
- Compute standard bivariate statistics by class target and class segment.
- Compute correlation statistics for interval variables by interval input and target.

In data-mining applications, the number of variables can be quite large due to data complexity or lack of prior information about the variables. A challenge in data-mining graphics is to reduce the number of displayed attributes so that the plot is readable, meaningful, and does not consume disproportionate amounts of client workstation memory and network bandwidth. As a result, the StatExplore node works to select only the most important variables for automated display.

You can combine the functions of the StatExplore node with other Enterprise Miner node functions to perform data-mining tasks.

Some tasks that you can perform by combining the StatExplore node with other Enterprise Miner nodes are as follows:

- Use the StatExplore node with the Metadata node to reject variables.
- Use the StatExplore node with the Transform Variables node to to suggest transformations.
- Use the StatExplore node with the Regression node to create interaction terms.

StatExplore Node Properties

StatExplore Node General Properties

The following general properties are associated with the StatExplore node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first StatExplore node added to a diagram has a Node ID of Stat. The second StatExplore node added to a diagram has a Node ID of Stat2.
- **Imported Data** — accesses the Imported Data — StatExplore window. The Imported Data - StatExplore window contains a lists of the ports that provide data sources to the StatExplore. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — StatExplore window. The Exported Data - StatExplore window contains a list of the output data ports that the StatExplore node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

StatExplore Node Train Properties

The following train properties are associated with the StatExplore node:

- **Variables** — Click the ... icon to open a variables table showing the data that is imported into the StatExplore node. You can click cells in the Use column if you want to change the status of an imported variable to **No** or **Yes**. The Default setting in the Variables window for the StatExplore node simply preserves the variable status that is passed to StatExplore by the predecessor node. You can click on any column heading to toggle between ascending and descending table sorts.

  You can select one or more variable rows in the table and click the Explore button to view additional information about the variables, such as Sample Properties and Details, tables of variable values, and plots of the variable distribution. More detailed information on exploring variables is in the section on the StatExplore node's Variables window.

StatExplore Node Train Properties: Data

- **Number of Observations** — Specifies the number of observations to use for generating various reports. When **ALL** is selected, the entire data set is used. When a number n is specified, the first n observations are used.
- **Validation** — Specify **Yes** if you want summary and association statistics to be generated for the validation data set.
- **Test** — Specify **Yes** if you want summary and association statistics to be generated for the test data set.

StatExplore Node Train Properties: Standard Reports

- **Interval Distributions** — Specify **Yes** if you want to generate distribution reports for interval variables. These reports include overall summary statistics of all selected interval input variables and across all levels of class target variables. They also include the Interval Variable plots that display the scaled mean deviations across all levels of the class target variables.
- **Class Distributions** — Specify **Yes** if you want to generate distribution reports for class variables. These reports include summary statistics of all segment, target, and selected class input variables, plus summary statistics across all levels of class target variables. The reports include the Class Variables plots that display the distribution selected class variables for all levels of the class target variables. The reports also include the Chi-Square plot that identifies the selected variables that are most highly associated with the class target variables.
- **Level Summary** — Specify **Yes** if you want to generate the level summary report. The level summary report displays the number of distinct levels for ID, segment, and class target variables.
- **Use Segment Variables** — Set the Use Segment Variables property to **Yes** if you want to generate summary and association statistics for all segment variables in the data set that you submitted to the StatExplore node. The default setting for the Segment Variables property is **No**.
- **Cross-Tabulation** — Click the ... icon to open a dialog box that enables you to specify the cross-tabulation variables.

StatExplore Node Train Properties: Variable Selection

- **Hide Rejected Variables** — Set the Hide Rejected Variables property of the StatExplore node to **No** if you want to analyze all the inputs in a data set. In its default setting of **Yes**, the Hide Variable property drops input variables that are not analyzed from the exported metadata, or it sets the variable roles to rejected. Hide Rejected Variables is especially useful when you use input data sources with very large numbers of input variables.

- **Number of Selected Variables** — Use the Number of Selected Variables property of the StatExplore node to specify a positive integer value for the maximum number of variables to select. When target variables are present, variables are selected using a worth measure computed from a decision tree. When there are no target variables, the number selected is based on maximum variability. Choose from discrete values of 50, 100, 200, 500, 1000, 2000, 4000, 6000, 8000, 10000. The default setting for the Maximum Variables property is 1000.

StatExplore Node Train Properties: Chi-Square Statistics

- **Chi-Square** — Set the Chi Square property to **No** if you want to suppress the generation of chi-square statistics for the data set that you submitted to the StatExplore Node. The default setting for the Chi-Square property is **Yes**.
- **Interval Variables** — set the Interval Variables property of the StatExplore node to **Yes** if you want to generate chi-square statistics for the interval variables by binning the variables. If you set the Interval Variables property to **Yes**, you must specify N, the number of bins with the Number of Bins property. The default setting for the Interval Variables property is **No**.
- **Number of Bins** — When you set the StatExplore node Interval Variables property to **Yes**, you must use the Number of Bins property to specify N, the number of bins that you want to generate for the interval variables in your data set. The StatExplore node uses the binned values to calculate chi-square statistics for the interval variables in your data set. Permissible values for the Number of Bins property are integers ranging from 2 to 10. The default value is 5.

StatExplore Node Train Properties: Correlation Statistics

- **Correlations** — Set the Correlations property to **No** if you want to suppress the generation of correlation statistics for the data set that you submitted to the StatExplore node. The default setting for the Correlations property is **Yes**.
- **Pearson Correlations** — Set the Pearson correlations property to **No** if you want to suppress the generation of Pearson correlation statistics for the data set that you submitted to the StatExplore node. The default setting for the Pearson property is **Yes**.
- **Spearman Correlations** — Set the Spearman property to **Yes** if you want to generate Spearman correlation statistics for the data set that you submitted to the StatExplore node. The default setting for the Spearman property is **No**.

StatExplore Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

StatExplore Node Results

After the StatExplore node successfully runs, you can open the Results — StatExplore window by right-clicking the node in the Diagram Workspace and selecting **Results** from the pop-up menu. The Results — StatExplore window contains an imputation summary table and a window displaying the node's output. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

StatExplore Node Results Window Menus

Select **View** from the main menu to view the following information in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the StatExplore node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the StatExplore node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — the code that Enterprise Miner used to train the node.
- **SAS Results**
  - **Log** — the SAS log of the StatExplore node run.
  - **Output** — the SAS output of the StatExplore node run. The StatExplore SAS output includes a variables summary, a variable levels summary, class variable summary statistics, distribution of class target and segment variables, interval variable summary statistics, class variable summary statistics by class target, interval variable summary statistics by class target, and chi-square statistics.
  - **Flow Code** — the SAS code used to produce the output that the StatExplore node passes on to the next node in the process-flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the StatExplore node does not generate PMML code.
- **Summary Statistics**
  - **Interval Variables** — displays a table of summary statistics for interval variables. The columns of the “Interval Variable Summary Statistics” on page 573 table are as follows:

  - Target
  - Target Value
  - Variable
  - Mean
  - Standard Deviation
  - Nonmissing
  - Missing
  - Min
  - Median
  - Max
  - Scaled Mean Deviation
  - Maximum Deviation
  - Level ID

- **Class Variables** — displays the "Class Variable Summary Statistics" on page 573 table. The columns of the table are as follows:
  - Target
  - Target Value
  - Variable
  - Type
  - Numeric Value
  - Frequency Count
  - Percent Within
  - Role
  - Percent
  - Plot
- **Cell Chi-Squares** — displays the "Chi-Squared Summary Statistics" on page 574 information for each segment. Segments comprise the rows of the table. The columns of the table are as follows:
  - Segment ID
  - Target
  - Input
  - Target Formatted Value
  - Input Formatted Value
  - Frequency Count
  - Target Numeric Value
  - Input Numeric Value
  - Chi Square
  - Log Chi Square

- **Plots** — The StatExplore node produces several results plots. The plots produced vary according to the relationship between the inputs and the targets and the presence of segment variables.
  - **Variable Worth Plot** — The Variable Worth plot ranks input variables according to their calculated worth. Hovering your cursor over a chart element makes a pop-up appear with importance ranking, calculated worth, and the variable name.
  - **Correlation Plot** — If no segment variables are present, and there are any interval target variables and the Correlations property in the StatExplore Properties Panel is set to Yes, then the scatter chart displays the Pearson and Spearman correlation statistics.

    If there are one or more segment variables present, and there are any interval target variables, and the Correlations property in the StatExplore node Properties Panel is set to Yes, then a heat map plot is used to display the correlation statistics across segments.
  - **Chi-Square Plot** — If there are no segment variables present, and there are class target variables, and the Chi-Square property in the StatExplore node Properties Panel is set to Yes, the Chi-Square Plot displays the measure of association (Cramer's V Statistic). If the Use Segment Variables property in the StatExplore node is set to yes, and there are segment variables present, and there are class target variables, and the Chi-Square property in the StatExplore node Properties Panel is set to Yes, the heat map plot displays the measure of association (Cramer's V statistic) across the segments.
  - **Class Variables Plot** — The Class Variable plot illustrates the distribution of values for discrete class levels.
  - **Interval Variables Plot** — The Interval Variable plot illustrates the distribution of values for interval variables.
- **Tables** — displays the data table for a graph that you select.
- **Plot** — opens the Graph Wizard for the table that you select.

StatExplore Results Plots

Overview

The format of the graphs for interval and class variables that the StatExplore node produces depends on the presence and type of target and segment variables.

For most of the plots that follow, variables for summarization and plots were screened by a variation measure appropriate for the variable types. Typically, summary statistics are stored in SAS tables for the 100 variables with the most variation, and lattice plots are generated for the top 20 variables.

Finally, in the user interface, both the tables and graphs are displayed. The tables are also printed to the output listing. The user may make new graphs that create their own views of the summarization using the data in the tables.

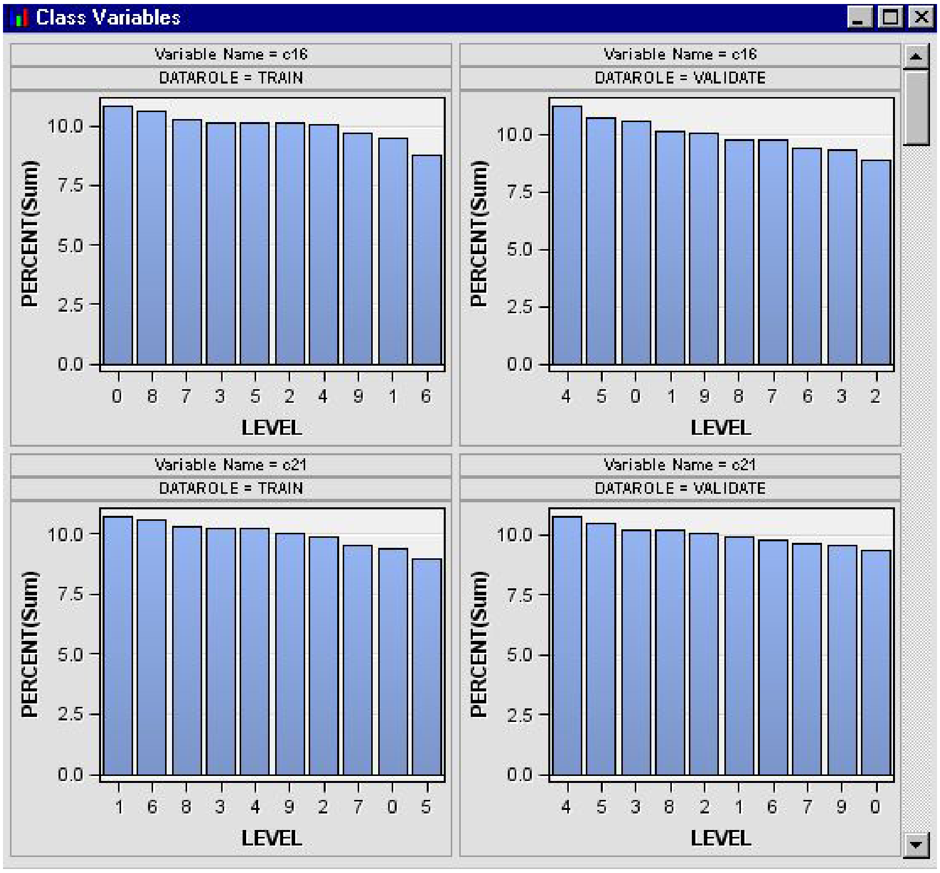
Some of the Variable Distribution Plots available are as follows:

- “Case 1: No Class Target and No Segment Variable” on page 560
- “Case 2: Class Target Present and No Segment Variable” on page 562
- “Case 3: No Class Target and One or More Segment Variables” on page 566

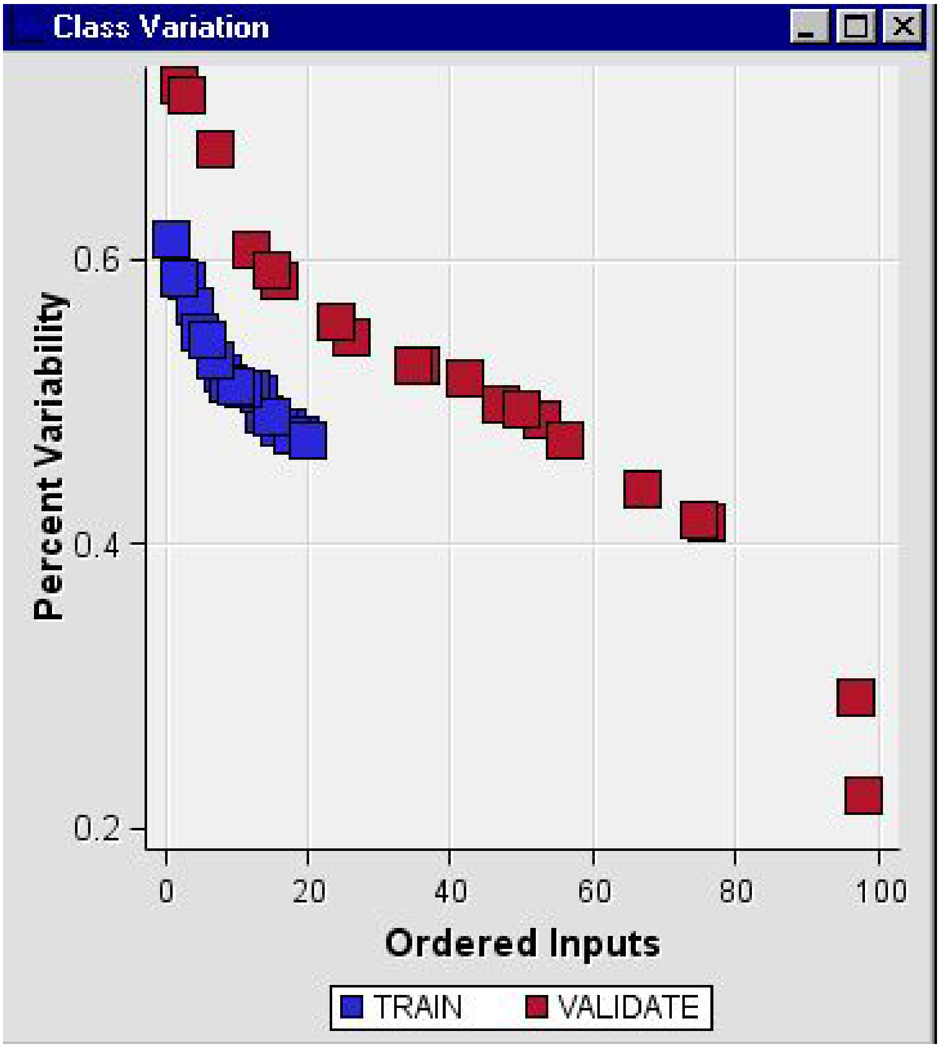
- "Case 4: Class Target and One or More Segment Variables" on page 568
- "Case 5: Interval Targets" on page 572

Case 1: No Class Target and No Segment Variable

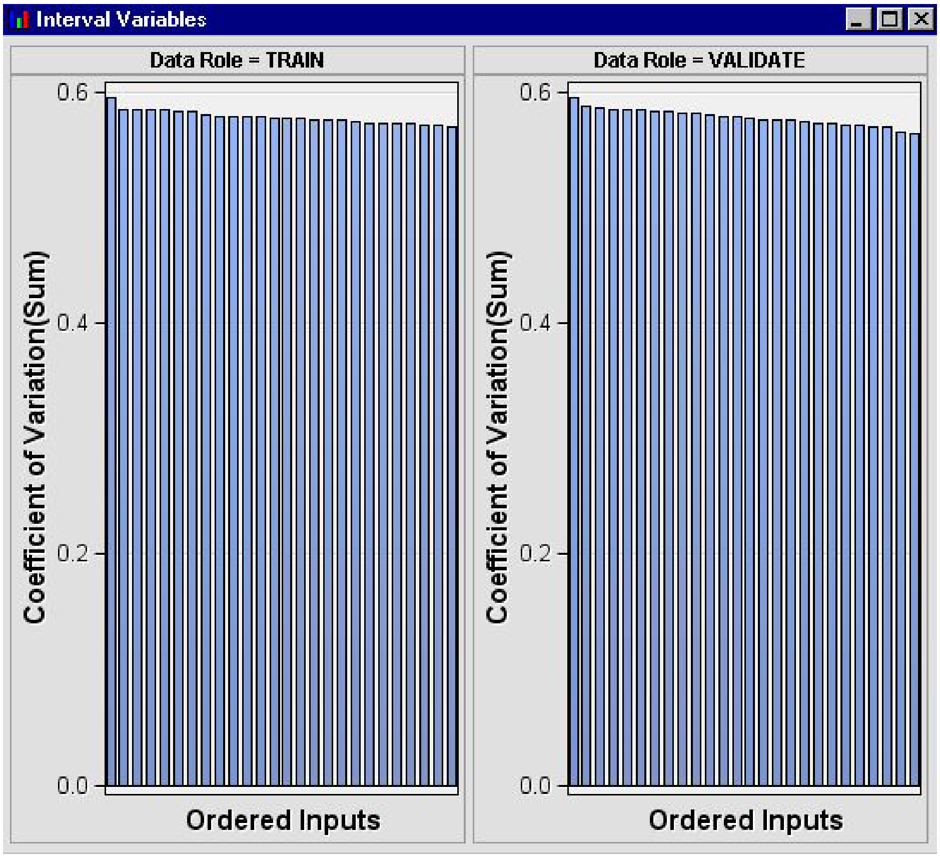
If the class data you submit to the StatExplore node contains no class target and no defined segment variable, the node produces a matrix of bar charts for the class variables, up to a maximum of 20 variables.



The variables are selected according to amount of variability in the bar charts. The criterion plot displays the computed criterion for each variable.

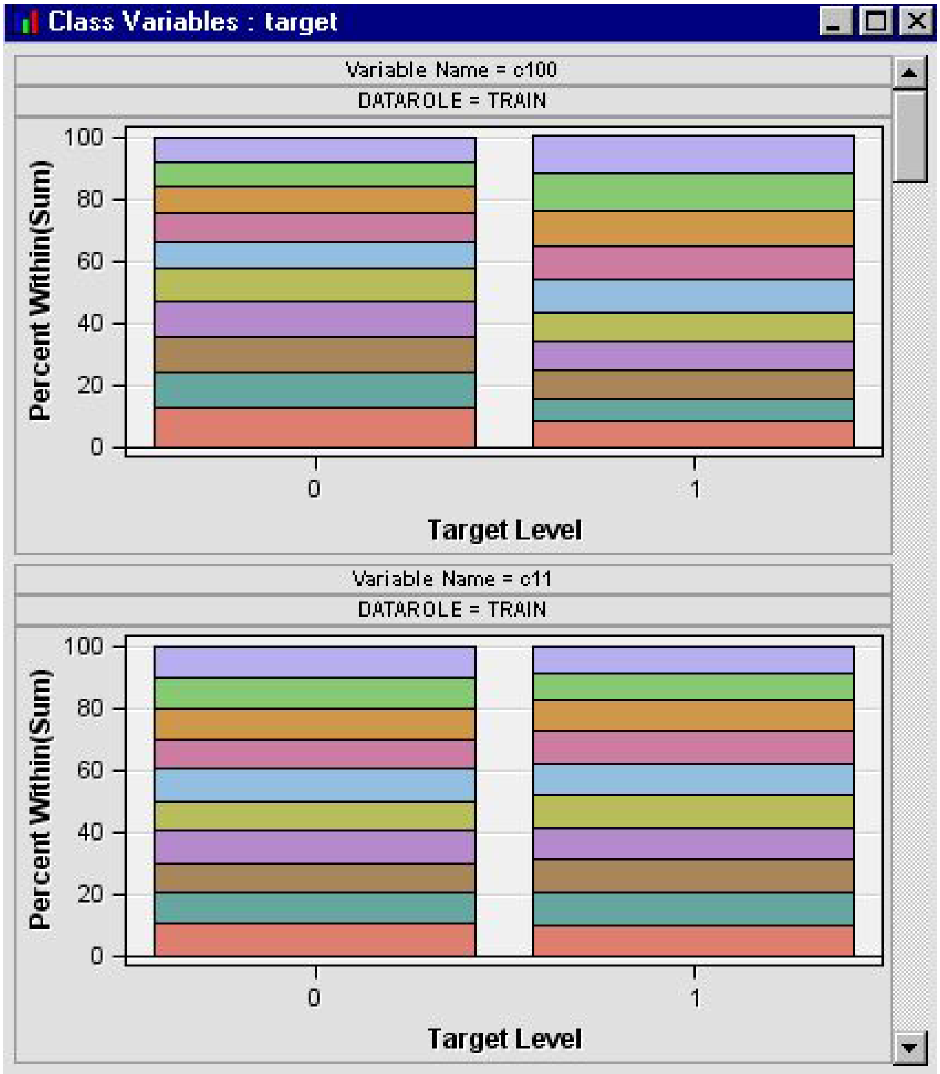


An ordered bar chart for interval input variables is also generated. The interval input variables are ordered by their coefficient of variation statistic.

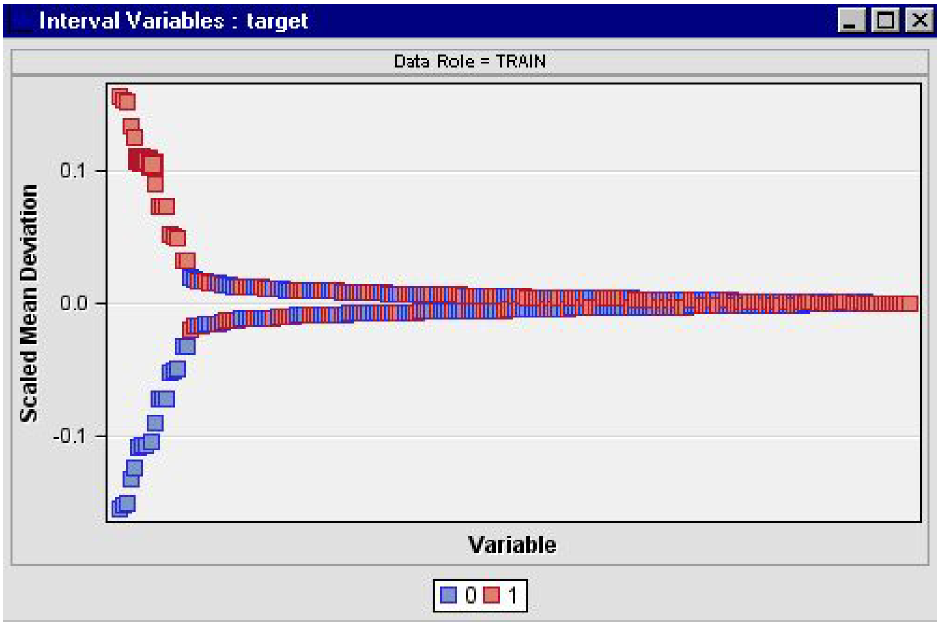


Case 2: Class Target Present and No Segment Variable

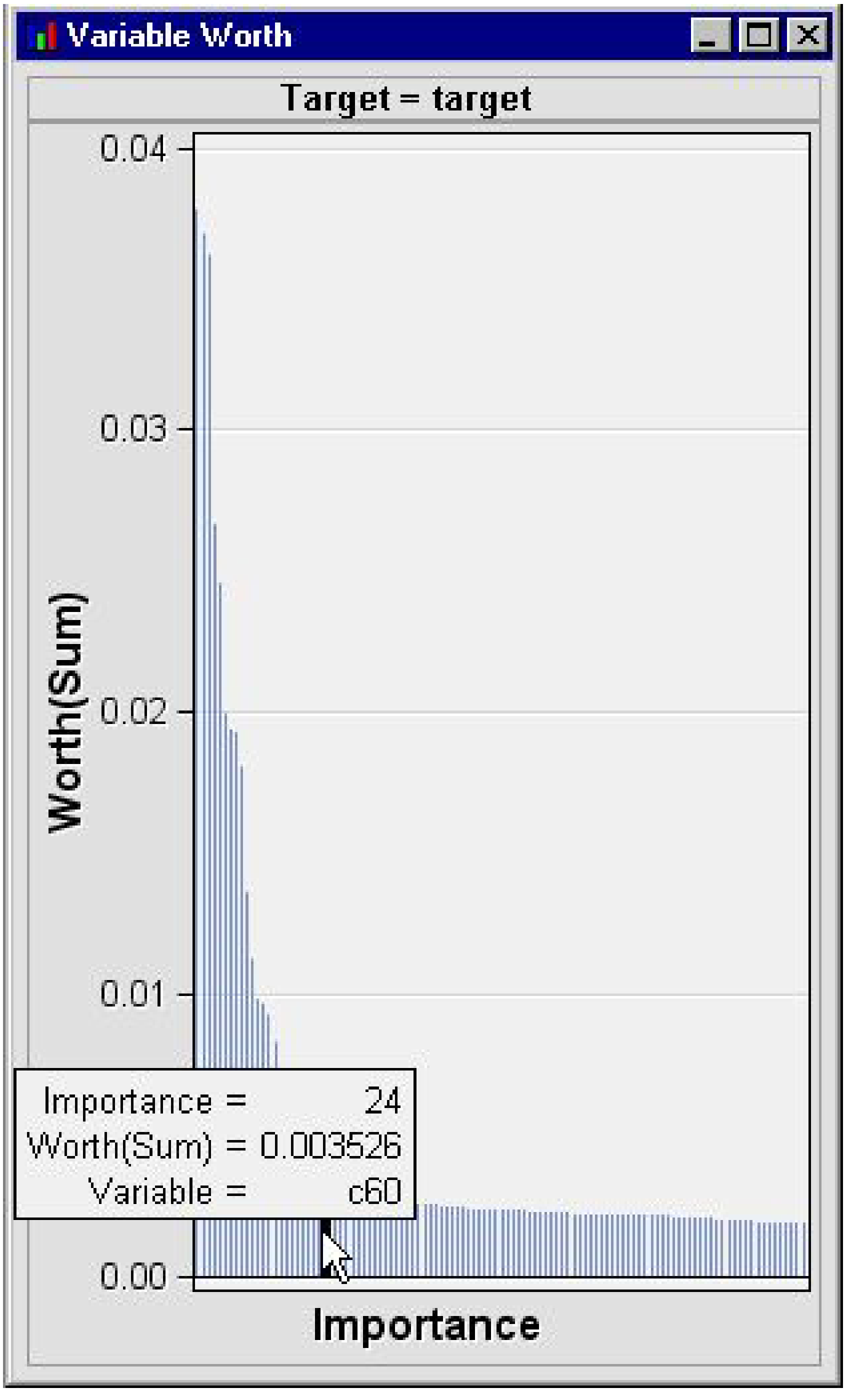
If the class data you submit to the StatExplore node contains a class target but has no defined segment variable, the node produces a lattice of bar charts that shows the distribution of class variables within each level of the target variable.



The StatExplore node creates an additional plot that shows the scale deviation between the overall mean of an interval input variable and its mean for each level of the target variable. The plot orders the variables from the variable with the most deviation to the variable with the least deviation. If there is more than one target variable, a separate set of plots is created for each target variable.

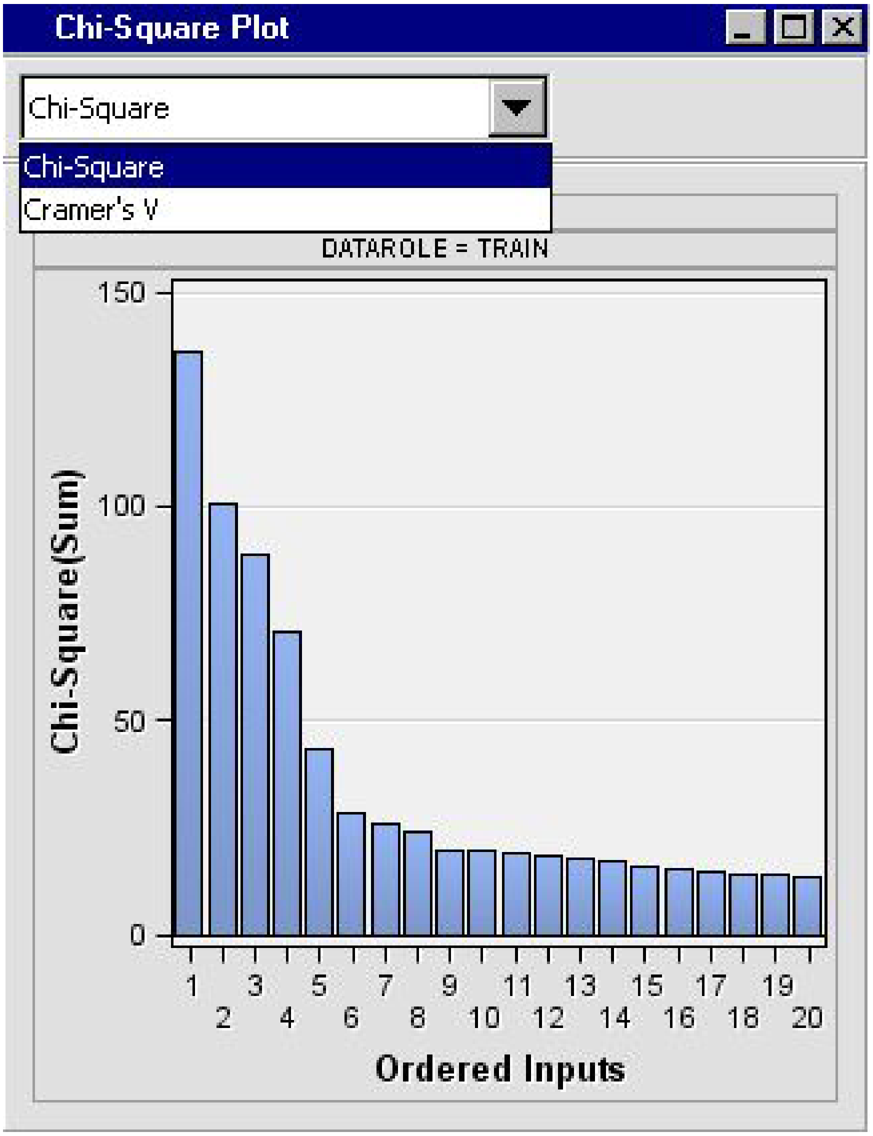


The StatExplore node also generates a Variable Worth plot that ranks input variables according to their calculated worth. See Variable Importance in the Help for the Decision Tree node for details on how a variable's worth (importance) is computed.



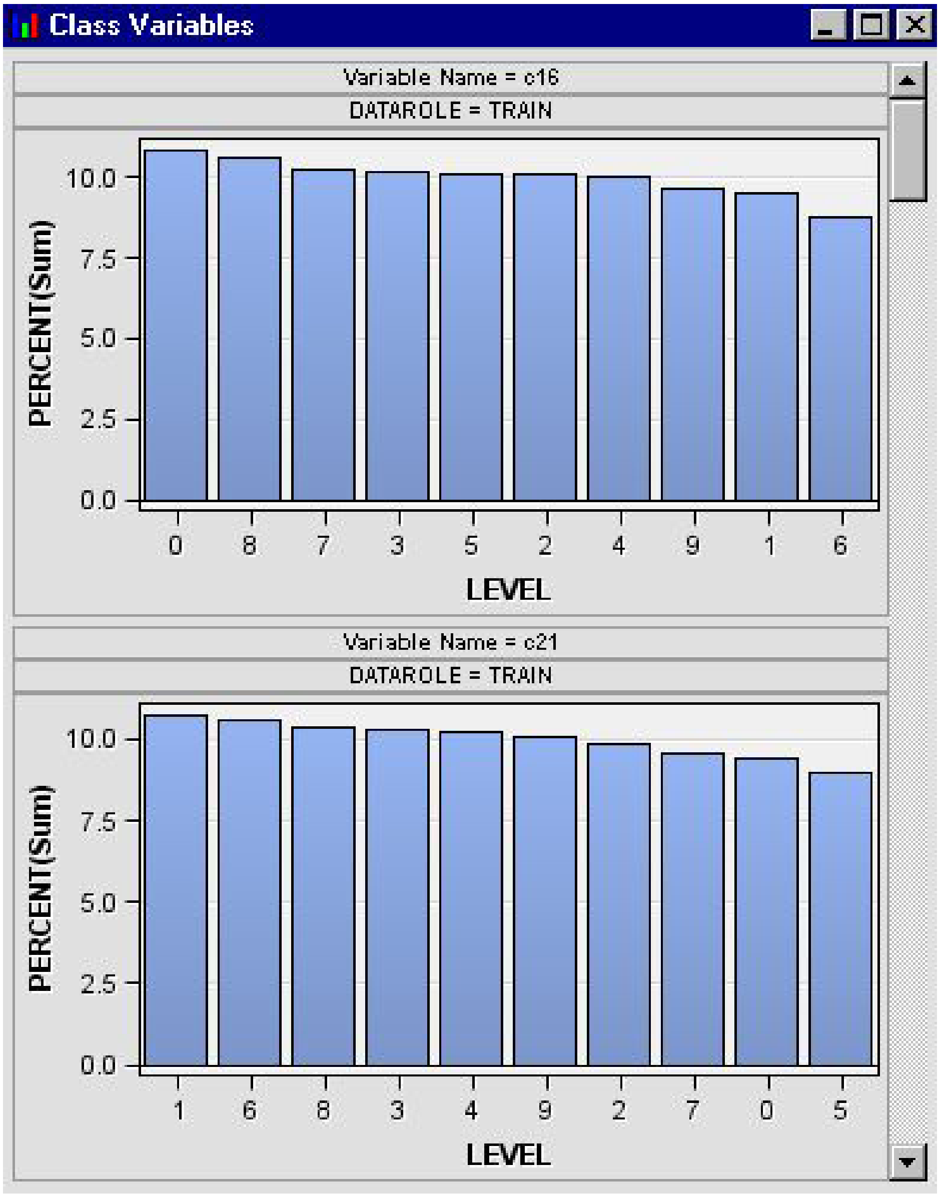
Importance = 24
Worth(Sum) = 0.003526
Variable = c60

If there are class target variables and the Chi-Square property in the StatExplore node Properties Panel is set to Yes, a combo plot of ordered inputs is also produced. Up to 20 variables are selected using a chi-square statistic or a Cramer's V staistic as the selection criterion. If more than one class target variable is specified, a Results Package is created for each class target variable.

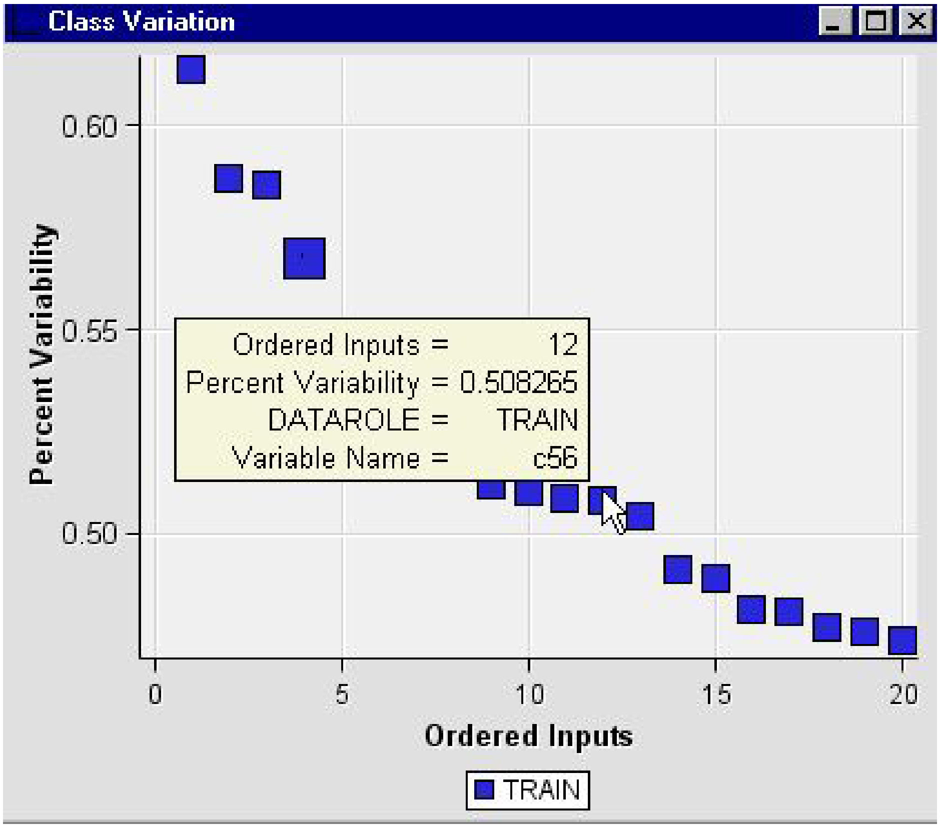


Case 3: No Class Target and One or More Segment Variables

If the class data you submit to the StatExplore node contains no class target but has one or more defined segment variables, the node produces a lattice of bar charts that shows the distribution of class variable values within each segment. Up to 20 variables are selected using a Cramer's V criterion.

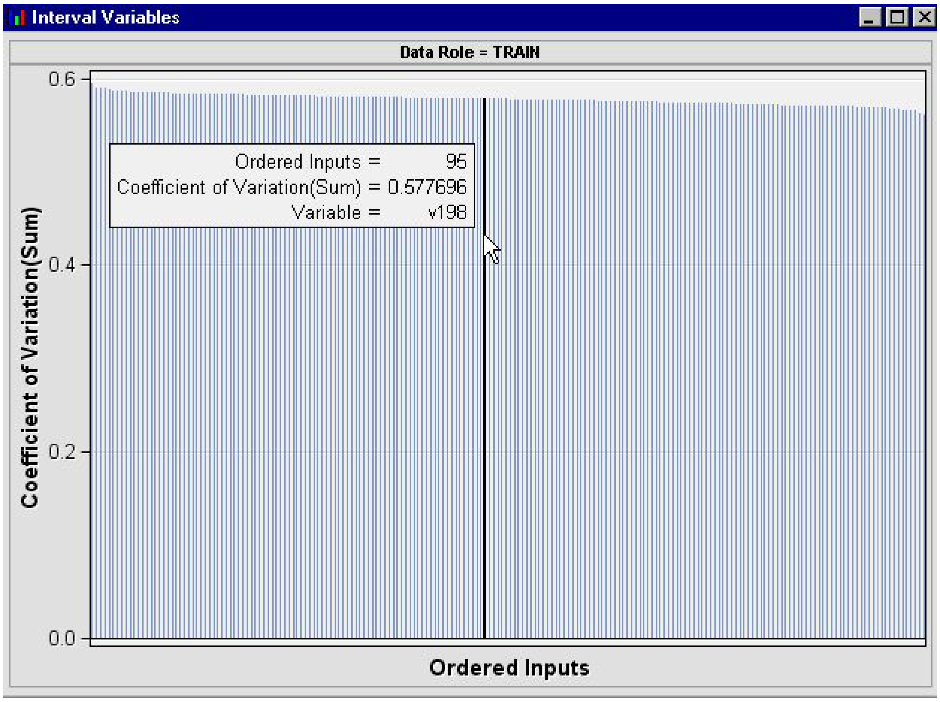


The variables are selected according to amount of variability in the bar charts. The Class Variation plot displays the computed criterion for each variable.



Ordered Inputs = 12
Percent Variability = 0.508265
DATAROLE = TRAIN
Variable Name = c56

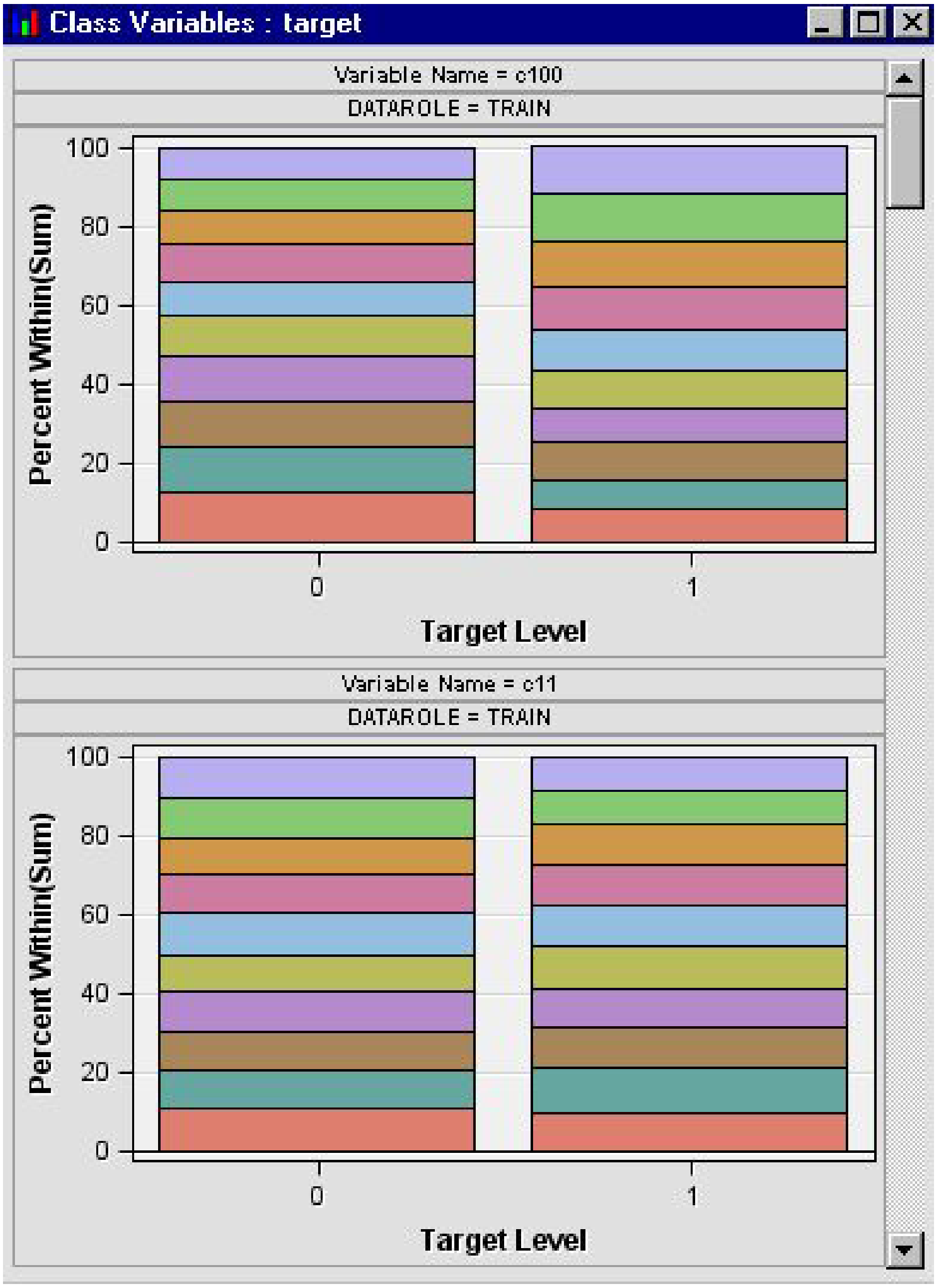
A plot of ordered interval inputs is also produced. The ordering criterion is the coefficient of variation statistic.



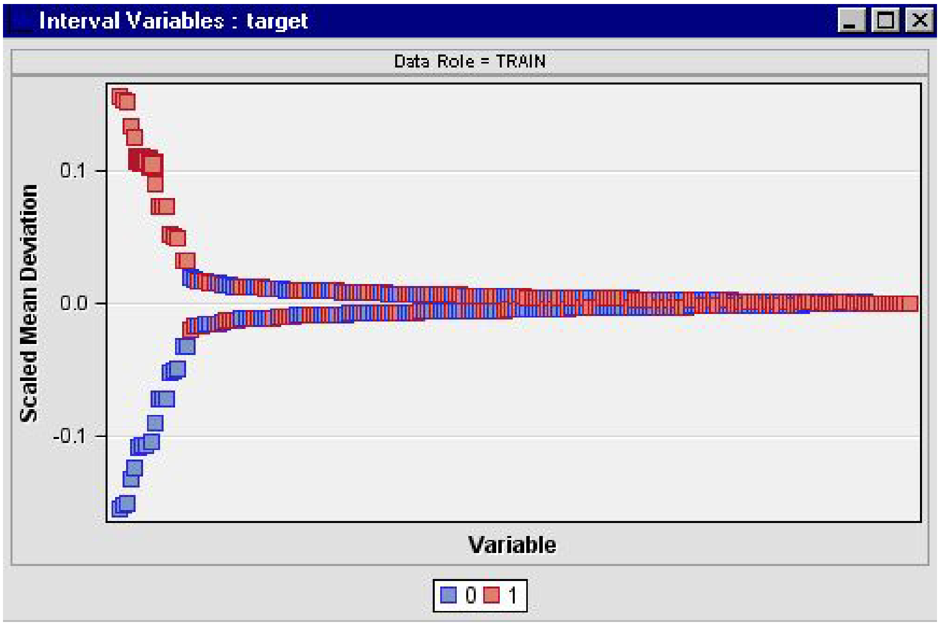
Ordered Inputs = 95
Coefficient of Variation(Sum) = 0.577696
Variable = v198

Case 4: Class Target and One or More Segment Variables

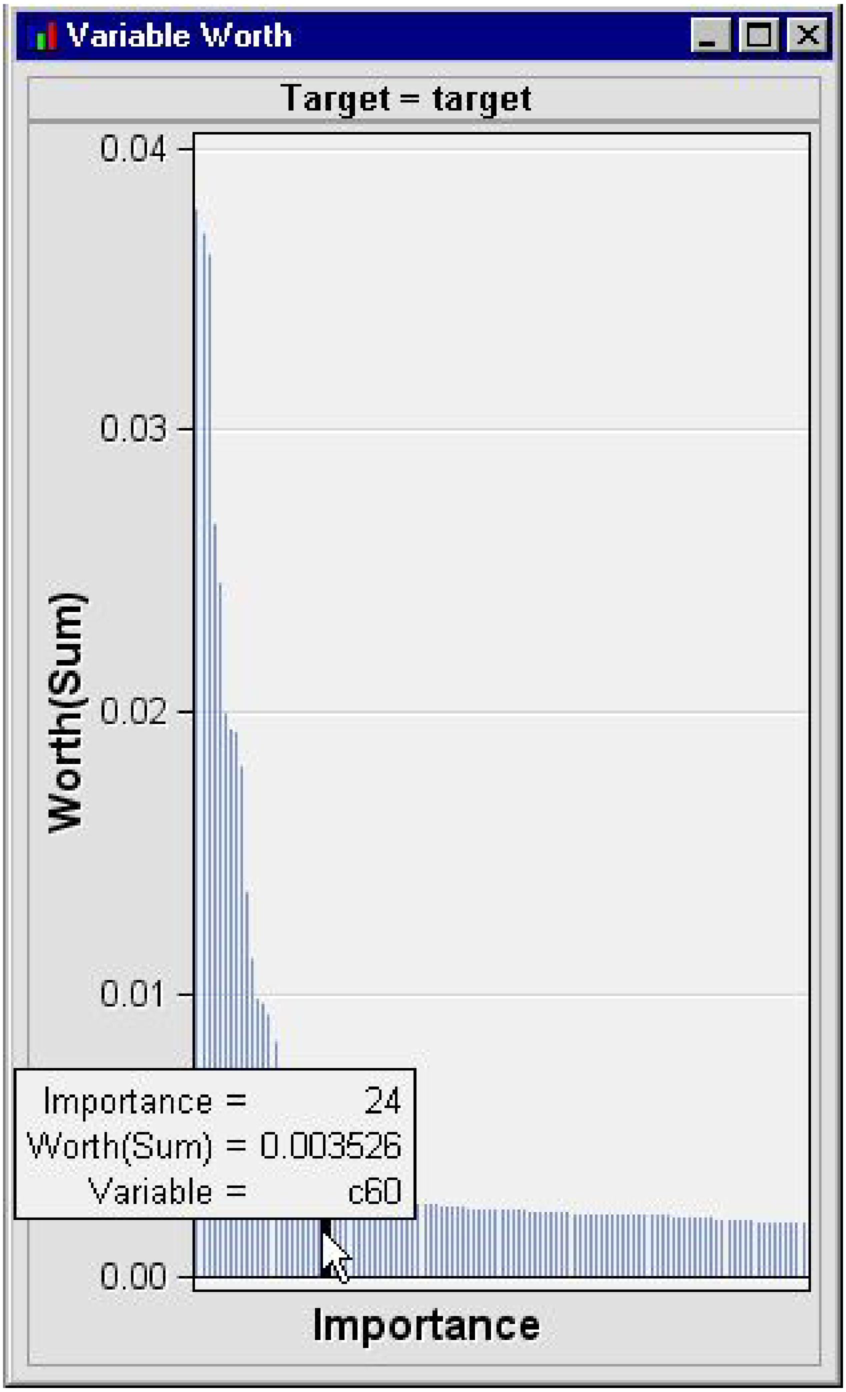
If the class data you submit to the StatExplore node contains both class target variables and segment variables, the node produces a lattice of bar charts that shows the distribution of values for each class variable within segment and target value. A maximum of 20 variables are selected according to a Cramer's V criterion.



The StatExplore node creates an additional plot that shows the scale deviation between the overall mean of an interval input variable and its mean for each level of the target variable. The plot orders the variables from the variable with the most deviation to the variable with the least deviation. If there is more than one target variable, a separate set of plots is created for each target variable.

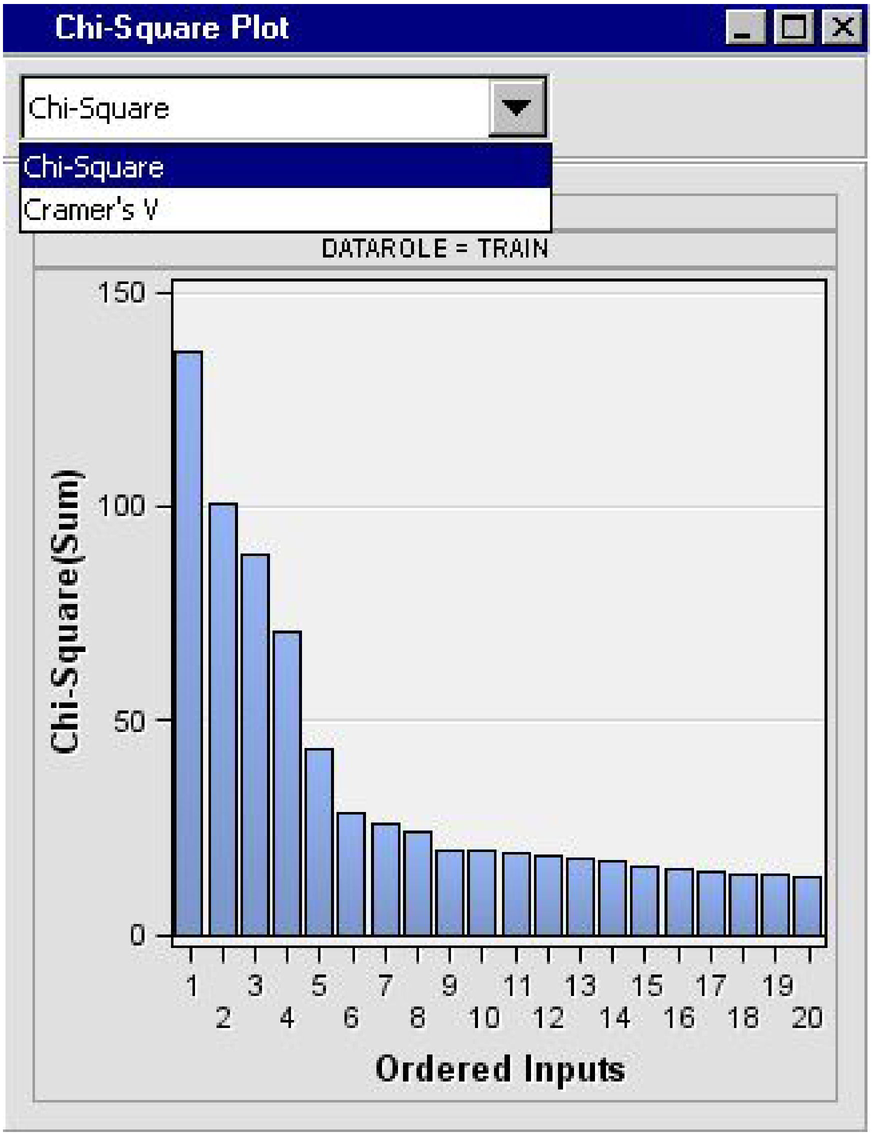


Finally, the StatExplore node generates a Variable Worth plot that ranks input variables according to their calculated worth. See Variable Importance in the help for the Decision Tree node for details on how a variable's worth (importance) is computed.



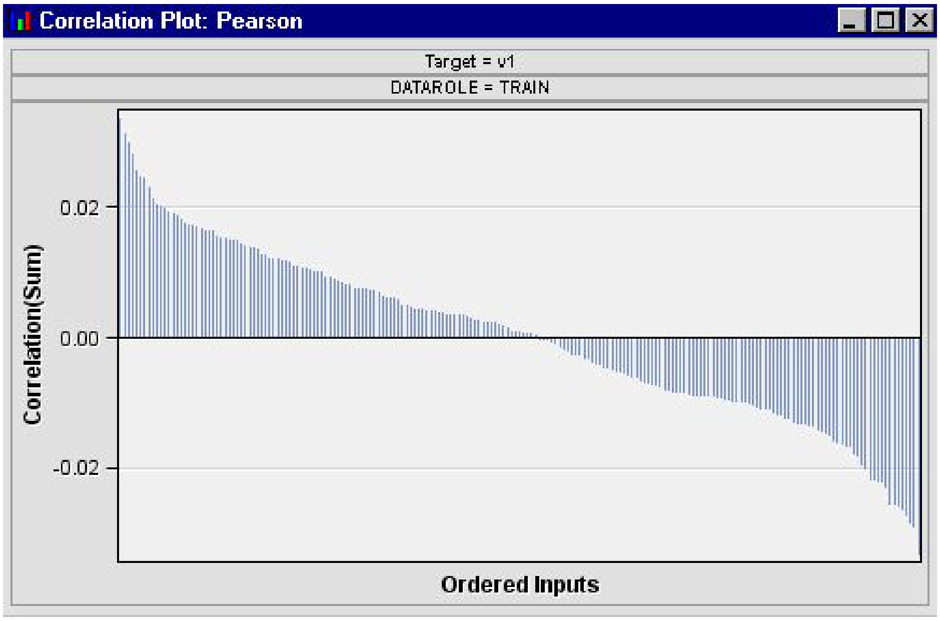
Importance = 24
Worth(Sum) = 0.003526
Variable = c60

If there are class target variables and the Chi-Square property in the StatExplore node Properties Panel is set to Yes, a combo plot of ordered inputs is also produced. Up to 20 variables are selected using a chi-square statistic or a Cramer's V staistic as the selection criterion. If more than one class target variable is specified, a Results Package is created for each class target variable.



Case 5: Interval Targets

If there are interval target variables and the Correlations property in the StatExplore Properties Panel is set to Yes, then the following scatter chart displays the correlation statistics:



When there are multiple interval targets, a separate correlation plot is produced for each target variable. If the Spearman Correlations property is set to Yes, then a Spearman Correlations plot is available. A Variable Worth plot, Class Variation plot, Class Variables plot, and Interval Variables plot, as depicted in the cases 1 and 2 above, are also available.

Class Variable Summary Statistics

The StatExplore node produces the following summary statistics for each class variable:

- Variable
- Role
- Number of Categories (Numcat)
- Number of Observations with Missing Variables (Nmiss)
- Mode n
- Mode n Percent

The StatExplore node also produces a table that lists the distribution of Class Targets and Segment Variables:

- Variable
- Role
- Formatted Values
- Frequency Count
- Percent of Total Frequency

Interval Variable Summary Statistics

The StatExplore node produces the following summary statistics for each interval variable:

- Variable

- Role
- Mean
- Standard Deviation
- N
- Min
- Median
- Max

The StatExplore node also produces a table that lists Interval Variable Summary Statistics by Class Target for each variable in the data set:

- Target
- Target Value
- Mean
- Standard Deviation
- N
- Min
- Median
- Max

Chi-Squared Summary Statistics

The StatExplore node produces a Chi-Square Statistics report for each target:

- Target Name
- Input
- Chi-Square Value
- Degrees of Freedom (DF)
- Probability

StatExplore Node Output Data Sources

The StatExplore node is an output-only node. The StatExplore node exports all input data sets without changing the data sets.

Note on Statistical Terms for Association of Categorical Variables

The Chi-square test is computed as:

$$\chi^2 = \frac{\sum\sum\left(n_{i,j} - e_{i,j}\right)^2}{e_{i,j}}$$

where

$$e_{i,j} = \frac{n_{i,.}n_{.,j}}{n}$$

In the case of a 2 x 2 relationship, this yields a phi-coefficient with a range of [-1,1].

$$\varphi = \sqrt{\frac{\chi^2}{n}}$$

Cramer's V is designed for two nominal variables with R and C categories with a range of [-1,1] for 2x2 variables and [0,1] otherwise.

$$V = \sqrt{\frac{\frac{\chi^2}{n}}{\min\left(R-1, C-1\right)}}$$

V is dependent on n , R, and, C. For example, increasing n decreases V. Also, small values of V often correspond to quite large proportional differences between groups. V should not be considered in isolation due to these issues.

Chapter 39

Variable Clustering Node

Variable Clustering Node

Overview of the Variable Clustering Node

The **Variable Clustering** node is on the **Explore** tab of the Enterprise Miner tools bar. Variable clustering is a useful tool for data reduction, such as choosing the best variables or cluster components for analysis. Variable clustering removes collinearity, decreases variable redundancy, and helps reveal the underlying structure of the input variables in a data set.

Suppose a direct mail company wants to build models using a database that has hundreds or thousands of variables for each customer. Should a predictive or segmentation model attempt to use all of the available variables? Large numbers of variables can complicate the task of determining the relationships that might exist between the independent variables and the target variable in a model. Models that are built with too many redundant variables can destabilize parameter estimates, confound variable interpretation, and increase the computing time that is required to run the model. Variable clustering can reduce the number of variables that are required to build reliable predictive or segmentation models.

Variable clustering divides numeric variables into disjoint or hierarchical clusters. The resulting clusters can be described as a linear combination of the variables in the cluster. The linear combination of variables is the first principal component of the cluster. The first principal components are called the cluster components. Cluster components provide scores for each cluster. The first principal component is a weighted average of the variables that explains as much variance as possible. The algorithm for Variable Clustering seeks the maximum variance that is explained by the cluster components, summed over all the clusters.

The **Variable Clustering** node cluster components are oblique, and not orthogonal, even when the cluster components are first principal components. In an ordinary principal component analysis, all components are computed from the same variables. The first principal component is orthogonal to the second principal component and to each other principal component. With the **Variable Clustering** node, each cluster component is computed from a different set of variables than all the other cluster components. The first principal component of one cluster might be correlated with the first principal component of another cluster. The **Variable Clustering** node performs a type of oblique component analysis.

As in principal component analysis, either the correlation or the covariance matrix can be analyzed. If correlations are used, all variables are treated as equally important. If covariances are used, variables with larger variances have more importance in the analysis.

When properly used as a variable-reduction tool, the **Variable Clustering** node can replace a large set of variables with the set of cluster components with little loss of information. A given number of cluster components does not generally explain as much variance as the same number of principal components on the full set of variables. However, the cluster components are usually easier to interpret than the principal components, even if the latter are rotated.

For example, an educational test might contain fifty items. The **Variable Clustering** node can be used to divide the items into, say, five clusters. Each cluster can then be treated as a subtest, with the subtest scores given by the cluster components.

Variable Clustering Node Algorithm

The algorithm behind the **Variable Clustering** node is both divisive and iterative. By default, the **Variable Clustering** node begins with all variables in a single cluster.

It then repeats the following steps:

1. A cluster is chosen for splitting. Depending on the options specified, the selected cluster has either the smallest percentage of variation explained by its cluster component (using the **Variation Proportion** property) or the largest eigenvalue that is associated with the second principal component (using the **Maximum Eigenvalue** property).
2. The chosen cluster is split into two clusters by finding the first two principal components, performing an orthoblique rotation (raw quartimax rotation on the eigenvectors; Harris and Kaiser, 1964), and assigning each variable to the rotated component with which it has the higher squared correlation.
3. Variables are iteratively reassigned to clusters to try to maximize the variance accounted for by the cluster components. You can require the reassignment algorithms to maintain a hierarchical structure for the clusters (using the Keep Hierarchies property).

The **Variable Clustering** node algorithm stops splitting when either:

- the maximum number of clusters as specified by the Maximum Clusters property is reached,

- each cluster satisfies the stopping criteria specified in the Variation Proportion property and / or the Maximum Eigenvalue properties

Using default settings, the **Variable Clustering** node stops splitting when each cluster has only one eigenvalue greater than 1, thus satisfying the most popular criterion for determining the sufficiency of a single underlying dimension.

The iterative reassignment of variables to clusters proceeds in two steps. The first step is a nearest component sorting phase, similar in principle to the nearest centroid sorting algorithms described by Anderberg (1973). In each iteration, the cluster components are computed, and each variable is assigned to the component with which it has the highest squared correlation. The second phase involves a search algorithm in which each variable is tested to see whether assigning it to a different cluster increases the amount of variance explained. If a variable is reassigned during the search phase, the components of the two clusters involved are recomputed before the next variable is tested. The nearest component sorting step is much faster than the search step but is more likely to be trapped by a local optimum.

Using principal components, the nearest component sorting step is an alternating least squares method and converges rapidly. The search step can be very time consuming for a large number of variables. But if the default initialization method is used, the search step is rarely able to substantially improve the results of the nearest component sorting step. In this case, the search takes fewer iterations. If random initialization is used, the nearest component sorting step might be trapped by a local optimum from which the search phase can escape.

You can use the **Variable Clustering** node to perform hierarchical clustering by using the Keep Hierarchies property. The Keep Hierarchies property restricts the reassignment of variables so that split clusters maintain a tree structure. When a cluster is split during hierarchical clustering, a variable in one of the two newly formed clusters can be reassigned to the other new cluster, but not to any other cluster.

Variable Clustering Node and Missing Data Set Values

If an observation contains missing values, the **Variable Clustering** node excludes the observation from the analysis. If the input data set that you want to perform variable clustering on contains a significant amount of observations with missing values, it might be beneficial to use the **Replacement** or **Impute** nodes to replace or impute missing variable values before submitting the data set to the **Variable Clustering** node.

Variable Clustering Node and Large Data Sets

The **Variable Clustering** node is most computationally effective when used with data sets that have less than 100 variables and less than 100,000 rows. Running the standard variable clustering algorithms with larger data sets results in noticeable processor performance decreases. When you need to perform variable clustering on a data set that has more than 100 variables or more than 100,000 rows, you should make changes to your process flow diagram configuration to improve the computational performance.

If your data set contains more than 100,000 observations, you should use the **Sampling** node to obtain a sample of the data set for variable clustering. You can select from several sampling strategies that reduce the number of observations in your process flow diagram to less than 100,000.

If your data set contains more than 100 variables, use the node's two-stage variable clustering algorithm. More detailed information is available in the section on “Two-Stage Variable Clustering” on page 584 .

Variable Clustering Node and Variable Roles

The **Variable Clustering** node is designed to cluster numeric variables. You can use the Include Class Variables property to analyze class variables through the use of dummy variables, but care should be used when including class variables in the analysis. **Variable Clustering** node performance can suffer when class variables that have a large number of levels are analyzed. This happens because the **Variable Clustering** node creates a dummy variable for each measurement level that it detects. Nominal class variables can be troublesome because every non-unique combination of characters or text would become a new dummy variable.

If your **Variable Clustering** node creates an unusually complex cluster map, you might want to inspect the measurement levels for the variables in your input data set. When you select the input data set node in your process flow diagram, you can use the **Variables** property to view your input data set variable measurement levels. The **Variable Clustering** node might provide better results if you reclassify some of your nominal variables in the table into interval variables.

Using the Variable Clustering Node

The default configuration of the **Variable Clustering** node often provides satisfactory results. If you want to change the final number of clusters, you can modify the settings for the **Maximum Clusters**, **Maximum Eigenvalue**, or **Variation Proportion** properties. The **Maximum Eigenvalue** and **Variation Proportion** criteria usually produce similar results, but occasionally can cause different clusters to be selected for splitting. The **Maximum Eigenvalue** criterion tends to choose clusters that have a large number of variables. The **Variation Proportion** criterion is more likely to select clusters that have a small number of variables.

The **Variable Clustering** node usually requires more computer processing than a comparable Principal Component analysis, but it can be faster than some of the iterative factoring methods.

If you have more than 30 variables, you can want to reduce your Variable Clustering node processing time by using one or more of the following methods:

- Specify a numeric value for the Maximum Clusters property if you know how many clusters you want.
- Set the Keep Hierarchies property to **Yes**.
- Set the Two Stage Clustering property to **Yes**.

Variable Clustering Node Properties

Variable Clustering Node General Properties

The following general properties are associated with the **Variable Clustering** node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Variable Clustering** node added to a diagram will have a Node ID of VarClus. The second **Variable Clustering** node added to a diagram will have a Node ID of VarClus2, and so on.
- **Imported Data** — The **Imported Data** property provides access to the Imported Data — Variable Clustering window. The Imported Data — Variable Clustering

window contains a list of the ports that provide data sources to the **Variable Clustering** node. Select the ... button to the right of the **Imported Data** property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Exported Data** — The **Exported Data** property provides access to the Exported Data — Variable Clustering window. The Exported Data — Variable Clustering window contains a list of the output data ports that the **Variable Clustering** node creates data for when it runs. Select the ... button to the right of the **Exported Data** property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Variable Clustering Node Train Properties

The following train properties are associated with the **Variable Clustering** node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **Variable Clustering** node. Select the ... button to open a window containing the variables table. You can set the variable status to either **Use** or **Don't Use** in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Clustering Source** — Use the Clustering Source property to specify the source matrix that you want to use for variable clustering analysis. The default setting for the Clustering Source property is Correlation.
  - **Correlation** — Setting the Clustering Source property to Correlation uses the correlation matrix of standardized variables. When the Cluster Component property is set to Principal Components, the source matrix provides eigenvalues.
  - **Covariance** — Setting the Clustering Source property to Covariance uses the covariance matrix of raw variables. The Covariance property setting permits variables that have a large variance to have more effect on the cluster components than variables that have a small variance. When the Cluster Component property is set to Principal Components, the source matrix provides eigenvectors.

- **Keep Hierarchies** — Use the Keep Hierarchies property to specify whether clusters should be maintained at different levels in order to create a hierarchical cluster structure. The default setting for the Keep Hierarchies property is **Yes**.
- **Include Class Variables** — The **Variable Clustering** node is designed to create clusters from numerical variables. The **Include Class Variables** property provides a way to include class variables in the variable clustering. The default setting for the **Include Class Variables** property is **No**. When the **Include Class Variables** property is set to **Yes**, class variables are converted to dummy variables and are treated as interval inputs. Users should pay close attention to the behavior of clustered dummy variables because dummy variables from one class variable can be sorted into different clusters. When dummy variables are created, the dummy variables are passed on to the node that follows the **Variable Clustering** node. When the **Include Class Variables** property is set to **No**, all class variables are passed on to the successor nodes without any processing.
- **Two Stage Clustering** — Use the **Two Stage Clustering** property to specify whether to perform two-stage clustering. Two-stage clustering is normally used when submitting data sets with more than 100 variables or 100,000 observations to the **Variable Clustering** node. The default setting for the **Two Stage Clustering** property is **Auto**. For more detailed information, see the section "Two-Stage Variable Clustering" on page 584 .

Variable Clustering Node Train Properties: Stopping Criteria Properties

- **Maximum Clusters** — Use the **Maximum Clusters** property to specify the largest number of clusters desired. If you do not specify a value for the **Maximum Clusters** property, the default setting counts the number of variables in the input data set and uses that value. The **Variable Clustering** node stops splitting clusters after reaching the **Maximum Clusters** value, regardless of what other splitting options are specified. Valid values are integers greater than or equal to 1.
- **Maximum Eigenvalue** — Use the **Maximum Eigenvalue** property to specify the largest permissible threshold for the second eigenvalue of each cluster. When the second eigenvalue of a cluster exceeds the **Maximum Eigenvalue** setting, the **Variable Clustering** node stops splitting the cluster. If you do not specify a value for the **Maximum Eigenvalue** property, the default setting counts the number of variables in the input data set and uses that value. Valid values are real numbers greater than or equal to 1.
- **Variation Proportion** — Use the **Variation Proportion** property to specify the upper threshold for the proportion of variation criterion. Using this criterion, the **Variable Clustering** node splits the cluster that has the smallest proportion of variation explained, as long as the proportion of variation value is less than or equal to the value specified for the **Variation Proportion** property. The default value for the **Variation Proportion** property is 0. The **Variation Proportion** property accepts real numbers between 0 and 1.0.

  If you specify values for both the **Variation Proportion** property and the **Maximum Eigenvalue** property, the **Variable Clustering** node searches first for a cluster to split using the **Maximum Eigenvalue** criterion. If no cluster meets the **Maximum Eigenvalue** criterion, the **Variable Clustering** node next looks for a cluster to split based on the **Variation Proportion** criterion. If you specify values for both the **Variation Proportion** property and the **Maximum Clusters** property, the resulting number of clusters will always be less than or equal to the **Maximum Clusters** property value.

- **Print Option** — Use the **Print Option** property to configure the granularity of the printed output from the **Variable Clustering** node.
    - **Summary** — Prints variable clustering results summary information only.
    - **Short** — Suppresses printing of the cluster structure, scoring coefficients, and inter-cluster correlation matrices. **Short** is the default setting for the **Print Option** property.
    - **All** — Prints all variable clustering results information.
    - **None** — Suppresses printing of all variable clustering results information.
- **Suppress Sampling Warning** — Use the **Suppress Sampling Warning** property to configure whether you want the **Variable Clustering** node to suppress the sampling warning when you submit a large data set for clustering. Performing variable clustering on large data sets that have 100,000 or more observations can be very computationally intensive. SAS Enterprise Miner recommends that you use sampling to perform variable clustering on large data sets. The default setting for the **Suppress Sampling Warning** is **No**. Set the **Suppress Sampling Warning** property to **Yes** if you want to run the **Variable Clustering** node without any restriction on the number of submitted observations.

Variable Clustering Node Score Properties

The following score properties are associated with the **Variable Clustering** node:

- **Variable Selection** — Use the **Variable Selection** property to specify the variables or components that you want to export from the clusters.
    - **Cluster Component** — the **Variable Clustering** node exports a linear combination of the variables from each cluster. **Cluster Component** is the default setting for the **Variable Selection** property.
    - **Best Variables** — the **Variable Clustering** node exports the variables in each cluster that have the minimum R-square ratio values.
- **Interactive Selection** — Use the **Interactive Selection** property to open an interactive variable selection table that permits you to manually choose important variables. Select the ... button to the right of the **Interactive Selection** property to open the variables table for interactive selection.
- **Hide Rejected Variables** — Use the **Hide Rejected Variables** property to specify whether to hide the rejected variables from all successor nodes in the process flow diagram. The default setting for the **Hide Rejected Variables** is **Yes**.

Variable Clustering Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.

- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

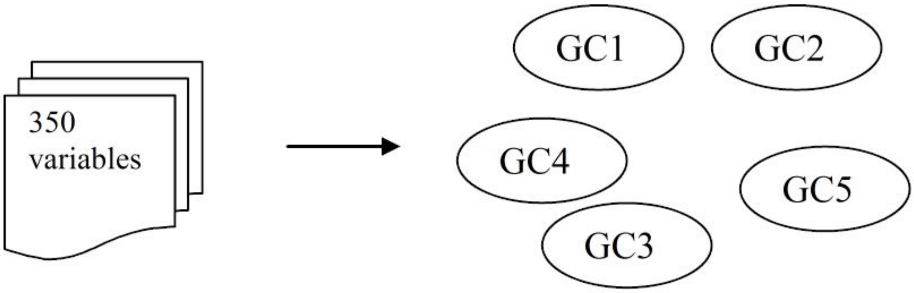
Two-Stage Variable Clustering

Variable clustering is a computationally intensive process. If the data set that you want to cluster contains more than 100 variables, you should use two-stage variable clustering to avoid long computation times.

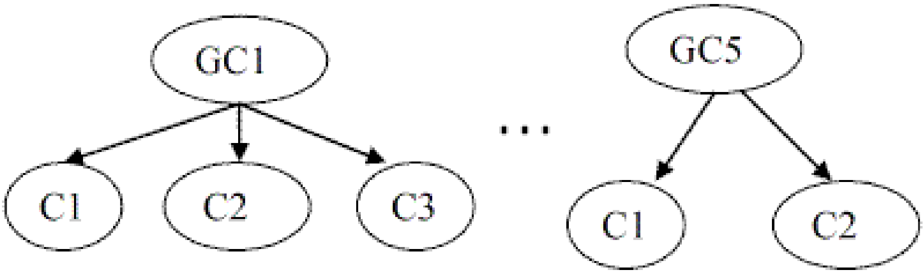
The two-stage variable clustering algorithm performs the following steps:

1. The **Variable Clustering** node identifies the variables, and then calculates correlations for all pairs of variables in the submitted data set.
2. The correlation matrix is used to cluster the variables into initial disjoint clusters called global clusters. The number of global clusters to be formed is calculated using the formula: Number of clusters = INT ((number of variables / 100) + 2).

   For example, a data set with 350 input variables undergoing two-stage variable clustering would create five initial global clusters: Number of clusters = INT ((350 / 100) + 2) = INT (5.5) = 5.



3. Variable clustering is performed on each of the global clusters.



4. The cluster components are calculated for each of the global clusters and used to reconstruct hierarchy clusters among global clusters.

Variable Clustering Node Results

You can open the Results window of the **Variable Clustering** node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

The default display for the **Variable Clustering** node Results window includes the following plots and tables:

- **Dendrogram** — a dendrogram tree that displays the cluster organization. Its initial view is based on Total Proportion of Variance statistics. You can change the x-axis values for this plot by right-clicking the dendrogram window and selecting the **Data Options** pop-up menu item.
- **Cluster Plot** — a spatial map that displays the orientation, size, and relative distance of clusters and cluster members.
- **Variable Selection Table** — a tabular listing of all of the clustered variables that includes cluster statistics and correlation values.
- **Variable Frequency Table** — a tabular listing of the clustered variables by frequency count and percent of total frequency.
- **Selected Variables Table** — a tabular listing of all of the selected clustered variables, including cluster statistics and correlation values.
- **Output** — the SAS output of the **Variable Clustering** node run. Typical output includes information such as a Variable Summary by Role, Level and Count, R-Squares, Chosen Effects, and Analysis of Variance (ANOVA) tables for the target variable, Estimating Logistic and Cutoff Classification tables, Node Split History, Split Effect Summary, and a Summary of Variable Selection.

Select **View** from the main menu to view the following additional results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the **Variable Clustering** node Properties Panel. The information was captured when the node was last run. Use the **Show Advanced Properties** check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the **Variable Clustering** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information about the data set submitted to the **Variable Clustering** node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — allows users to read previously created notes.
- **SAS Results**
  - **Log** — the SAS log of the **Variable Clustering** node run.
  - **Output** — the SAS output of the **Variable Clustering** node run.
  - **Flow Code** — the SAS code used to produce the output that the **Variable Clustering** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the **Variable Clustering** node does not generate SAS Code. The **SAS Code** menu item is dimmed and unavailable in the **Variable Clustering** node Results window.
  - **PMML Code** — the **Variable Clustering** node does not generate PMML code.

Variable Clustering Node Examples

Basic Variable Clustering Example

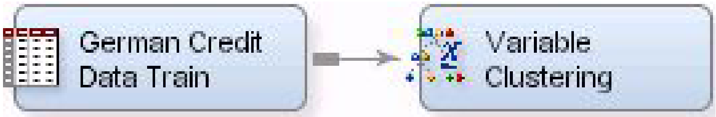
This example performs variable clustering on an example SAS data set composed of training metadata about applicants for credit. The training data has a binary target variable named GOOD_BAD. The GOOD_BAD status indicates whether the individual in the training data defaulted on his loan. The example uses variable clustering to create a model that uses a combination of latent vectors to predict the good risks among a pool of credit applicants.

Use the SAS Enterprise Miner Create Data Source Wizard to create the German Credit Data data source. The example data is located in the SAS sample library SAMPSIO. Use the SAS sample German Credit data set, SAMPSIO.DMAGECR.

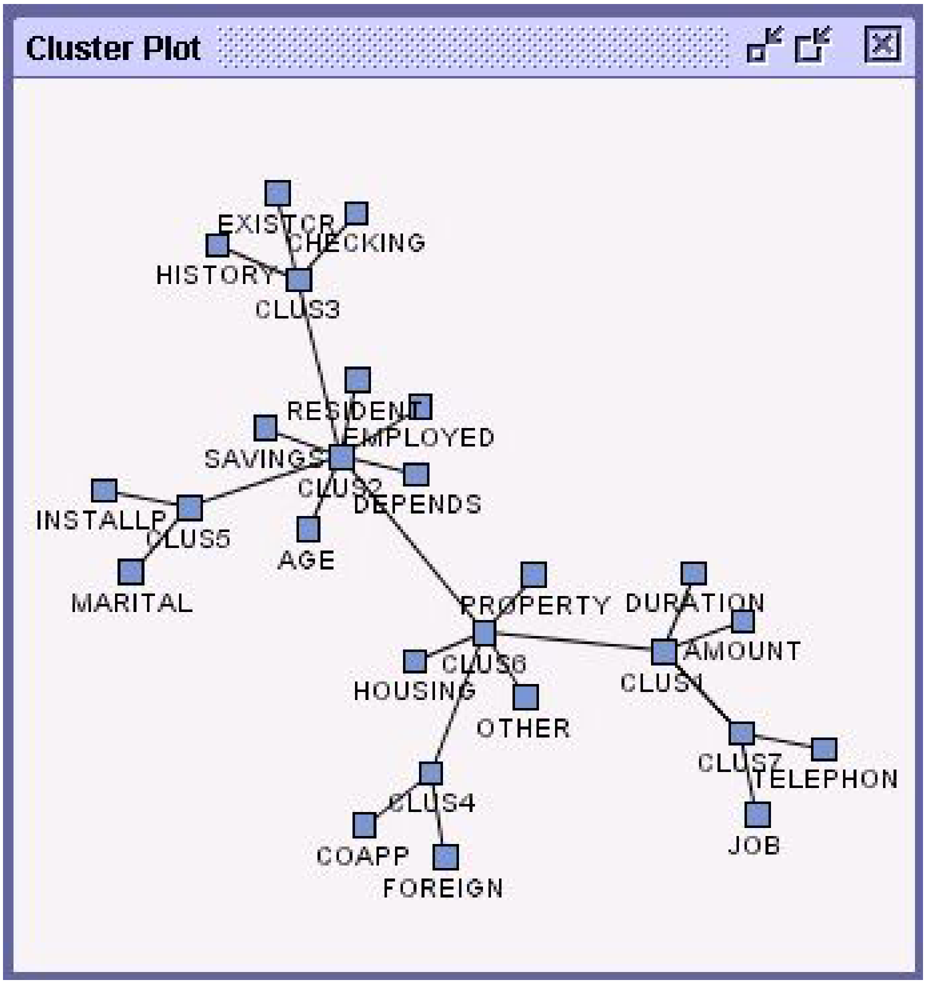
Configure the variables in the SAMPSIO.DMAGECR data set as follows:

- Set the role of the binary variable GOOD_BAD as the target variable.
- Set the measurement level of all the input variables except the character variable PURPOSE to Interval.
- Ensure that the measurement level of PURPOSE is set to Nominal, and that the measurement level of GOOD_BAD is Binary.
- Save the SAMPSIO.DMAGECR data set using the role of either Train or Raw.

Create a new process flow diagram. Next, connect the **German Credit** data source with a **Variable Clustering** node (found in the **Explore** tab of the node toolbar) as shown below. Use the default property settings for the **Variable Clustering** node.



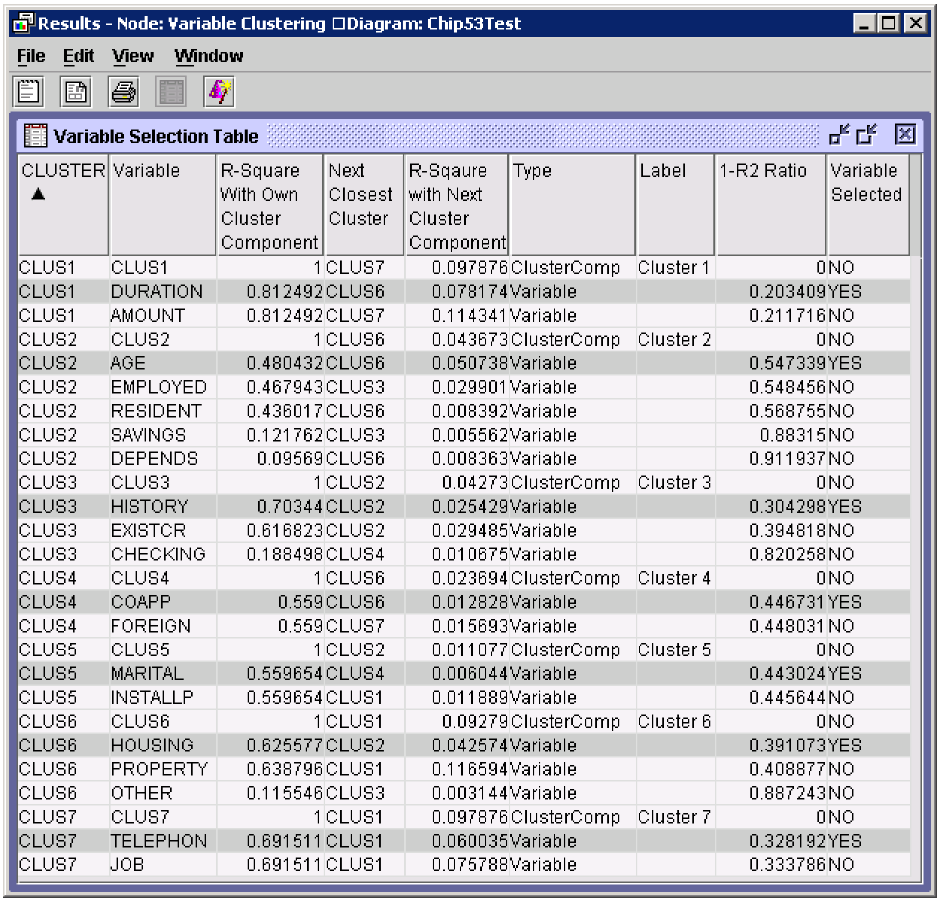
Right-click the **Variable Clustering** node, and then select **Run** to run the process flow diagram. When the Run Status window indicates that the run is completed, select the **Results** button to open the **Variable Clustering** node Results window. Examine the Cluster Plot to view the seven clusters that were created from the set of interval input variables in the German Credit training data:



Examine the Variable Selection table in the Results window to view the cluster statistics, sorted by cluster. The components in the Variable Selection table will be exported to the node that follows the **Variable Clustering** node in a process flow diagram. The **R-Square** with **Next Cluster Component** column of the table indicates the R-square scores with the nearest cluster. If the clusters are well separated, R-square score values should be low. Small values in the **1–R^2 Ratio** column of the Variable Selection table also indicate good clustering.

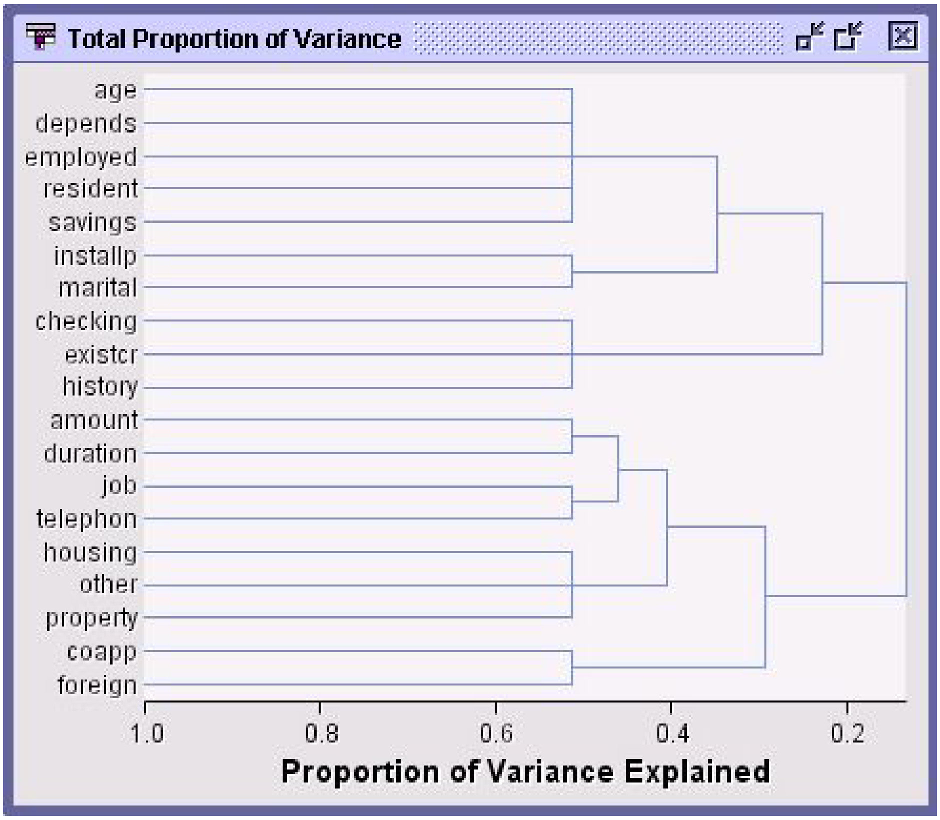
The **Variable Clustering** node run was performed using its default property settings. By default, the **Variable Clustering** node exports cluster components to the node that follows in a process flow diagram. Note that the **Variable Selected** column of the Variable Selection table in this case reads **YES** only for the cluster component variables (**CLUS1**, **CLUS2**, and so on.) The **Variable Clustering** node can also export the best variable from each cluster to successor nodes if you desire.

To export the best variable from each cluster (instead of cluster components), close the **Variable Clustering** node Results window, and then change the value of the **Variable Selection** property from **Cluster Component** to **Best Variables**. Run the node after you make the configuration changes and open the Results window. When you examine the Variable Selection table, the **Variables Selected** column displays **YES** for the best variable in each cluster. When you use the **Best Variable** setting for the **Variable Selection** property, the best variable in each cluster is the variable that has the lowest **1–R^2 Ratio**.

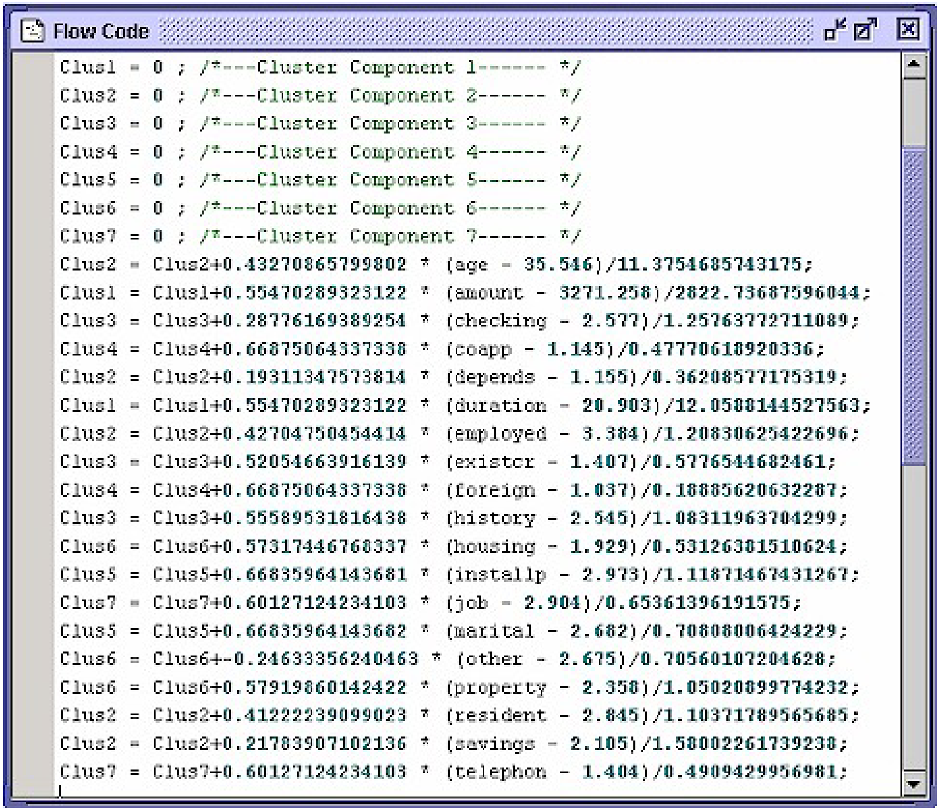


CLUSTER ▲	Variable	R-Square With Own Cluster Component	Next Closest Cluster	R-Sqaure with Next Cluster Component	Type	Label	1-R2 Ratio	Variable Selected
CLUS1	CLUS1	1	CLUS7	0.097876	ClusterComp	Cluster 1	0	NO
CLUS1	DURATION	0.812492	CLUS6	0.078174	Variable		0.203409	YES
CLUS1	AMOUNT	0.812492	CLUS7	0.114341	Variable		0.211716	NO
CLUS2	CLUS2	1	CLUS6	0.043673	ClusterComp	Cluster 2	0	NO
CLUS2	AGE	0.480432	CLUS6	0.050738	Variable		0.547339	YES
CLUS2	EMPLOYED	0.467943	CLUS3	0.029901	Variable		0.548456	NO
CLUS2	RESIDENT	0.436017	CLUS6	0.008392	Variable		0.568755	NO
CLUS2	SAVINGS	0.121762	CLUS3	0.005562	Variable		0.88315	NO
CLUS2	DEPENDS	0.09569	CLUS6	0.008363	Variable		0.911937	NO
CLUS3	CLUS3	1	CLUS2	0.04273	ClusterComp	Cluster 3	0	NO
CLUS3	HISTORY	0.70344	CLUS2	0.025429	Variable		0.304298	YES
CLUS3	EXISTCR	0.616823	CLUS2	0.029485	Variable		0.394818	NO
CLUS3	CHECKING	0.188498	CLUS4	0.010675	Variable		0.820258	NO
CLUS4	CLUS4	1	CLUS6	0.023694	ClusterComp	Cluster 4	0	NO
CLUS4	COAPP	0.559	CLUS6	0.012828	Variable		0.446731	YES
CLUS4	FOREIGN	0.559	CLUS7	0.015693	Variable		0.448031	NO
CLUS5	CLUS5	1	CLUS2	0.011077	ClusterComp	Cluster 5	0	NO
CLUS5	MARITAL	0.559654	CLUS4	0.006044	Variable		0.443024	YES
CLUS5	INSTALLP	0.559654	CLUS1	0.011889	Variable		0.445644	NO
CLUS6	CLUS6	1	CLUS1	0.09279	ClusterComp	Cluster 6	0	NO
CLUS6	HOUSING	0.625577	CLUS2	0.042574	Variable		0.391073	YES
CLUS6	PROPERTY	0.638796	CLUS1	0.116594	Variable		0.408877	NO
CLUS6	OTHER	0.115546	CLUS3	0.003144	Variable		0.887243	NO
CLUS7	CLUS7	1	CLUS1	0.097876	ClusterComp	Cluster 7	0	NO
CLUS7	TELEPHON	0.691511	CLUS1	0.060035	Variable		0.328192	YES
CLUS7	JOB	0.691511	CLUS1	0.075788	Variable		0.333786	NO

The Results window dendrogram uses a tree hierarchy to display how the clusters were formed.



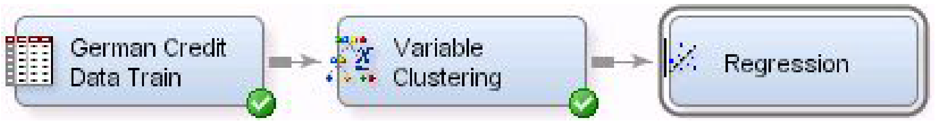
You can use the Results window menu to display the flow code for the cluster components. The flow code contains the score code: **View** ⇨ **SAS Results** ⇨ **Flow Code**.



```
Clus1 = 0 ; /*---Cluster Component 1------ */
Clus2 = 0 ; /*---Cluster Component 2------ */
Clus3 = 0 ; /*---Cluster Component 3------ */
Clus4 = 0 ; /*---Cluster Component 4------ */
Clus5 = 0 ; /*---Cluster Component 5------ */
Clus6 = 0 ; /*---Cluster Component 6------ */
Clus7 = 0 ; /*---Cluster Component 7------ */
Clus2 = Clus2+0.43270865799802 * (age - 35.546)/11.3754685743175;
Clus1 = Clus1+0.55470289323122 * (amount - 3271.258)/2822.73687596044;
Clus3 = Clus3+0.28776169389254 * (checking - 2.577)/1.25763772711089;
Clus4 = Clus4+0.66875064337338 * (coapp - 1.145)/0.47770618920336;
Clus2 = Clus2+0.19311347573814 * (depends - 1.155)/0.36208577175319;
Clus1 = Clus1+0.55470289323122 * (duration - 20.903)/12.0588144527563;
Clus2 = Clus2+0.42704750454414 * (employed - 3.384)/1.20830625422696;
Clus3 = Clus3+0.52054663916139 * (existcr - 1.407)/0.5776544682461;
Clus4 = Clus4+0.66875064337338 * (foreign - 1.037)/0.18885620632287;
Clus3 = Clus3+0.55589531816438 * (history - 2.545)/1.08311963704299;
Clus6 = Clus6+0.57317446768337 * (housing - 1.929)/0.53126301510624;
Clus5 = Clus5+0.66835964143681 * (installp - 2.973)/1.11871467431267;
Clus7 = Clus7+0.60127124234103 * (job - 2.904)/0.65361396191575;
Clus5 = Clus5+0.66835964143682 * (marital - 2.682)/0.70808006424229;
Clus6 = Clus6+-0.24633356240463 * (other - 2.675)/0.70560107204628;
Clus6 = Clus6+0.57919860142422 * (property - 2.358)/1.05020899774232;
Clus2 = Clus2+0.41222239099023 * (resident - 2.845)/1.10371789565685;
Clus2 = Clus2+0.21783907102136 * (savings - 2.105)/1.58002261739238;
Clus7 = Clus7+0.60127124234103 * (telephon - 1.404)/0.49094299956981;
```

The type of output that the **Variable Clustering** node produces varies according to how the node **Export Options** are configured. You can verify which variables are exported during a **Variable Clustering** node run by attaching a successor node to the **Variable Clustering** node and running the successor node. After the run, open the table for the successor node **Variables** property to view the input variables that were exported by the **Variable Clustering** node.

This example has produced both cluster components and best variable cluster values during separate **Variable Clustering** node runs. To view how exported data passes to successor nodes, add a successor **Regression** node and configure the **Variable Clustering** node to export cluster components.

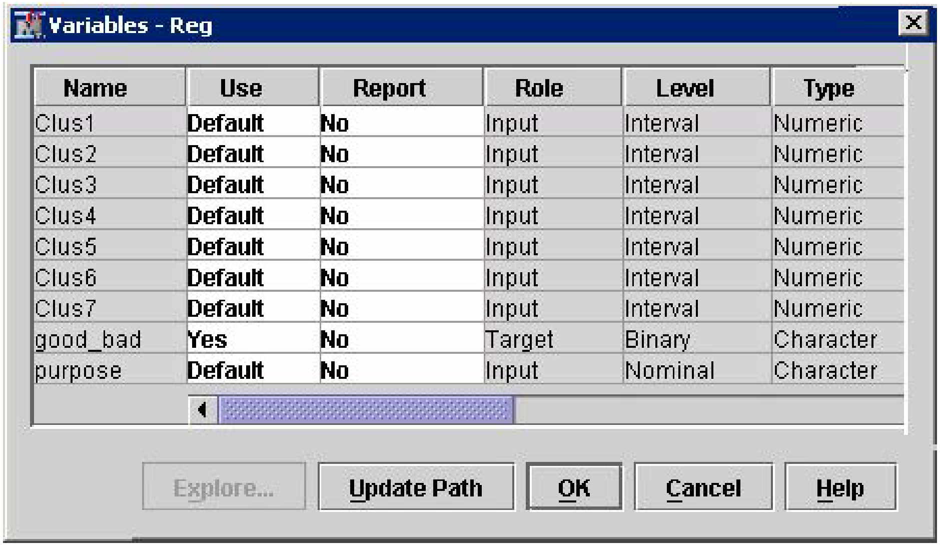


Drag a **Regression** node from the **Model** tab of the node tools bar onto your process flow diagram. Connect the **Variable Clustering** node to the **Regression** node. Leave the **Regression** node properties in their default states. The **Variable Clustering** node needs to be configured to export cluster components instead of the best variables within each cluster.

Select the **Variable Clustering** node in the process flow diagram, and then set the following properties:

- set the **Variable Selection** property in the **Export Options** section to **Cluster Component**.
- ensure that the **Hide Rejected Variables** property in the **Export Options** section is set to **Yes**.
- set the **Include Class Variables** property to **No**.

When the run completes, select **OK** in the Run Status window. (This example does not visit the results of the **Regression** node.) With the **Regression** node selected in the process flow diagram, click on the ... button to the right of the **Variables** property in the **Regression** node general properties panel. A table opens that displays all of the variables that the **Variable Clustering** node exported to the **Regression** node:



Name	Use	Report	Role	Level	Type
Clus1	Default	No	Input	Interval	Numeric
Clus2	Default	No	Input	Interval	Numeric
Clus3	Default	No	Input	Interval	Numeric
Clus4	Default	No	Input	Interval	Numeric
Clus5	Default	No	Input	Interval	Numeric
Clus6	Default	No	Input	Interval	Numeric
Clus7	Default	No	Input	Interval	Numeric
good_bad	Yes	No	Target	Binary	Character
purpose	Default	No	Input	Nominal	Character

The table shows the exported variables: the GOOD_BAD target variable, the nominal PURPOSE variable that was neither analyzed nor rejected, and the seven variable cluster components that the **Variable Clustering** node created.

Because the value of the **Hide Rejected Variables** property of the **Variable Clustering** node was set to **Yes**, none of the rejected variables are passed on to the successor node.

Two Stage Variable Clustering Example

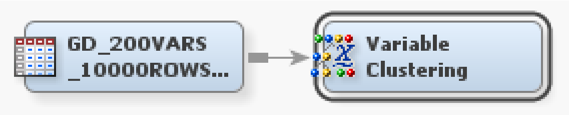
The two stage variable clustering example uses a data set called GD_200VARS_10000ROWS_NOCLASS. The example data set is typical of a data source that should use two-stage variable clustering. The example data source is a raw data set that has 200 interval input variables, no class variables, and 10,000 observations. The data set has a binary target variable called TARGET. This example configures the input data source as a training data set.

For the input data set, use the **Create Data Source Wizard** to create a data source from the SAS SAMPSIO sample library. Use the SAS data set, SAMPSIO.GD_200VARS_10000ROWS_NOCLASS to create a new data source.

Confirm that the newly created data source is configured as follows:

- the **Advanced** advisor is selected.
- the binary variable TARGET is the target variable.
- ID_CUST and ID_STORE are nominal ID variables.
- SEGMENT is a nominal segment variable.
- the remaining variables are all interval input variables.

Construct the following process flow diagram:



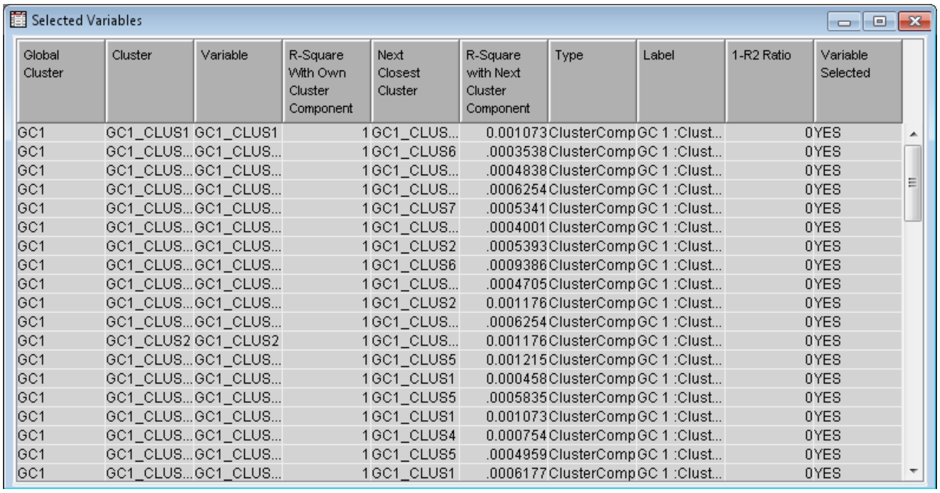
Select the **Variable Clustering** node and configure the following properties:

- Set the **Two Stage Clustering** property to **Yes**.
- Leave all other property settings in their default state. The node will export cluster components.

Right-click the **Variable Clustering** node in the process flow diagram and select **Run**. In the Confirmation window, select **Yes**. In the Run Status window, select **Results**.

You can see that within each global cluster, all results are similar to those for single-stage clustering. The global clusters in the **GCluster** column are labeled **GC1**, **GC2**, and so on. The formula for the number of global clusters is Number of clusters = INT ((number of variables / 100) + 2). Therefore, the number of clusters is INT ((200 / 100) + 2) = 4.

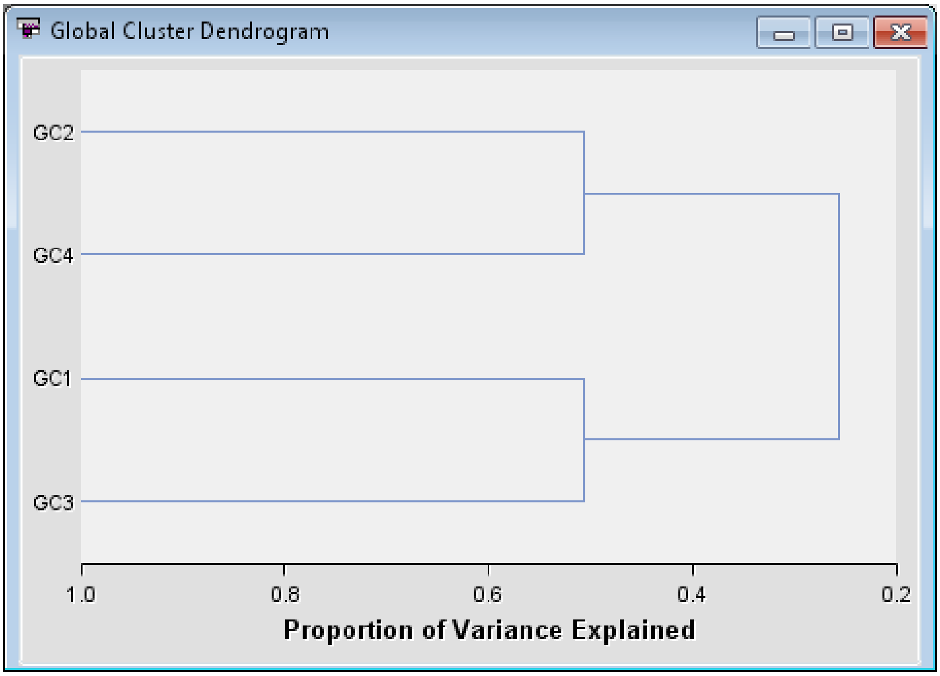
Because there are four global clusters, the results contain four cluster plots, four cluster tables, and so on. The table for global cluster 1 (**GC1**) is shown below:



Selected Variables

Global Cluster	Cluster	Variable	R-Square With Own Cluster Component	Next Closest Cluster	R-Square with Next Cluster Component	Type	Label	1-R2 Ratio	Variable Selected
GC1	GC1_CLUS1	GC1_CLUS1	1	GC1_CLUS...	0.001073	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS6	.0003538	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS...	.0004838	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS...	.0006254	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS7	.0005341	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS...	.0004001	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS2	.0005393	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS6	.0009386	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS...	.0004705	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS2	0.001176	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS...	.0006254	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS2	GC1_CLUS2	1	GC1_CLUS...	0.001176	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS5	0.001215	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS1	0.000458	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS5	.0005835	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS1	0.001073	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS4	0.000754	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS5	.0004959	ClusterComp	GC 1 :Clust...	0	YES
GC1	GC1_CLUS...	GC1_CLUS...	1	GC1_CLUS1	.0006177	ClusterComp	GC 1 :Clust...	0	YES

The Global Cluster Dendrogram below is generated from scored cluster components and hierarchical clustering. It shows the relationship between global clusters and provides a link to the gap between global clusters and sub-clusters.



Predictive Modeling with Variable Clustering Example

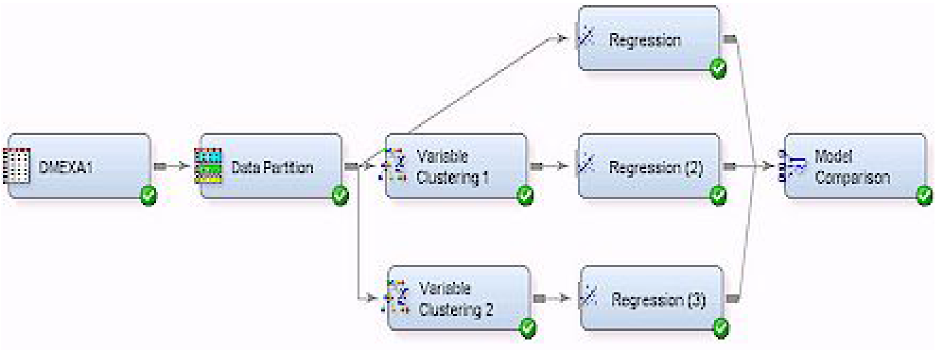
This example demonstrates how the **Variable Clustering** node can enhance the performance of a data mining predictive model. The SAS sample data set SAMPSIO.DMEXA1 is used as a data source. The data source is partitioned into training and validation data sets for a regression analysis.

The partitioned data is passed on to three identically configured successor **Regression** nodes with the following variations:

- The data that flows to the first **Regression** node has no variable clustering performed on the data.
- The data that flows to the second **Regression** node has variable clustering performed on the data. The **Variable Clustering 1** node is configured to output cluster components.

- The data that flows to the third **Regression** node has variable clustering performed on the data. The **Variable Clustering 2** node is configured to output the best cluster variables.

The output from the three regression models is then submitted to a **Model Comparison** node to assess the regression model with the best performance.



1. Configure the DMEXA1 data source as follows:

   - Use the Create Data Source Wizard to select the sample SAS data set SAMPSIO.DMEXA1.
   - After you specify the SAS table SAMPSIO.DMEXA1 in the Data Source Wizard, select the **Basic** setting for the metadata advisor.
   - In the Basic Column Metadata table for SAMPSIO.DMEXA1, make the following configurations:
     - Set the **Role** for variable ACCTNUM to **ID**.
     - Set the **Role** for variable STATECOD to **Rejected**.
     - Set the **Role** for variable PURCHASE to **Target**.
   - In the Data Source Attributes table for SAMPSIO.DMEXA1, set the **Data Source Role** attribute to your choice of **Train** or **Raw**.

   The newly created SAMPSIO.DMEXA1 data source should have 43 interval input variables, 4 nominal input variables, and 1,966 observations.

2. Add and connect a **Data Partition** node to the diagram. In the **Data Set Percentages** section of the **Data Partition** node properties panel, make the following configurations:

   - Set the **Training** property value to 60%.
   - Set the **Validation** property value to 40%.

3. Add and connect two **Variable Clustering** nodes to the diagram as shown. In the **Export Options** section of the **Variable Clustering** node properties panel, make the following configurations:

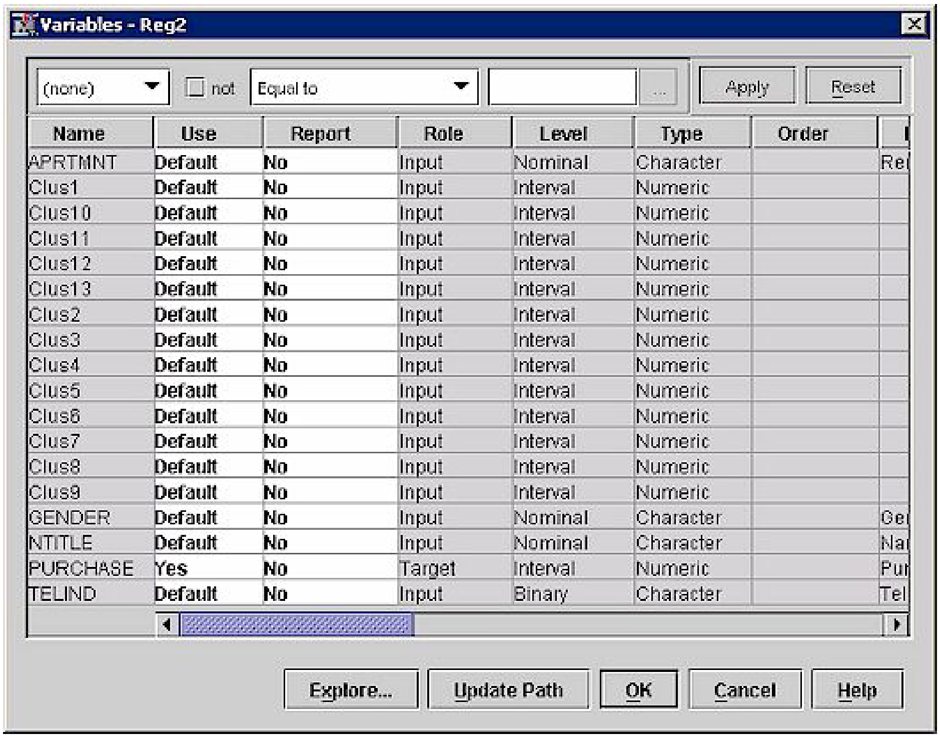
   - Set the **Variable Selection** property value on the uppermost **Variable Clustering** node in the diagram to **Cluster Components**.
   - Set the **Variable Selection** property value on the remaining **Variable Clustering** node in the diagram to **Best Variables**.

4. Add and connect three **Regression** nodes to the diagram as shown. Leave all three **Regression** nodes in their default configuration.

5. Add and connect the **Model Comparison** node to the diagram as shown. Leave the **Model Comparison** node in its default configuration.

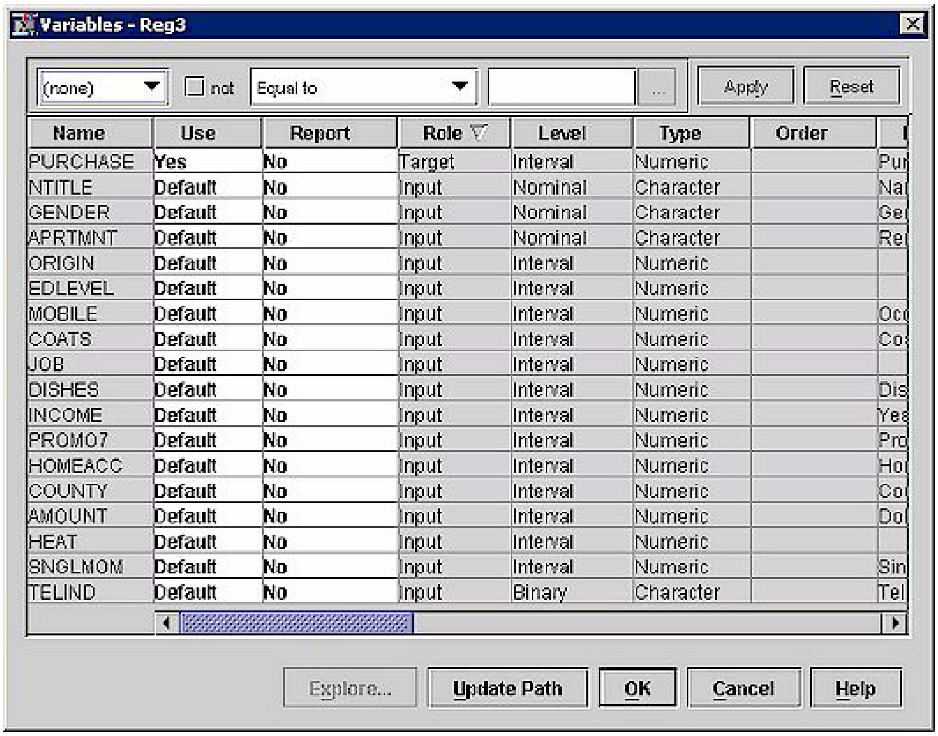
6. Right-click the **Model Comparison** node and select **Run** from the pop-up menu. In the Run Status window, select **OK**.

7. Before examining the **Model Comparison** node results, examine the variables that were passed to the two **Regression** nodes that follow **Variable Clustering** nodes. In the properties panel for the **Regression** node that follows the uppermost **Variable Clustering** node, select the ... button that is located to the right of the **Variables** property to view the table of imported variables. The table shows that the first **Variable Clustering** node summarized 43 interval variables in 13 cluster components, and then passed the cluster components to the successor **Regression** node as new inputs.



Name	Use	Report	Role	Level	Type	Order	I
APRTMNT	Default	No	Input	Nominal	Character		Re
Clus1	Default	No	Input	Interval	Numeric		
Clus10	Default	No	Input	Interval	Numeric		
Clus11	Default	No	Input	Interval	Numeric		
Clus12	Default	No	Input	Interval	Numeric		
Clus13	Default	No	Input	Interval	Numeric		
Clus2	Default	No	Input	Interval	Numeric		
Clus3	Default	No	Input	Interval	Numeric		
Clus4	Default	No	Input	Interval	Numeric		
Clus5	Default	No	Input	Interval	Numeric		
Clus6	Default	No	Input	Interval	Numeric		
Clus7	Default	No	Input	Interval	Numeric		
Clus8	Default	No	Input	Interval	Numeric		
Clus9	Default	No	Input	Interval	Numeric		
GENDER	Default	No	Input	Nominal	Character		Ge
NTITLE	Default	No	Input	Nominal	Character		Na
PURCHASE	Yes	No	Target	Interval	Numeric		Pu
TELIND	Default	No	Input	Binary	Character		Tel

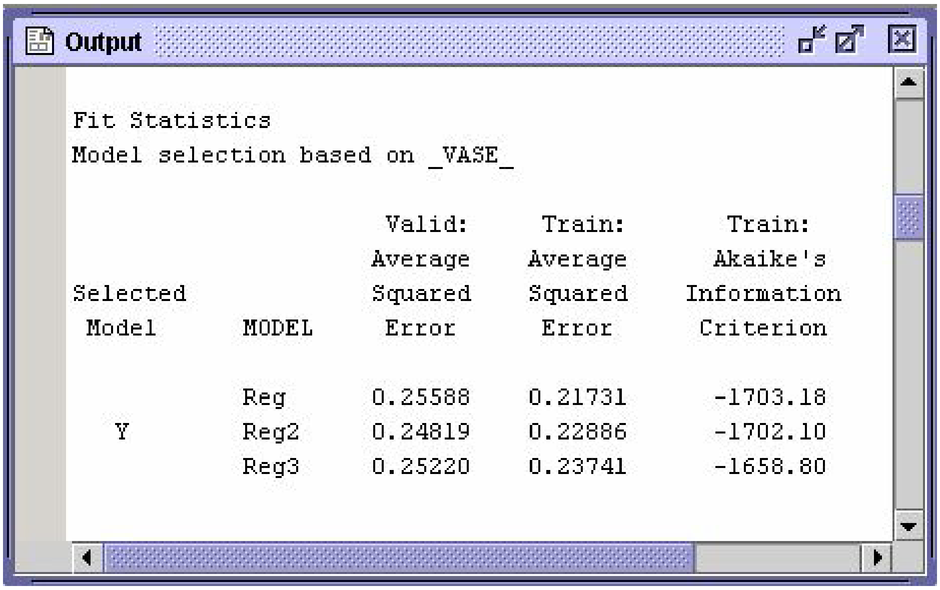
8. In the properties panel for the **Regression** node that follows the second **Variable Clustering** node, select the ... button that is located to the right of the **Variables** property to view the table of imported variables. The table shows that the second **Variable Clustering** node chose 13 interval variables, each variable the best cluster representative. The 13 best cluster variables are passed to the successor **Regression** node as new inputs.



Name	Use	Report	Role ▽	Level	Type	Order	I
PURCHASE	Yes	No	Target	Interval	Numeric		Pur
NTITLE	Default	No	Input	Nominal	Character		Nai
GENDER	Default	No	Input	Nominal	Character		Gei
APRTMNT	Default	No	Input	Nominal	Character		Rei
ORIGIN	Default	No	Input	Interval	Numeric		
EDLEVEL	Default	No	Input	Interval	Numeric		
MOBILE	Default	No	Input	Interval	Numeric		Occ
COATS	Default	No	Input	Interval	Numeric		Co:
JOB	Default	No	Input	Interval	Numeric		
DISHES	Default	No	Input	Interval	Numeric		Dis
INCOME	Default	No	Input	Interval	Numeric		Yes
PROMO7	Default	No	Input	Interval	Numeric		Pro
HOMEACC	Default	No	Input	Interval	Numeric		Hoi
COUNTY	Default	No	Input	Interval	Numeric		Coi
AMOUNT	Default	No	Input	Interval	Numeric		Dol
HEAT	Default	No	Input	Interval	Numeric		
SNGLMOM	Default	No	Input	Interval	Numeric		Sin
TELIND	Default	No	Input	Binary	Character		Tel

9. Right-click the **Model Comparison** node in the diagram and select **Results** from the pop-up menu to compare the results of the three **Regression** models.

10. The Fit Statistics table and the SAS Log show that the **Model Comparison** node chose the regression model that followed the **Variable Clustering** node that exported cluster components as the best predictive model. The regression models use unclustered variable inputs or best cluster variable inputs based on validation error and misclassification rates.



```
Fit Statistics
Model selection based on _VASE_

                            Valid:      Train:       Train:
                            Average     Average      Akaike's
Selected                    Squared     Squared     Information
 Model        MODEL          Error       Error       Criterion

              Reg          0.25588     0.21731      -1703.18
   Y          Reg2         0.24819     0.22886      -1702.10
              Reg3         0.25220     0.23741      -1658.80
```

Chapter 40
Variable Selection Node

Variable Selection Node

Overview of the Variable Selection Node

The Variable Selection node tool is located on the Explore tab of the Enterprise Miner tools bar.

Many data mining databases have hundreds of potential model inputs (independent or explanatory variables) that can be used to predict the target (dependent or response variable). The Variable Selection node can assist you in reducing the number of inputs by setting the status of the input variables that are not related to the target as rejected. Although rejected variables are passed to subsequent nodes in the process flow, these variables are not used as model inputs by a successor modeling node.

The Variable Selection node quickly identifies input variables which are useful for predicting the target variable or variables. The input status is assigned to these variables. You can override the automatic selection process by assigning the input status to a rejected variable or the rejected status to an input variable. The information rich inputs can then be evaluated in more detail by one of the modeling nodes.

To preselect the important input variables, the Variable Selection node uses either a R-square or Chi-square selection criterion. Note that if the target is nominal or ordinal, create a binary dummy target variable or variables instead of using the original ordinal or nominal target variable. A step-wise regression analysis with large entry and stay probabilities may be suitable for variable screening.

In the Variable Selection node, missing values are treated as follows:

- Observations that have missing values in the target variables are excluded from the analysis.

- Missing values in the categorical input variable are treated as an additional category.
- Missing values in the interval input variable are replaced by the weighted mean of the variable.

The node can be run prior to any other analysis and the results passed to any Enterprise Miner node or any procedure in the SAS System.

Variable Selection Node Properties

Variable Selection Node General Properties

The following general properties are associated with the Variable Selection node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Variable Selection node added to a diagram will have a Node ID of Varsel. The second Variable Selection node added to a diagram will have a Node ID of VarSel, and so on.
- **Imported Data** — accesses the Imported Data — Variable Selection window. The Imported Data — Variable Selection window contains a list of the ports that provide data sources to the Variable Selection node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — Variable Selection window. The Exported Data — Variable Selection window contains a list of the output data ports that the Variable Selection node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Variable Selection Node Train Properties

The following train properties are associated with the Variable Selection node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the Variable Selection node. Select the ... button to open a window containing the variables table. You can set the variable status to either **Use** or **Don't Use** in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Max Class Level** — Use the Max Class Level property to specify the maximum number of class levels that you want to allow. If the number of class levels exceeds the specified threshold, the class variable is rejected. Permissible values for the Max Class Level property are 5, 10, 20, 30, 40, 50, 75, 100, 125, 150, 200, 250, 300, 500, 750, 1000, and 10000. The default setting is 100.
- **Max Missing Percentage** — Use the Max Missing Percentage property to specify the maximum percentage of observations where an input variable has missing data. Variables that exceed the missing data threshold value are rejected. Permissible values for the Max Missing Percentage property are real numbers between 0 and 100. The default setting is 50.
- **Target Model** — Use the Target Model property to choose the variable selection method that you want to use.
    - **Default** — The Default selection uses target and model information to choose the selection method. If the target is binary and the model has greater than 400 degrees of freedom, the Chi-Square method is selected. Otherwise, the R-Square method is used.
    - **R-Square** — the R-Square selection uses a forward stepwise least squares regression that maximizes the model R-square value. This criterion provides a fast preliminary variable assessment and facilitates the rapid development of predictive models with large volumes of data. You can quickly identify input variables that are useful for predicting the target variable(s) based on a linear models framework. The following three step process is performed when you apply the R-Square variable selection criterion to a binary target (the last step is not applied if the target variable is non-binary):
        - **Compute the Squared Correlations** — The squared correlation coefficient (simple R2) for each input variable is computed and compared to the default Minimum R-Square of 0.005. If the squared correlation coefficient for an input is less than the cut-off-criterion, then the input variable is set to the rejected role. Depending on your data mining needs, you may want to change the default Minimum R-Square (cutoff criterion). Specific knowledge about your project or field may suggest an appropriate cutoff. Increasing the cutoff value will tend to remove more input variables; decreasing the cutoff value will tend to retain more input variables. The squared correlation coefficient is the proportion of target variation explained by a single input variable; the effect of the other input variables is not included in it's calculation. Statisticians also refer to it as the coefficient of determination, which ranges from 0 (no linear relationship between an input and the target) to 1 (the input explains all of the target variability). The Variable Selection node performs a simple linear regression to obtain the squared correlation coefficient value for interval variables, such as salary or average daily balance. The Variable Selection node performs a one-way analysis of variance to calculate the squared correlation coefficient value for class (group) variables, such as region or department.
        - **Forward Stepwise Regression** — After computing the squared correlation coefficient for each variable, the remaining significant variables are evaluated using a forward stepwise R2 regression. The sequential forward selection process starts by selecting the input variable that explains the largest amount of variation in the target. This is the variable that has the highest squared

        correlation coefficient. At each successive step, an additional input variable is chosen that provides the largest incremental increase in the model R2. The stepwise process terminates when no remaining input variables can meet the Stop R-Square criterion (the default value is 0.0005).

        - **Logistic Regression for Binary Targets** — If the target is a binary variable, then a final logistic regression analysis is performed using the predicted values that are output from the forward stepwise selection as the independent input. Because there is only one input, only two parameters are estimated (the intercept and the slope). The range of the predicted values is divided into a number of equidistant intervals (knots), on which the logistic function is interpolated.

    - **Chi-Square** — The Chi-Square selection criterion is available for binary targets only; you can use a class target but it will be converted into a binary target prior to variable selection modeling. This criterion provides a fast preliminary variable assessment and facilitates the rapid development of predictive models with large volumes of data. You can quickly identify input variables which are useful for predicting a binary target variable(s) based on a chi-square assessment. The Chi-Square selection uses binary variable splits to maximize the chi-square values of a 2 X 2 frequency table.

    - **R and Chi Square** — Both the R-Square and Chi-Square selection criteria are applied, depending on the target measurement level. If the target is an interval target, then only the R-Square criterion is applied. If the target is a class target, both the R-Square and the Chi-Square criteria are applied.

    - **None** — the model has no target associated with it. Enterprise Miner still applies the settings that you specified for the Max Class Level and Max Missing Percentage properties if you set the Target Model property to None.

- **Manual Selector** — Select the ... button to the right of the Manual Selector property to open a Manual Variable Selector window. The Manual Variable Selector window holds a table of the input variables that were submitted to the Variable Selection node. You can use the Manual Variable Selector window to individually select and reject variables. Click in the Role column for individual variables to choose between **Input** and **Rejected** variable status.

- **Rejects Unused Input** — Use the Rejects Unused Input property to specify whether to reject unused input variables in successor nodes.

Variable Selection Node Train Properties: Bypass Options

Use the Bypass Options to specify properties about variables that you want to bypass from the variable selection process. You configure whether bypassed variables have a role of input or rejected.

- **Variable** — Use the Variable bypass option to specify a type of variable that you want to bypass the Variable Selection process:

    - **None** — No variables in the submitted data bypass the Variable Selection process. None is the default setting.

    - **Interval** — All interval variables in the submitted data bypass the variable selection process.

    - **Class** — All class variables in the submitted data bypass the variable selection process.

- **Role** — Use the Role option to specify the role that bypassed variables will have when passed to successor nodes. The Role property is dimmed and unavailable unless the Variable property is set to Interval or Class.

- **Input** — Bypassed variables will have an Input role.
- **Rejected** — Bypassed variables will have a Rejected role.

Variable Selection Node Train Properties: Chi-Square Options

- **Number of Bins** — Use the Number of Bins property to specify the number of strata in which the range of a numeric interval variable is divided for splits. The Number of Bins property works only with the Chi-Square selection criterion. Permissible values are positive nonzero integers. The default value is 50.
- **Maximum Pass Number** — The Maximum Pass Number property specifies an upper bound for the number of passes that are made through the input data set to determine the optimum number of binary splits. Permissible values are positive nonzero integers; the default value is 6. Increasing the number of bins tends to provide more accurate results, but the run time and memory is increased. Decreasing the number of bins tends to provide less optimal results, but the run time and memory used is decreased.
- **Minimum Chi-Square** — use the Minimum Chi-Square property to specify the lower bound of the chi-square value in which a variable is still eligible for selection. The value must be greater than 0 and the default value is **3.84**. As you increase the chi-square value, the procedure performs fewer splits. The chi-square value represents the x-axis value that corresponds to the probability of the chi-Square distribution. Values that are greater than the minimum chi-square value can indicate that the relationship between the target value and the variable is not random. For the value **3.84**, P(chi-square statistic > 3.84) = 0.05, so at the significance level p < 0.05, the evidence of the data indicates that there is a non-random relationship between the target variable and the value. The following table lists some of the more common significance levels and their corresponding chi-square statistics:

Probability	X-axis Value
0.01	6.635
0.05	3.841
0.10	2.706

Variable Selection Node Train Properties: R-Square Options

- **Maximum Variable Number** — Use the Maximum Variable Number property to specify the upper bound for the number of variables that are selected for the model. This is an upper bound for the number of rows and columns of the X'X matrix of the regression problem. Permissible values are integers greater than 2. The default setting is 3000.
- **Minimum R-Square** — Use the Minimum R-Square property to specify the lower bound for the individual R-square value of a variable in order to be eligible for the model selection process. Permissible values range from 0 to 1. The default value is 0.005.
- **Stop R-Square** — Use the Stop R-Square property to specify the incremental model R-square value lower bound value. When the lower bound value is reached, the variable selection process stops. Permissible values range from 0 to 1. The default value is 0.0005.

- **Use AOV16 Variables** — Use the Use AOV16 Variables property to indicate if you want to bin interval variables into 16 equally-spaced groups. The AOV16 variables are created to help identify non-linear relationships with the target. The default setting is **No**.

  The following example explains how the AOV16 variables identify non-linear relationships with the target. Suppose that you have some data where Y represents income and X is age. The age variable has 5 levels with replicate income measurements at each age level. You could perform a simple linear regression using the following code:

  ```
  PROC GLM
     data=WORK.INCOME;
     model Y = X;
  run;
  ```

  In this case the covariate (regressor) X would support 1 degree of freedom. You could also run the following model:

  ```
  PROC GLM
     data=WORK.INCOME;
     class X;
     model Y = X;
  run;
  ```

  In this case, X would support 4 degrees of freedom and the source of variation explained by X would be equivalent to a fourth degree polynomial model:

  ```
  PROC GLM
     data=WORK.INCOME;
     model Y = X X*X  X*X*X  X*X*X*X;
  run;
  ```

  Thus the AOV16, which can account for at most 15 degrees of freedom, can be thought of as a way of explaining a single degree covariate in a nonlinear fashion. Since the covariate (input) typically has more than 16 levels, the above regression won't work out exactly but the spirit of the idea is the same.

- **Use Group Variables** — Use the Use Group Variables property to indicate whether you want to reduce class variables to group variables. The levels of the class variables are reduced based on the relationship of the variable to the target. If a class variable can be reduced to a group variable with fewer levels, then only the group version of the variable is used in the model. The default setting is **Yes**.

  The group class variables method can be viewed as an analysis of the ordered cell means to determine if specific levels of a class input can be collapsed into a single group. For example, suppose that you have a four level class input that explains 20% of the total variation in the target (R-square value = .20). The node first orders the input levels by the mean responses. The ranking determines the order that the node considers the input levels for grouping. The node always works from top to bottom when considering the levels that can be collapsed.

  Table 40.1 Ordered Mean Response for a Four Level Input

Target Mean	Input Level
90	C
85	B

Target Mean	Input Level
50	D
42	A

The node combines the first two levels (C and B) into a single group if the reduction in the explained variation is less than or equal to 5% of the target variable (R-square value of at least 19). If the C and B levels can be combined into a group, then the node determines if CB and D can be collapsed. If C and B cannot be combined, then the node determines if B and D can be combined into one group. The node stops combining levels when the R-square threshold is not met for all possible ordered, adjacent levels or when it reduces the original variable to two groups.

- **Use Interactions** — Use the Use Interactions property to indicate whether you want to include all possible two-way interactions of class variables in the variable selection process. A two-way interaction measures the effect of a classification input variable across the levels of another classification input variable. Setting the Use Interactions property to **Yes** will also set the Use Group Variables property to **Yes**. The default setting for the Use Interactions property is **No**.
- **Use SPDE Library** — Use this property to indicate whether you want to use the multithreading processing capabilities provided by SAS Scalable Performance Data Engine. The default setting is **Yes**.
- **Print Option** — Select a print option from **Default**, **All**, and **None**. **Default** suppresses some details of outputs, such as interactions. **None** suppresses all outputs.

Variable Selection Node Score Properties

The following score properties are associated with the Variable Selection node:

- **Hides Rejected Variables** — Set the Hides Rejected Variables property to No if you want to keep the original variables in data sets passed to successor nodes. The default setting is Yes. When set to Yes, the original variables will be removed from the exported metadata that is sent to successor nodes, but the original variables are not removed from the exported data sets and data views.
- **Hides Unused Variables** — Set the Hides Unused Variables property to No if you want to keep unused original variables in the transformed data sets passed to successor nodes. If a variable's role is set as Input and the variable's status is set to Don't Use, the variable will not be used for variable selection at all. When Reject Unused Variables is set to Yes, then the variable will be rejected in successor nodes. The default setting for the Reject Unused Variables property is Yes.

Variable Selection Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Variable Selection Node Results

Open the Results window of the Variable Selection node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Variable Selection node Properties panel. The information was captured when the node was last run.
  - **Run Status** — displays the status of the Variable Selection node run. Information about the run start time, run duration, and completion status are displayed in this window.
  - **Variables** — displays a read-only table of variable meta information on the data set submitted to the Variable Selection node. The table includes columns for the variable name, variable role, variable level, and the model used.
  - **Train Code** — displays the code that Enterprise Miner used to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — the SAS log of the Variable Selection node run.
  - **Output** — the SAS output of the Variable Selection run. Typical output includes information such as:
    - Variable Summary by Role
    - Level and Count
    - R-Squares
    - Chosen Effects
    - Analysis of Variance (ANOVA) tables for the target variable
    - Estimating Logic and Cutoff Classification tables
    - Node Split History
    - Split Effect Summary
    - Summary of Variable Selection

    The "Examine the Results" on page 606 section of the Variable Selection example contains additional information on the contents of the SAS Output window.
  - **Flow Code** — the SAS code used to produce the output that the Variable Selection node passes on to the next node in the process flow diagram. The Flow

Code code menu item is typically dimmed and unavailable in the Variable Selection Results window menu.

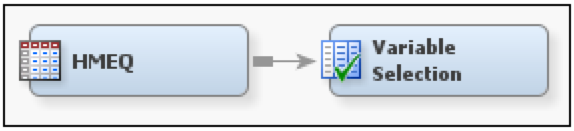
- **Scoring**
  - **SAS Code** — the SAS code that was created by the node. The SAS code can be used outside of the Enterprise Miner environment in custom user applications. The SAS code menu item is typically dimmed and unavailable in the Variable Selection Results menu.
  - **PMML Code** — the Variable Selection node does not generate PMML code.
- **Variable Tables**
  - **Variable Selection** — displays a read-only table reporting the variable names along with their role, level, type, labels and comments.
  - **Group Variables** — displays a read-only table of the group's name, group, variable and level.
  - **AOV16 Variables** — displays a read-only table of the AOV16 name, group ,variable and bin cutoff.
  - **Group Interactions** — displays a read-only table of the group interaction's name, group, variable 1, level 1, variable 2 and level 2.
- **R-Square: Plots**
  - **R2 Values** — displays a bar chart and table of the R-square values for each variable in the selection model.
  - **Effects in the Model** — displays a bar chart and table of the effects that remain in the model after selection. The R-square values for all effects are shown.
- **Chi-Square: Tree Views**
  - **Tree** — displays a map of the decision tree created by the chi-square target model.
  - **Tree Map** — displays a tree map of the decision tree created by the chi-square target model.
  - **Leaf Statistics** — displays a table of the leaf statistics belonging to the decision tree created by the chi-square target model.
  - **Variable Importance** — displays a histogram of input variables, scaled by their relative importance as predictors for the target variable.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph wizard to modify an existing Results plot or create a Results plot of your own. The Graph wizard menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Variable Selection Node Example

Create the Example

The following brief example provides an overview of using the Variable Selection node. You can use any Enterprise Miner project for this example. This example uses a project called Variable Selection.

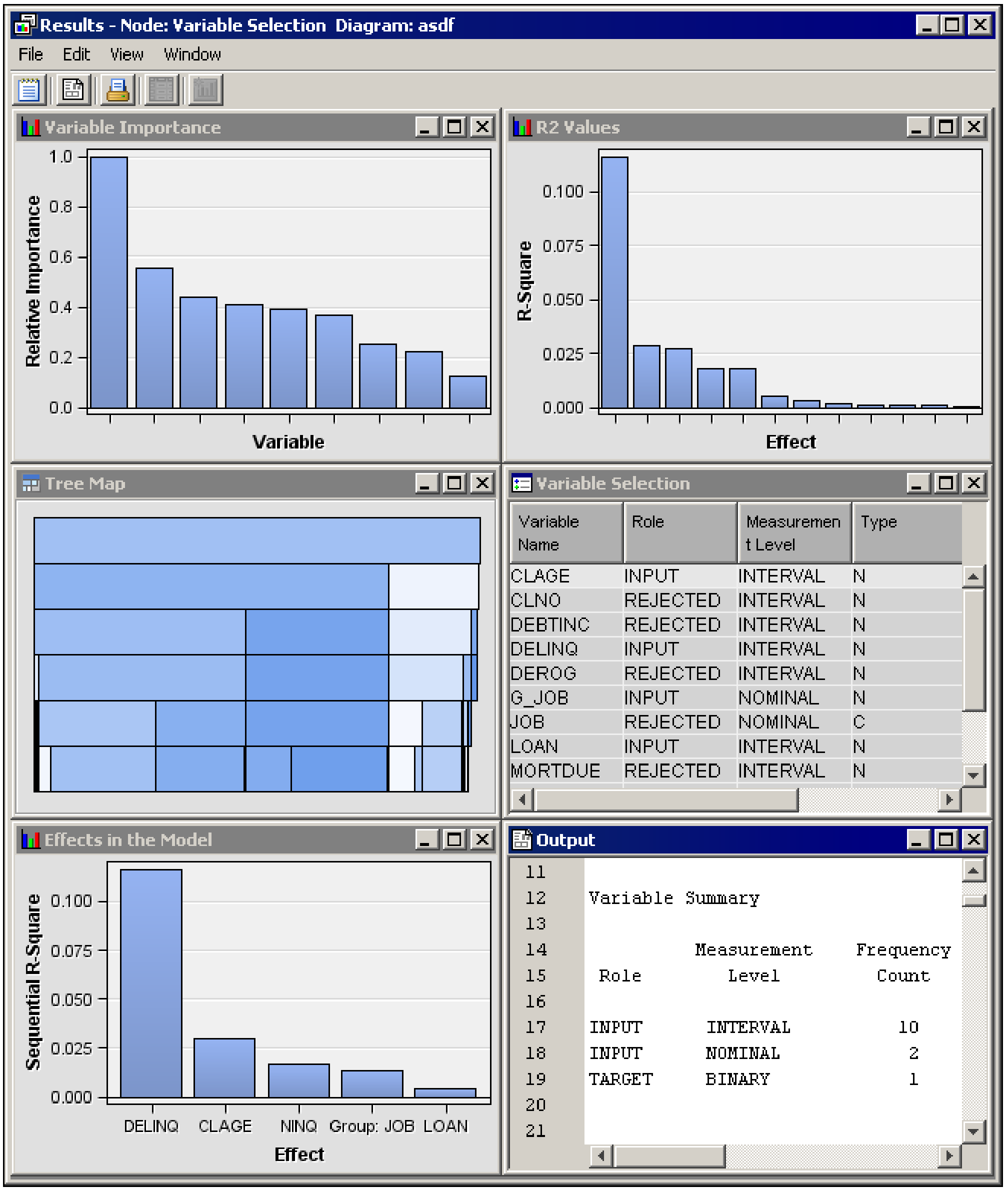
1. Create the Data Source — This example uses the sample SAS data set called SAMPSIO.HMEQ. You must use the data set to create an Enterprise Miner Data Source. Right-click the **Data Sources** folder in the Project Navigator and select **Create Data Source** to launch the Data Source wizard.
   - Choose SAS Table as your metadata source and click **Next**.
   - Enter `Sampsio.Hmeq` in the Table field and click **Next**.
   - Continue to the Metadata Advisor step and choose the **Basic Metadata Advisor**.
   - In the Column Metadata window, set the role of the variable Bad to **Target** and set the level of the variable Bad to **Binary**. Click **Next**.
   - There is no decision processing. Click **Next**.
   - In the Create Sample window you are asked if you want to create a sample data set. Select **No**. Click **Next**.
   - Set the role of the Hmeq data set to **Train**, and then click **Finish**.
2. Place the Nodes on the Diagram Workspace — Drag the Hmeq data source that you just created from the **Data Sources** folder of the Project Navigator onto the Diagram Workspace. Next, drag a Variable Selection node from the **Explore** folder onto the Diagram Workspace and connect the two nodes:



3. Configure the Variable Selection Node — Click the Variable Selection node in the Diagram Workspace to select it. Then go to the Properties Panel for the Variable Selection node and make the following configuration settings:
   - **Variable Selection Train Properties**:
     - Set the Max Missing Percentage property to **10**.
     - Set the Target Model property to **R and Chi Square**.
4. Run the Variable Selection Node — Right-click the Variable Selection node in the Diagram Workspace and select **Run**.
5. Open the Variable Selection Results Window — After the Variable Selection node successfully runs, select **Results**.

Examine the Results

The Results — Variable Selection window displays a ranked plot of the Variable Importance values, a ranked plot of the R2 effect values, a tree map, the Variable Selection table, ranked Model Effects, and the SAS output. Below, you will find a description of each element of the output.



Variable Name	Role	Measurement Level	Type
CLAGE	INPUT	INTERVAL	N
CLNO	REJECTED	INTERVAL	N
DEBTINC	REJECTED	INTERVAL	N
DELINQ	INPUT	INTERVAL	N
DEROG	REJECTED	INTERVAL	N
G_JOB	INPUT	NOMINAL	N
JOB	REJECTED	NOMINAL	C
LOAN	INPUT	INTERVAL	N
MORTDUE	REJECTED	INTERVAL	N

```
Variable Summary

             Measurement      Frequency
 Role           Level           Count

INPUT        INTERVAL             10
INPUT        NOMINAL               2
TARGET       BINARY                1
```

- Variable Importance Plot — The Variable Importance window shows a histogram of the relative importance, or overall contribution of the variables toward making a prediction based on chi-squared scores. Hover your mouse pointer over a bar to view the relative importance score.

  The relative importance of an input variable υ in subtree T is computed as

  $$I(v;T) \propto \sqrt{\sum_{\tau \in T} a(s_v,\tau)\Delta SSE(\tau)}$$

  where the sum is over nodes τ in T, and sυ denotes the primary or surrogate splitting rule using υ. The function $a(s_\upsilon,\tau)$ is the measure of agreement for the rule using υ in node τ:

  $$a(s_v,\tau) = \begin{cases} 1 & \text{if } s_v \text{ is the primary splitting rule} \\ agreement & \text{if } s_v \text{ is a surrogate rule} \\ 0 & \text{otherwise} \end{cases}$$

ΔSSE(τ) is the reduction in sum of square errors from the predicted values:

$$\Delta SSE(\tau) = SSE(\tau) - \sum_{b \in B(\tau)} SSE(\tau_b)$$

$$SSE(\tau) = \begin{cases} \sum_{i=1}^{N(\tau)} (Y_i - \hat{Y}(\tau))^2 & \text{for interval target } Y \\ \sum_{i=1}^{N(\tau)} \sum_{j=1}^{J} (\delta_{ij} - \hat{p}_j(\tau))^2 & \text{for target with } J \text{ categories} \end{cases}$$

where

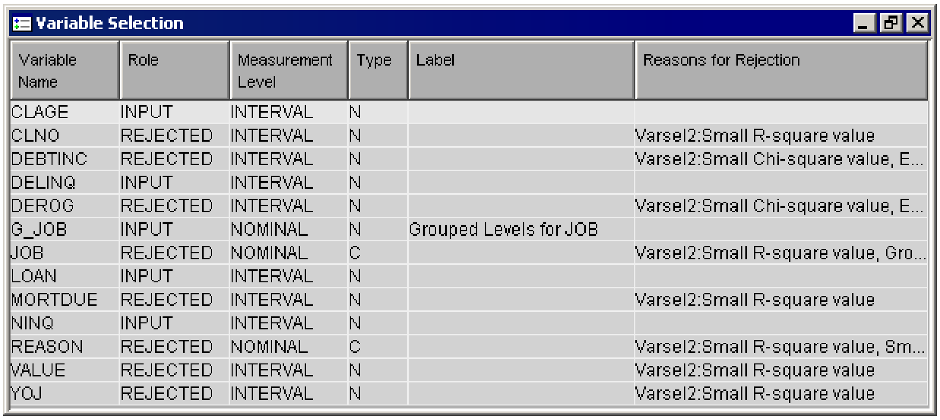
$$\begin{aligned} B(\tau) &= \text{set of branches from } \tau \\ \tau_b &= \text{child node of } \tau \text{ in branch } b \\ N(\tau) &= \text{number of observations in } \tau \\ \hat{Y}(\tau) &= \text{average } Y \text{ in training data in } \tau \\ \delta_{ij} &= 1 \text{ if } Y_i = j, 0 \text{ otherwise} \\ \hat{p}_j(\tau) &= \text{average } \delta_{ij} \text{ in training data in } \tau \end{aligned}$$

where pj is the proportion of the validation data with target value j, and N, pj, and pj-hat are evaluated in node τ.

The relative importance is large for a variable used in one very effective split, and for variables used in very many splits of mediocre worth. Input variables that are assigned the input role are used as model inputs by the successor modeling node. By default, the node assigns the input model role to those variables that have a value of relative importance greater than or equal to 0.05. All other variables are assigned the rejected role. Rejected variables are passed to the subsequent modeling nodes, but they are not used as model inputs.

- R2 Values Plot — The R2 Values window displays a histogram with ranked variable effects. The ranking uses the variable R-square values, sorted from highest to lowest.
- Tree Map — The Tree Map displays a tree map of the decision tree that was created by the chi-square target model. Hover your mouse pointer over graphical elements to display summary details on variable composition and counts.
- Variable Selection Table — The Variable Selection summary window shows that eight inputs were rejected:
  - CLNO, JOB, MORTDUE, REASON, VALUE, and YOJ were rejected due to small R-square values.
  - DEBTINC, DEROG, and REASON were rejected due to small chi-square values.

You can sort the table by clicking on various column headers. Successive clicks toggle between ascending and descending column sorts.



Variable Selection

Variable Name	Role	Measurement Level	Type	Label	Reasons for Rejection
CLAGE	INPUT	INTERVAL	N		
CLNO	REJECTED	INTERVAL	N		Varsel2:Small R-square value
DEBTINC	REJECTED	INTERVAL	N		Varsel2:Small Chi-square value, E...
DELINQ	INPUT	INTERVAL	N		
DEROG	REJECTED	INTERVAL	N		Varsel2:Small Chi-square value, E...
G_JOB	INPUT	NOMINAL	N	Grouped Levels for JOB	
JOB	REJECTED	NOMINAL	C		Varsel2:Small R-square value, Gro...
LOAN	INPUT	INTERVAL	N		
MORTDUE	REJECTED	INTERVAL	N		Varsel2:Small R-square value
NINQ	INPUT	INTERVAL	N		
REASON	REJECTED	NOMINAL	C		Varsel2:Small R-square value, Sm...
VALUE	REJECTED	INTERVAL	N		Varsel2:Small R-square value
YOJ	REJECTED	INTERVAL	N		Varsel2:Small R-square value

- Effects in the Model Plot — The Effects in the Model plot displays a histogram with ranked model effects. The ranking uses the sequential R-squared values, sorted from highest to lowest. Hover your mouse pointer on a bar to view histogram statistics.
- Output Window — The Output window of the Variable Selection node Results provides detailed summary information about the Variable Selection run. The following Output window examples show portions of the log from the example run:
  - Variable Summary — The Variable Summary of the Variable Selection Output window breaks down the input variables by Role, Level, and Count.
  - R-Squares for Target Variable BAD — The R-Squares for Target Variable BAD table of the Variable Selection Output window had columns for the Effect, Degrees of Freedom (DF), and R-square scores. If a variable effect's R-square score is less than the specified R-square minimum, that information is noted in the table.
  - Effects Chosen for Target: BAD — The Effects Chosen for Target: BAD table of the Variable Selection Output window has columns for the effects that were selected as target variable predictors during the Variable Selection node run. The table lists the Effect name, Degrees of Freedom (DF), the R-square, F, p-values, Sum of Squares, and Error Mean Square associated with the chosen effects.
  - Analysis of Variance Table for Target: BAD — The Final Analysis of Variance (ANOVA) table for the target variable BAD of the Variable Selection Output window has columns for the degrees of freedom (DF), R-square values, and Sum of Squares scores for the model.
  - Estimating Logistic — The Estimating Logistic table of the Variable Selection Output window lists alpha and beta estimators for early iterations.
  - Classification Table for Cutoff — The Classification table for Cutoff = xxx of the Variable Selection Output window shows observed vs. predicted values for the target variable BAD using the training data and the displayed cutoff criteria. In this example, the cutoff was 0.500 an the predictions were 82.68% accurate compared to the training data.
  - History of Node Splits — The History of Node Splits table of the Variable Selection Output window provides detailed information about the splitting tree on each splitting iteration. For each split iteration, the table provides the number of the splitting node, the parent to the splitting node, the chi-square score, the name of the splitting variable, the value of the splitting variable at the split, and where pertinent, the levels of the splitting variable.
  - Effect Summary — The Effect Summary table of the Variable Selection Output window lists the variable effects in descending order. The table has columns for Effect, Node 1st Split, and Total Times split.
  - Summary of Variable Selection — The Summary of Variable Selection table of the Variable Selection Output window encapsulates the actions taken during the node run. It provides a summary that includes the target name and level, the selection model that was used, and the number of used, unused, and rejected input variables.

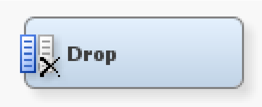
Part 12

Node Reference: Modify Nodes

Chapter 41

Drop Node

Drop Node



Overview of the Drop Node

In the Enterprise Miner SEMMA data mining methodology, the Drop node belongs to the Modify category. You use the Drop node to remove variables from data sets or hide variables from the metadata. You can remove all variables with the role type that you specify in the Drop node Properties Panel, or you can manually specify individual variables to drop using the "Drop Node Variables Table" on page 616 . For example, you could remove all hidden, rejected, and residual variables from your exported data set, or just a few variables that you identify yourself.

Drop Node Input Data Requirements

The node should follow one or more nodes that export a raw or train data set. The Drop node can drop variables from data sets that have roles of Train, Validate, Test, Score, or Transaction.

Drop Node Properties

Drop Node General Properties

The following general properties are associated with the Variable Selection node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Drop node that is added to a diagram will have a Node ID of Drop. The second Drop node added to a diagram will have a Node ID of Drop2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Drop window. The Imported Data — Drop window contains a list of the ports that provide data sources to the Drop node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Drop window. The Exported Data — Drop window contains a list of the output data ports that the Drop node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Drop Node Train Properties

The following train properties are associated with the Drop node:

- **Variables** — Use the Variables window to view variable information, and indicate whether to drop a variable using the Drop node. Select the ... button to open a window containing the variables table. You can specify whether to drop a variable in the Drop column, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. The Variables window allows you to research information about a variable, and decide whether to drop it. You can use the following buttons to accomplish these tasks:

- **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
- **Reset** — Changes metadata back to its state before use of the **Apply** button.
- **Label** — Adds a column for a label for each variable.
- **Mining** — Adds columns for the Order, Report, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
- **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
- **Statistics** — Adds statistics metadata for each variable.
- **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

Drop Node Train Properties: Drop Selection Options

- **Drop from Tables** — Use this property to specify whether the selected variables should be dropped from the created tables (in addition to dropping them from the metadata). When the Drop from Tables property is set to **Yes**, the output training data, drop_train, is expressed as a data set. When the property remains in its default state of **No**, the output training data is expressed as a view. For more information, see the section on the "Drop Node Variables Table" on page 616 .
- **Assess** — use the Assess property to drop or keep variables with a role of Assess from the scored data set. Select **Yes** to drop the variables with a role of Assess; select **No** to keep the variables with a role of Assess. The default setting for the Assess property is **No**.
- **Classification** — use the Classification property to drop or keep variables with a role of Classification. Select **Yes** to drop the variables with a role of Classification; select **No** to keep the variables with a role of Classification. The default setting for the Classification variable is **No**.
- **Frequency** — use the Frequency property to drop or keep variables with a role of Frequency. Select **Yes** to drop the variables with a role of Frequency; select **No** to keep the variables with a role of Frequency. The default setting for the Classification variable is **No**.
- **Hidden** — use the Hidden property to drop or keep variables that appear in the training data set, but do not appear in the metadata. Select **Yes** to drop the variables with a role of Hidden; select **No** to keep the variables with a role of Hidden. The default setting for the Hidden variable is **Yes**.
- **Input** — use the Input property to drop or keep variables with a role of Input. Select **Yes** to drop the variables with a role of Input; select **No** to keep the variables with a role of Input. The default setting for the Input variable is **No**.
- **Predict** — use the Predict property to drop or keep variables with a role of Predict. Select **Yes** to drop the variables with a role of Predict; select **No** to keep the variables with a role of Predict. The default setting for the Predict variable is **No**.
- **Rejected** — use the Rejected property to drop or keep variables with a role of Rejected. Select **Yes** to drop the variables with a role of Rejected; select **No** to keep the variables with a role of Rejected. The default setting for the Rejected variable is **Yes**.
- **Residual** — use the Residual property to drop or keep variables with a role of Residual. Select **Yes** to drop the variables with a role of Residual; select **No** to keep

the variables with a role of Residual. The default setting for the Residual variable is **No**.

- **Target** — use the Target property to drop or keep variables with a role of Target. Select **Yes** to drop the variables with a role of Target; select **No** to keep the variables with a role of Target. The default setting for the Target variable is **No**.
- **Other** — use the Other property to drop or keep variables with roles that are not listed above. Select **Yes** to drop the variables with a role of Other; select **No** to keep the variables with a role of Other. The default setting for the Other variable is **No**.

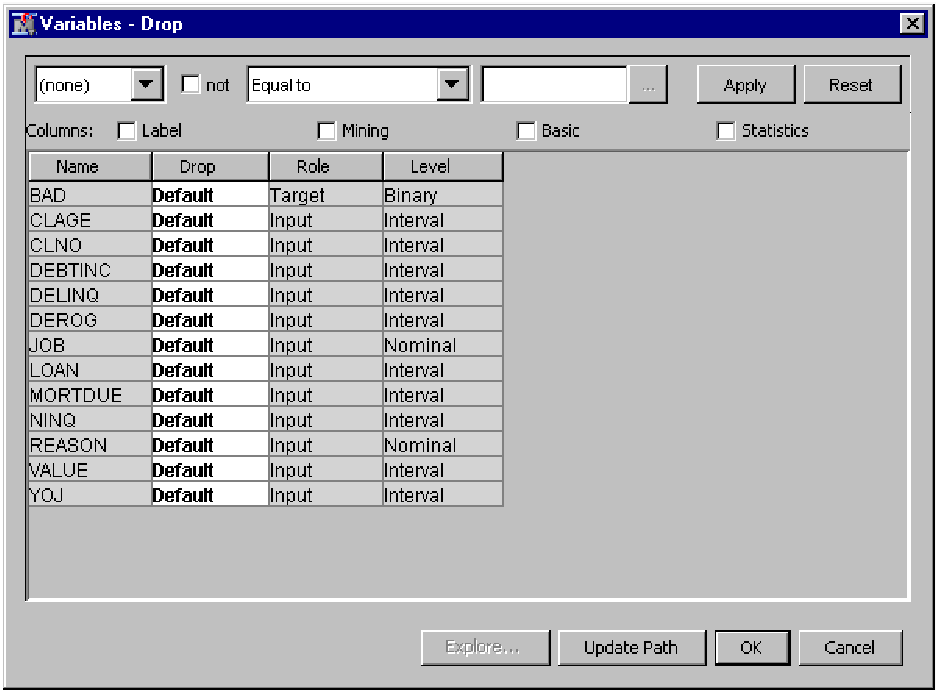
Drop Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Drop Node Variables Table

Use the table in the Drop node Variables window to specify the individual variables in the imported data that you want to drop, to examine the columnsmeta information of the imported data, and through the Explore Variables Window, to explore variable sampling, distribution, and view variable values.



Name	Drop	Role	Level
BAD	Default	Target	Binary
CLAGE	Default	Input	Interval
CLNO	Default	Input	Interval
DEBTINC	Default	Input	Interval
DELINQ	Default	Input	Interval
DEROG	Default	Input	Interval
JOB	Default	Input	Nominal
LOAN	Default	Input	Interval
MORTDUE	Default	Input	Interval
NINQ	Default	Input	Interval
REASON	Default	Input	Nominal
VALUE	Default	Input	Interval
YOJ	Default	Input	Interval

The table in the Variables — Drop window contains the following columns:

- **Name** — displays the name of the variable
- **Drop** — click the cell to specify which individual variables you want to drop. You can choose from:
  - **Default** — (Default) Uses the drop criteria that are specified in the Drop node Properties Panel under the "Drop Node Train Properties: Drop Selection Options" on page 615 .
  - **Yes** — the Drop node drops the variable.
  - **No** — the Drop node retains the variable, overriding any role-base settings specified in the Properties Panel. If the Drop node Properties Panel is configured to drop all residual variables, but you configure No in the Drop column of one of the residual variables, when the node runs, all residual variables are dropped except for the variable that was marked No.
- **Role** — displays the variable role
- **Level** — displays the level for class variables

The Variables — Drop window contains checkboxes that surface additional columns of information when they are selected:

- **Label** — displays a column that contains the text label to be used for variables in graphs and output.
- **Mining** — displays the following columns of data mining metadata for the data set:
  - **Order** — displays the variable order
  - **Report** — indicates whether the variable is included in reports
  - **Lower Limit** — displays the lower limit of values for the variable
  - **Upper Limit** — displays the upper limit of values for the variable
  - **Creator** — displays the user who created the process flow diagram

    - **Comment** — displays the user-supplied comments about a variable
    - **Format Type** — displays the variable format
- **Basic** — displays the following columns of basic Enterprise Miner metadata:
    - **Type** — displays the variable type
    - **Format** — displays the variable format
    - **Informat** — displays the SAS informat for the variable
    - **Length** — displays the variable length
- **Statistics** — displays the following columns of data set statistics:
    - **Number of Levels** — displays the number of levels present in a class variable
    - **Percent Missing** — displays the proportion of observations where the variable value is missing data
    - **Minimum** — displays the minimum value of the variable
    - **Maximum** — displays the maximum value of the variable
    - **Mean** — displays the arithmetic mean of the variable's values
    - **Standard Deviation** — displays the standard deviation of the variable's values
    - **Skewness** — displays the skewness of the variable's values
    - **Kurtosis** — displays the kurtosis measurement of the variable's values

You can also use the expression builder at the top of the Variables — Drop window to create Boolean filters for values contained in any active display column.

Drop Node Results

You can open the Results window of the Drop node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
    - **Settings** — displays a window with a read-only table of the configuration information in the Drop node Properties Panel. The information was captured when the node was last run.
    - **Run Status** — indicates the status of the Drop node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
    - **Variables** — a read-only table of variable meta information on the data set submitted to the Drop node. The table includes columns to see the individual variable drops that were configured in the Variables Table for the Drop node. Columns can be sorted by clicking on column headers. Successive clicks toggle between ascending and descending sorts.
    - **Train Code** — the Drop node does not produce Train code.
    - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
    - **Log** — the SAS log of the Drop node run.

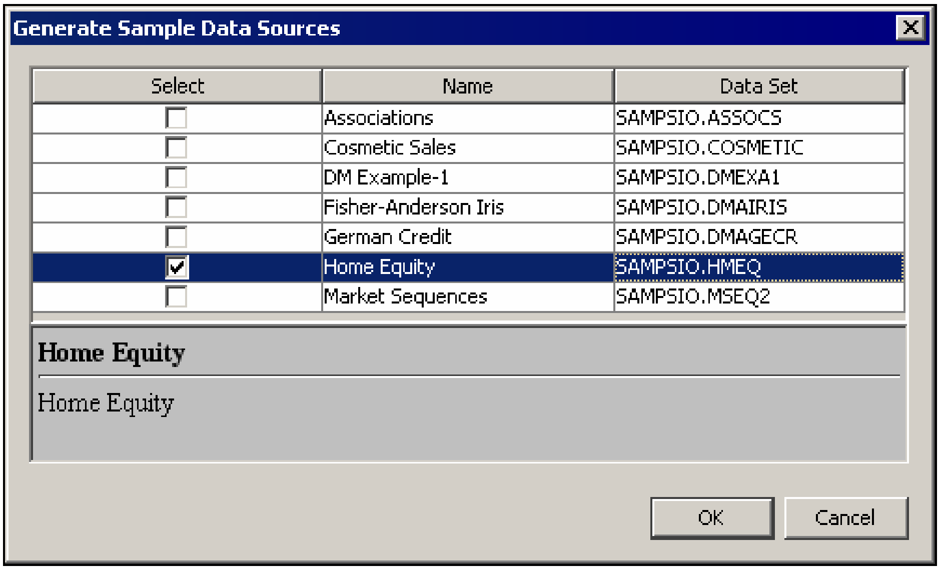
- **Output** — the SAS output of the Drop node run. Typical output includes a Variable Summary and a Dropped Variable Summary.
- **Flow Code** — the Drop node does not produce Flow code.

- **Scoring**
  - **SAS Code** — SAS score code is not available in the Drop node.
  - **PMML Code** — The Drop node does not generate PMML code.

Drop Node Example

The following example builds a process flow diagram that demonstrates a simple usage of the Drop Node:

1. First, you will need to generate the Home Equity data set. To do so, select **Help** ⇨ **Generate Sample Data Sources...** from the menu bar. In the Generate Sample Data Sources window, select only the **Home Equity** data set, as shown below.



Generate Sample Data Sources

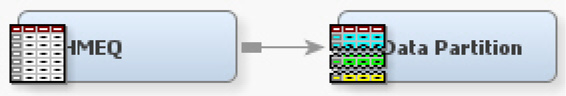
Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
□	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
☑	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Home Equity

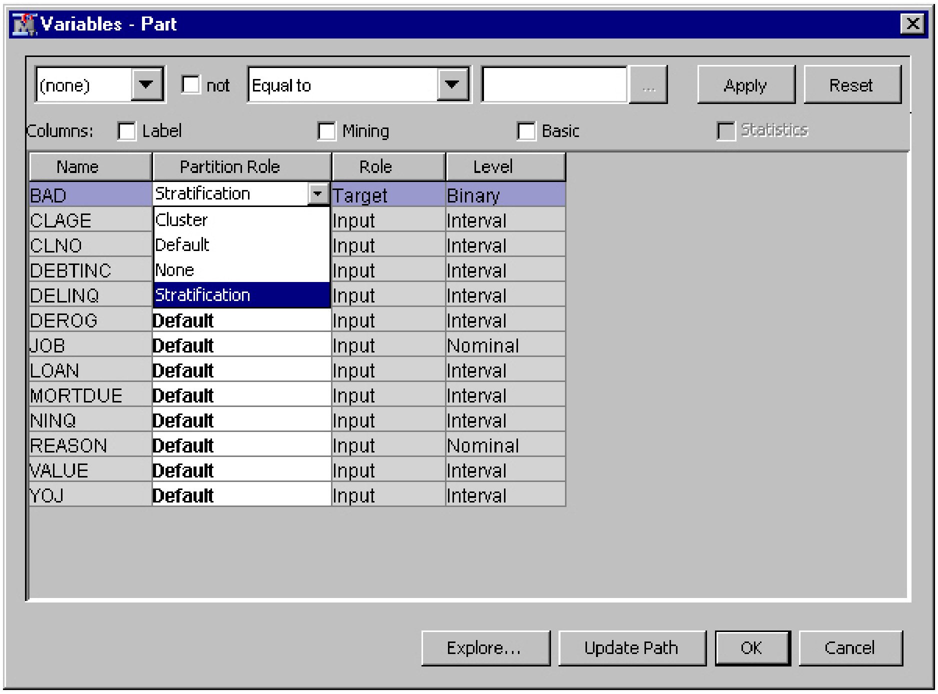
Home Equity

Click **OK**.

2. Drag the newly created SAMPSIO.HMEQ data source onto the diagram workspace. Then drag a Data Partition node icon from the Sample tab of the tools bar onto the diagram workspace, and connect the two nodes.

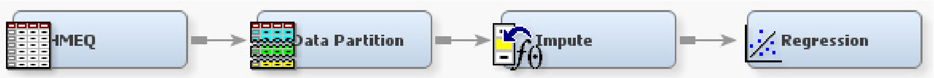


3. Select the Data Partition node and in the Data Partition node Properties Panel, select the ... button to the right of the Variables property to open the variables table. In the variables table, set the BAD variable partitioning as Stratification. Leave all of the other settings in their default positions. Click the **OK** button.



Name	Partition Role	Role	Level
BAD	Stratification ▼	Target	Binary
CLAGE	Cluster	Input	Interval
CLNO	Default	Input	Interval
DEBTINC	None	Input	Interval
DELINQ	Stratification	Input	Interval
DEROG	**Default**	Input	Interval
JOB	**Default**	Input	Nominal
LOAN	**Default**	Input	Interval
MORTDUE	**Default**	Input	Interval
NINQ	**Default**	Input	Interval
REASON	**Default**	Input	Nominal
VALUE	**Default**	Input	Interval
YOJ	**Default**	Input	Interval

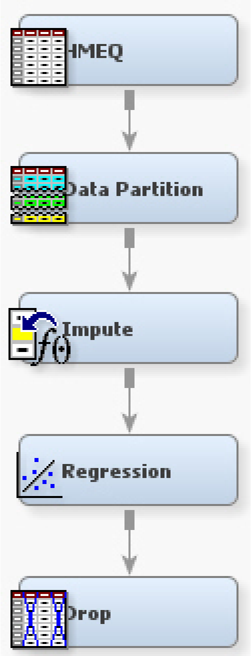
4. Add and connect the Impute node (located in the Modify tools tab) and Regression node (located in the Model tools tab) as shown. Leave both the Regression and Impute nodes in their default settings.



After you add the Regression node, right-click it in the Diagram Workspace and select **Run** from the pop-up menu. Select **Yes** in the Confirmation window. After the diagram execution completes, click **OK** in the confirmation window and continue on to the next step.

5. Add and connect the Drop node (located in the Modify tools tab) to the process flow diagram.

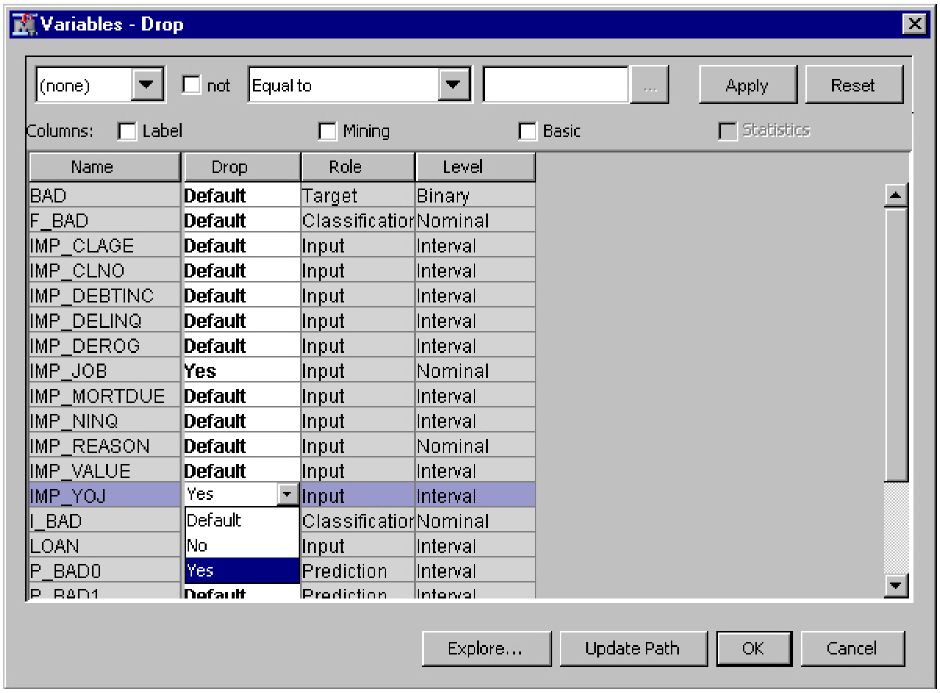
Note: The full process flow diagram shown below uses a vertical layout for improved reader clarity. You do not need to change your process flow diagram orientation in order to match the analytical results of this example.



We want to use the Drop node to remove all of the rejected, hidden and residual variables that the Regression node exports from the analysis. We also want to remove two imputed variables that are related to employment (JOB and YOJ) from the analysis. In the Drop Selection Options section of the Drop node Properties panel, verify that the drop properties for Hidden and Rejected variables are set to **Yes**, and set the Residual property to **Yes**. These settings drop all variables with roles of Hidden, Rejected, and Residual.

Use the Drop node Variables table to drop the two imputed employment variables. Click the ... button to the right of the Variables property in the Properties panel to open the Variables — Drop table.

6. In the variables table for the Drop node, find the imputed variables for JOB and YOJ. They should be, respectively, IMP_JOB and IMP_YOJ.



Name	Drop	Role	Level
BAD	**Default**	Target	Binary
F_BAD	**Default**	Classification	Nominal
IMP_CLAGE	**Default**	Input	Interval
IMP_CLNO	**Default**	Input	Interval
IMP_DEBTINC	**Default**	Input	Interval
IMP_DELINQ	**Default**	Input	Interval
IMP_DEROG	**Default**	Input	Interval
IMP_JOB	**Yes**	Input	Nominal
IMP_MORTDUE	**Default**	Input	Interval
IMP_NINQ	**Default**	Input	Interval
IMP_REASON	**Default**	Input	Nominal
IMP_VALUE	**Default**	Input	Interval
IMP_YOJ	Yes	Input	Interval
I_BAD	Default	Classification	Nominal
LOAN	No	Input	Interval
P_BAD0	Yes	Prediction	Interval
P_BAD1	**Default**	Prediction	Interval

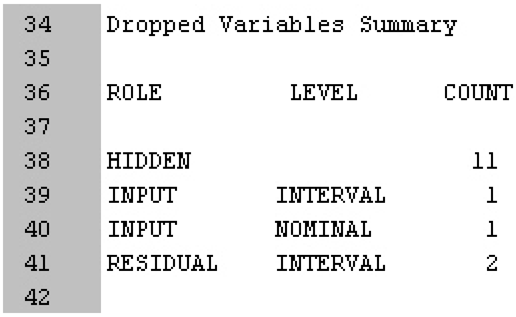
Set the Drop cells for the IMP_JOB and IMP_YOJ variables to **Yes**. These two individual input variables will be dropped along with any variables that have roles of Hidden, Rejected, or Residual. Select **OK** and close the Variables — Drop window.

7. Right-click the Drop node and re-run the entire process flow diagram. After the diagram runs, open the Results window.

8. The Output window displays when the Results window opens. The Variable Summary lists the types of variables that the Drop node found, and displays them sorted by role and level:

```
Variable Summary

                          Measurement    Frequency
Role                         Level         Count

ASSESS                     NOMINAL            1
CLASSIFICATION             NOMINAL            3
ID                         INTERVAL           1
INPUT                      INTERVAL          10
INPUT                      NOMINAL            2
PREDICT                    INTERVAL           2
RESIDUAL                   INTERVAL           2
TARGET                     BINARY             1

```

The Output window also includes a summary of the variables that the node dropped, and displays them sorted by role and level:

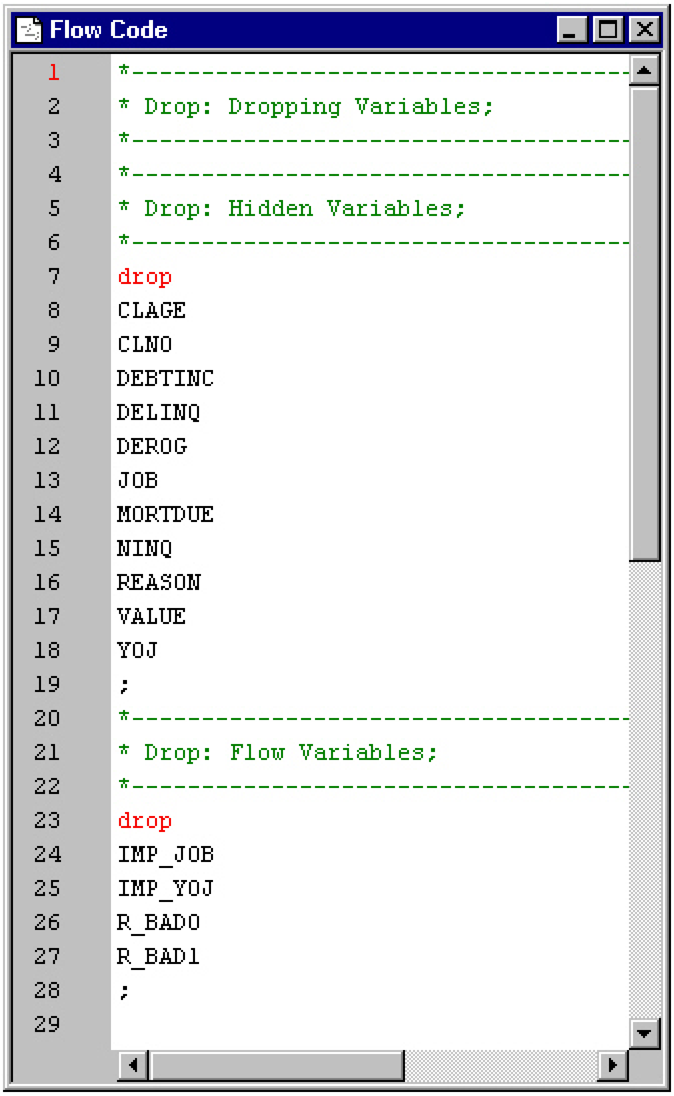


```
Dropped Variables Summary

ROLE          LEVEL      COUNT

HIDDEN                     11
INPUT       INTERVAL        1
INPUT       NOMINAL         1
RESIDUAL    INTERVAL        2

```

The Variable Summary indicates that all 11 hidden variables and both residual variables were dropped. The remaining two input variables that were dropped are the imputed input variables that we specified, IMP_JOB and IMP_YOJ.

9. If you set the **Drop from Tables** property to **Yes** before running the Drop node, you can verify the individual dropped variables by name by examining the Flow Code in the Drop node Results window. To do so, close the current Output window. In the **Drop Selection Options** property group of the Drop Node, set the **Drop from Tables** property to **Yes**. Next, run the process flow diagram again by right-clicking on the Drop Node, selecting **Run** from the pop-up menu, and then selecting **Yes** in the confirmation window.

   Once the process flow diagram has run successfully, select **Results** in the Run Status window. Now, select **View** ⇨ **SAS Results** ⇨ **Flow Clode**.



```
*------------------------------------
* Drop: Dropping Variables;
*------------------------------------
*------------------------------------
* Drop: Hidden Variables;
*------------------------------------
drop
CLAGE
CLNO
DEBTINC
DELINQ
DEROG
JOB
MORTDUE
NINQ
REASON
VALUE
YOJ
;
*------------------------------------
* Drop: Flow Variables;
*------------------------------------
drop
IMP_JOB
IMP_YOJ
R_BAD0
R_BAD1
;

```

You can identify by name each of the 11 hidden variables that were dropped, the 2 residual variables (R_BAD0 and R_BAD1) that were dropped, and the two individual imputed variables that were dropped (IMP_JOB and IMP_YOJ).

Chapter 42
Impute Node

Impute Node

Overview of the Impute Node

Use the Impute node to replace missing values in data sets that are used for data mining. The Impute node is typically used during the modification phase of the Sample, Explore, Modify, Model, and Assess (SEMMA) SAS data mining methodology.

Data mining databases often contain observations that have missing values for one or more variables. Missing values can result from data collection errors, incomplete customer responses, actual system and measurement failures, or from a revision of the data collection scope over time, such as tracking new variables that were not included in the previous data collection schema.

If an observation contains a missing value, then by default that observation is not used for modeling by nodes such as Neural Network, or Regression. However, rejecting all incomplete observations may ignore useful or important information which is still contained in the non-missing variables. Rejecting all incomplete observations may also bias the sample, since observations that missing values may have other things in common as well.

How should missing data values be treated? There is no single correct answer. Choosing the "best" missing value replacement technique inherently requires the researcher to make assumptions about the true (missing) data. For example, researchers often replace a missing value with the mean of the variable. This approach assumes that the variable's data distribution follows a normal population response. Replacing missing values with the mean, median, or another measure of central tendency is simple, but it can greatly affect a variable's sample distribution. You should use these replacement statistics carefully and only when the effect is minimal.

Another imputation technique replaces missing values with the mean of all other responses given by that data source. This assumes that the input from that specific data source conforms to a normal distribution. Another technique studies the data to see if the missing values occur in only a few variables. If those variables are determined to be insignificant, the variables can be rejected from the analysis. The observations can still be used by the modeling nodes.

The Impute node provides the following imputations for missing interval variables:

- Andrew's Wave on page 636
- Default Constant on page 634
- Distribution on page 634
- Huber on page 636
- Mean on page 634
- Median on page 634
- Mid-Minimum Spacing on page 634
- Midrange on page 634
- None on page 634
- Tree on page 636
- Tree Surrogate on page 636
- Tukey's Biweight on page 636

The Impute node provides the following imputations for missing class variables on page 628 :

- Count
- Default Constant
- Distribution
- None
- Tree
- Tree Surrogate

You can customize the default imputation statistics by specifying your own replacement values for missing and non-missing data. Missing values for the training, validation, test, and score data sets are replaced using imputation statistics that are calculated from the active training predecessor data set.

You can use the “Decision Tree Node” on page 739 to define either a decision alternative or surrogate rules in order to group missing values into a special category. You can also use the Cluster node to replace missing values.

Before performing data replacement, you should understand how the SAS System stores missing values. By default, missing numeric values display as periods (.) and missing character values display as blanks ( ). Numeric data can contain characters that represent special missing values (use these when you want to distinguish between types of missing values). The Impute node in SAS Enterprise Miner 12.3 now supports special missing numeric values. To learn more about missing values, see *SAS Language Reference: Concepts*.

The Impute node must follow a node that exports a data set, such as the Input Data, Sample, Data Partition, Variable Selection, Drop, Transform, Filter, Metadata, and Merge nodes.

Impute Node Properties

Impute Node General Properties

The following general properties are associated with the Impute node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Impute node that is added to a diagram will have a Node ID of Impute. The second Impute node added to a diagram will have a Node ID of Impute2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Impute window. The Imported Data — Impute window contains a list of the ports that provide data sources to the Impute node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Impute window. The Exported Data — Impute window contains a list of the output data ports that the Impute node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Impute Node Train Properties

The following train properties are associated with the Impute node:

- **Variables** — specifies the properties of each variable in the data source that you want to use. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can set a variable's report status to **Yes** or **No**.

- **Non Missing Variables** — Use the Non Missing Variables property of the Impute node to specify if you want to impute variables with no missing values. The default setting for the Non Missing Variables property is **No**.
- **Missing Cutoff** — Use the Missing Cutoff property of the Impute node to specify the maximum percent of missing allowed for a variable to be imputed. Variables whose percentage of missing exceeds this cutoff are ignored. The default setting for the Missing Cutoff property is 50 percent.

Impute Node Train Properties: Class Variables

- **Default Input Method** — Use the Default Input Method property of the Impute node to specify the imputation statistic that you want to use to replace missing class variables. For more information on the methods that are available, see the section on "Class Imputation Statistics" on page 636 . The choices are
  - **Count** — Use the Count setting to replace missing class variable values with the most frequently occurring class variable value.
  - **Default Constant Value** — Use the Default Constant setting to replace missing class variable values with the value that you enter in the Default Character Value property.
  - **Distribution** — Use the Distribution setting to replace missing class variable values with replacement values that are calculated based on the random percentiles of the variable's distribution. In this case, the assignment of values is based on the probability distribution of the non-missing observations. The distribution imputation method typically does not change the distribution of the data very much.
  - **Tree** — Use the Tree setting to replace missing class variable values with replacement values that are estimated by analyzing each input as a target. The remaining input and rejected variables are used as predictors. Use the Variables window to edit the status of the input variables. Variables that have a model role of target cannot be used to impute the data. Because the imputed value for each input variable is based on the other input variables, this imputation technique may be more accurate than simply using the variable mean or median to replace the missing tree values.
  - **Tree Surrogate** — Use the Tree Surrogate setting to replace missing class variable values by using the the same algorithm as Tree Imputation, except with the addition of surrogate splitting rules. A surrogate rule is a back-up to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all the surrogates from applying to an observation, the main rule assigns the observation to the branch that is assigned to receive missing values.
  - **None** — Missing class variable values are not imputed under the None setting.
- **Default Target Method** — Use the Default Target Method property of the Impute node to specify the imputation statistic that you want to use to replace missing class target variables. The choices are:
  - **Count** — Use the Count setting to replace missing target variable values with the most frequently occurring target variable value.
  - **Default Constant Value** — Use the Default Constant setting to replace missing target variable values with the value that you enter in the Default Character Value property.
  - **Distribution** — Use the Distribution setting to replace missing target variable values with replacement values that are calculated based on the random

percentiles of the variable's distribution. In this case, the assignment of values is based on the probability distribution of the non-missing observations. The distribution imputation method typically does not change the distribution of the data very much.

  - **None** — Missing target variable values are not imputed under the None setting.

- **Normalize Values** — When the Normalize property is set to **Yes**, the Impute node uses normalized values. The default setting for the Normalize property is **Yes**.

Impute Node Train Properties: Interval Variables

- **Default Input Method** — Use the Method Interval property of the Impute node to specify the imputation statistic that you want to use to replace missing interval variables. For more information on the methods that are available, see the section on "Interval Imputation Methods" on page 634 . The choices are:

  - **Mean** — Use the Mean setting to replace missing interval variable values with the arithmetic average, calculated as the sum of all values divided by the number of observations. The mean is the most common measure of a variable's central tendency; it is an unbiased estimate of the population mean. The mean is the preferred statistic to use to replace missing values if the variable values are at least roughly symmetric (for example, a bell-shaped normal distribution). Mean is the default setting for the Default Input Method for interval variables.

  - **Median** — Use the Mean setting to replace missing interval variable values with the 50th percentile, which is either the middle value or the arithmetic mean of the two middle values for a set of numbers arranged in ascending order. The mean and median are equal for a symmetric distribution. The median is less sensitive to extreme values than the mean or midrange. Therefore, the median is preferable when you want to impute missing values for variables that have skewed distributions. The median is also useful for ordinal data.

  - **Midrange** — Use the Midrange setting to replace missing interval variable values with the maximum value for the variable plus the minimum value for the variable divided by two. The midrange is a rough measure of central tendency that is easy to calculate.

  - **Distribution** — Use the Distribution setting to replace missing interval variable values with replacement values that are calculated based on the random percentiles of the variable's distribution. The assignment of values is based on the probability distribution of the non-missing observations. This imputation method typically does not change the distribution of the data very much.

  - **Tree** — Use the Tree setting to replace missing interval variable values with replacement values that are estimated by analyzing each input as a target. The remaining input and rejected variables are used as predictors. Use the Variables window to edit the status of the input variables. Variables that have a model role of target cannot be used to impute the data. Because the imputed value for each input variable is based on the other input variables, this imputation technique may be more accurate than simply using the variable mean or median to replace the missing tree values.

  - **Tree Surrogate** — Use the Tree Surrogate setting to replace missing interval variable values by using the same algorithm as Tree Imputation, except with the addition of surrogate splitting rules. A surrogate rule is a back-up to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all the surrogates from applying to an observation, the main rule assigns the observation to the branch that is assigned to receive missing values.

  - **Mid-Minimum Spacing** — Use the Mid-Minimum setting to replace missing interval variable values with mid-minimum spacing. The mid-minimum spacing method uses a numeric constant to specify the proportion of the data to be contained in the spacing.
  - **Tukey's Biweight** — Use the Tukey's Biweight setting to replace missing interval variable values with the Tukey's Biweight robust M-estimator value.
  - **Huber** — Use the Huber setting to replace missing interval variable values with the Huber's robust M-estimator value.
  - **Andrew's Wave** — Use the Andrew's Wave setting to replace missing interval variable values with the Andrew's Wave robust M-estimator value.
  - **Default Constant** — Use the Default Constant Value setting to replace missing interval variable values with the value that you enter in the Default Character Value property.
  - **None** — Specify the None setting if you do not want to replace missing interval variable values.

- **Default Target Method** — Use the Default Target Method property of the Impute node to specify the imputation statistic that you want to use to replace missing target variables. The choices are:
  - **Mean** — Use the Mean setting to replace missing target variable values with the arithmetic average, calculated as the sum of all values divided by the number of observations. The mean is the most common measure of a variable's central tendency; it is an unbiased estimate of the population mean. The mean is the preferred statistic to use to replace missing values if the variable values are at least roughly symmetric (for example, a bell-shaped normal distribution). Mean is the default setting for the Default Target Method for interval variables.
  - **Median** — Use the Mean setting to replace missing target variable values with the 50th percentile, which is either the middle value or the arithmetic mean of the two middle values for a set of numbers arranged in ascending order. The mean and median are equal for a symmetric distribution. The median is less sensitive to extreme values than the mean or midrange. Therefore, the median is preferable when you want to impute missing values for variables that have skewed distributions. The median is also useful for ordinal data.
  - **Midrange** — Use the Midrange setting to replace missing target variable values with the maximum value for the variable plus the minimum value for the variable divided by two. The midrange is a rough measure of central tendency that is easy to calculate.
  - **Distribution** — Use the Distribution setting to replace missing target variable values with replacement values that are calculated based on the random percentiles of the variable's distribution. The assignment of values is based on the probability distribution of the non-missing observations. This imputation method typically does not change the distribution of the data very much.
  - **Mid-Minimum Spacing** — Use the Mid-Minimum setting to replace missing target variable values with mid-minimum spacing. The mid-minimum spacing method uses a numeric constant to specify the proportion of the data to be contained in the spacing.
  - **Tukey's Biweight** — Use the Tukey's Biweight setting to replace missing target variable values with the Tukey's Biweight robust M-estimator value.
  - **Huber** — Use the Huber setting to replace missing target variable values with the Huber's robust M-estimator value.

  - **Andrew's Wave** — Use the Andrew's Wave setting to replace missing target variable values with the Andrew's Wave robust M-estimator value.
  - **Default Constant** — Use the Default Constant Value setting to replace missing target variable values with the value that you enter in the Default Character Value property.
  - **None** — Specify the None setting if you do not want to replace missing target variable values.

Impute Node Train Properties: Default Constant Value

Use the Default Constant properties to specify how you want to manage numeric and character constant values to be used during imputation.

- **Default Character Value** — Use the Default Character Value property of the Impute node to specify the character string or value that you want to use during constant character imputation.
- **Default Number Value** — Use the Default Number Value property of the Impute node to specify the numeric value that you want to use as a constant value for numeric imputation. The Default Number Value property defaults to a setting of 0.0.

Impute Node Train Properties: Method Options

- **Random Seed** — Use the Random Seed property of the Impute node to specify the initial Random Seed value that you want to use for random number generation. The default Random Seed value is 12345.
- **Tuning Parameters** — Select the ... button to the right of the Tuning Parameters property to open a window that you can use to customize the tuning parameters for robust methods. The available tuning parameters are
  - **ABW Tuning** — When you choose the Tukey's Biweight imputation statistic as your Method Interval or the Method Target Interval property, you must also use the ABW Tuning property of the Impute node to specify a numeric tuning value. The numeric value is the tuning constant for Tukey's biweight estimator. Permissible values are real numbers greater than or equal to 0. The default value for the Impute node ABW Tuning property is 9.0.
  - **AHUBER Tuning** — When you choose the Huber statistic as your Method Interval or the Method Target Interval property, you must also use the AHUBER Tuning property of the Impute node to specify a numeric tuning value. The value is the tuning constant for Huber's estimator. Permissible values are real numbers greater than or equal to 0. The default value for the Impute node AHUBER Tuning property is 1.5.
  - **AWAVE Tuning** — When you choose the Andrew's Wave statistic as your the Method Interval or the Method Target Interval property, you must also use the AWAVE Tuning property of the Impute node to specify a numeric tuning value. The tuning value is the tuning constant for Andrew's wave estimator. Permissible values are real numbers greater than or equal to 0. The default value for the Impute node AWAVE Tuning property is 6.283185.
  - **Spacing Proportion** — When you choose Spacing as your the Method Target Interval or the Method Interval imputation statistic, use the Spacing Proportion property of the Impute node to specify the numeric constant p, the proportion of the data that you want to be contained in the spacing. Permissible values are real numbers between 0 and 100. The default value for the Impute node Spacing Proportion property is 90%.

- **Tree Imputation** — Select the ... button to the right of the Tree Imputation property to open a window that you can use to customize the tree imputation options. The customizable imputation options are
  - **Leaf Size** — Use the Leaf Size property of the Impute node to specify the minimum number of training observations that are allowed in a leaf node. Permissible values are integers greater than or equal to 1. The default setting is 5.
  - **Maximum Branch** — Use the Maximum Branch property of the Impute node to specify the maximum number of branches that you want a splitting rule to produce. Permissible values for the Maximum Branch property are integers between 2 and 100. The minimum value of 2 results in binary trees. The default value for the Maximum Branch property is 2.
  - **Maximum Depth** — Use the Maximum Depth property of the Impute node to specify the maximum number of generations of nodes that you want to allow in your decision tree. The original node is the root node. Children of the root node are the first generation. Permissible values are integers between 1 and 100. The default number of generations for the Maximum Depth property is 6.
  - **Minimum Categorical Size** — Use the Minimum Categorical Size property of the Impute node to specify the minimum number of training observations that a categorical value must have before the category can be used in a split search. Permissible values are integers greater than or equal to 1. The default value for the Minimum Categorical Size property is 5.
  - **Number of Rules** — Use the Number of Rules property of the Impute node to specify the number of splitting rules that you want to save with each node. The tree uses only one rule. The remaining rules are saved for comparison. Permissible values are integers greater than or equal to 1. The default value for the Number of Rules property is 5.
  - **Number of Surrogate Rules** — Use the Number of Surrogate Rules property of the Impute node to specify the maximum number of surrogate rules that Impute seeks in each non-leaf node. The first surrogate rule is used when the main splitting rule relies on an input whose value is missing. Permissible values are nonnegative integers. The default value for the Number of Surrogate Rules property is 2.
  - **Split Size** — Use the Split Size property of the Impute node to specify the smallest number of training observations that a node must have to before it is eligible to be split. The Split Size property uses a default value of (2*Leaf Size), unless you specify an integer value that is greater than the calculated default value. If no value is specified for the Leaf Size property, then the default Split Size value is calculated as: [2*Min(5000,Max(5,N/1000))]. Permissible values for the Split Size property are integers between 2 and 32767.

Impute Node Score Properties

The following score properties are associated with the Impute node:

- **Hide Original Variables** — Set the Hide Original Variables property of the Impute node to **No** if you want to keep the original variables in the exported metadata from your imputed data set. In that case, the variables are exported with a role of Rejected. The default setting for the Hide Original Variables is **Yes**. When Set to **Yes**, the original variables are only removed from the exported metadata, and not removed from the exported data sets and data views.

Impute Node Score Properties: Indicator Variables

- **Indicator Variable** — Use the Indicator Variable property of the Impute node to indicate whether you want to create indicator variables to flag the imputed observations for each variable.

  You can choose from the following settings:

  - **None** — (default setting) Do not create an indicator variable.
  - **Single** — A single indicator variable is created that indicates one or more variables were imputed.
  - **Unique** — Unique binary indicator variables are created for every imputed variable.

- **Indicator Variable Source** — Use the Indicator Variable Source property of the Impute node to specify the role that you want to assign to the created indicator variables. You can choose between **Missing** and **Imputed**. The default setting is **Imputed**.
- **Indicator Variable Role** — Use the Indicator Variable Role property of the Impute node to specify the role that you want to assign to the created indicator variables. You can choose between the roles **Rejected** and **Input**. The default setting is **Rejected**.

Impute Node Report Properties

The following report properties are associated with the Impute node:

- **Validation and Test Data** — Use the Validation and Test Data property of the Impute node to specify whether to produce a count of the missing values that were replaced for Validation and Test data sets. The default setting is **No**.
- **Distribution of Missing** — Use the Distribution of Missing property of the Impute node to specify whether to produce a distribution of the missing values for the training data. The distribution shows how many observations had 0 variables imputed, 1 variable imputed, 2 variables imputed, and so on.

Impute Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Using Indicator Variables with the Impute Node

Use the Indicator Variables property to create flag variables that identify each variable's imputed observations. Select the check box to create flag variables for each imputed input variable. Flag variables are named by concatenating the prefix M_ with the input variable's name. Each flag variable contains an indicator value of 1 if the observation was imputed and an indicator value of 0 if it was not. For example, assume that you have two input variables named AGE and HEIGHT that contain the following values prior to imputation:

Age	Height
25	73
.	66
30	.

When you run the node, imputed flag variable(s) are created, depending on the setting that is configured for the Indicator Variable property:

Table 42.1 *Indicator Variable Setting = UNIQUE*

Age	Height	M_AGE	M_HEIGHT
25	73	0	0
34	66	1	0
30	62	0	1

Table 42.2 *Indicator Variable Setting = SINGLE*

Age	Height	M_VARIABLE
25	73	0
34	66	1
30	62	1

Imputation Methods

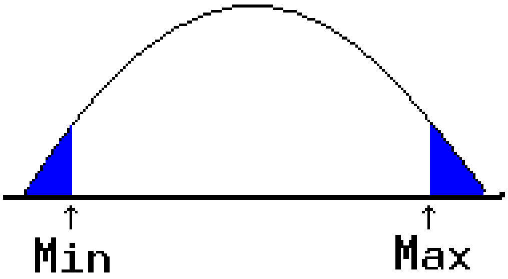
Interval Imputation Methods

To set the default interval imputation statistic, click the Method Interval or Method Target Interval property drop-down arrow and select one of the following imputation methods:

- Mean — (default) or the arithmetic average, calculated as the sum of all values divided by the number of observations. The mean is the most common measure of a variable's central tendency; it is an unbiased estimate of the population mean. The mean is the preferred statistic to use to replace missing values if the variable values are at least roughly symmetric (for example, a bell-shaped normal distribution).

- Median — or the 50th percentile, which is either the middle value or the arithmetic mean of the two middle values for a set of numbers arranged in ascending order.
  - For the set of numbers [3,4,4,5,6,8,8,8,10] the median is 6.
  - For the set of numbers [5,5,7,9,11,12,15,18] the median is ((9+11)/2) =10.

  The mean and median are equal for a symmetric distribution. The median is less sensitive to extreme values than the mean or midrange. Therefore, the median is preferable when you want to impute missing values for variables that have skewed distributions. The median is also useful for ordinal data.
- Mid-range — the maximum plus the minimum divided by two. The midrange is a rough measure of central tendency that is easy to calculate.
- Distribution — replacement values are calculated based on the random percentiles of the variable's distribution. In this case, the assignment of values is based on the probability distribution of the non-missing observations. This imputation method typically does not change the distribution of the data very much.
- Tree Imputation — replacement values are estimated by analyzing each input as a target, and the remaining input and rejected variables are used as predictors. Use the “Impute Node Variables Table” on page 637 to edit the status of the input variables. Variables that have a model role of target cannot be used to impute the data. Because the imputed value for each input variable is based on the other input variables, this imputation technique may be more accurate than simply using the variable mean or median to replace the missing tree values.
- Tree Imputation with Surrogate — the same as Tree Imputation, except with the addition of surrogate splitting rules. A surrogate rule is a back-up to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all the surrogates from applying to an observation, the main rule assigns the observation to the branch that is assigned to receive missing values.
- Mid-Minimum Spacing — the Mid-minimum spacing method uses a numeric constant to specify the proportion of the data to be contained in the spacing. To calculate this statistic
  - The data is trimmed using N percent of the data as specified in the Spacing Proportion property of the Tuning Parameters. By default, 90% of the data is used to trim the original data.
  - The maximum plus the minimum divided by two for the trimmed distribution is equal to the mid-minimum spacing.



- Tukey's Biweight — see “Robust M-Estimators of Location — Tukey's Biweight, Huber, and Andrew's Wave” on page 636 .

- Huber — see “Robust M-Estimators of Location — Tukey's Biweight, Huber, and Andrew's Wave” on page 636 .
- Andrew's Wave — see “Robust M-Estimators of Location — Tukey's Biweight, Huber, and Andrew's Wave” on page 636 .
- Default Constant — all replacements are equal to a fixed constant value. You configure the fixed constant value using the Default Character Value and the Default Number Value properties.
- None — do not impute the missing values.

Robust M-Estimators of Location — Tukey's Biweight, Huber, and Andrew's Wave

Tukey's biweight, Hubers, and Andrew's wave are all robust M-estimators of location. Common estimators such as the sum of squared residuals can become unstable when they use outlier data points and can distort the resulting estimators. M-Estimators try to reduce the effect of outliers by using substitute functions that are symmetric, have a unique minimum at zero, and increase less than standard squared residuals under summation. Many M-estimators exist.

A suitably chosen M-estimator shows robustness of efficiency in larger samples and has resistance to outliers or gross errors in the data.

An estimator has robustness of efficiency over a range of distributions if its variance is close to the minimum for each distribution. Robustness of efficiency guarantees that the estimator is good when repeated samples are drawn from a distribution that is not known precisely. An estimator is resistant if it is not changed much by small groups of outliers or by rounding and grouping errors among observations. Robustness of efficiency and resistance are the main reason why you would want to use one of the M-estimators for imputation.

A complete discussion of M-estimators of location is given in Goodall, C. (1983), "M-Estimators of Location: An Outline of Theory," in Hoaglin, D.C., Mosteller, M. and Tukey, J.W., eds., *Understanding Robust and Exploratory Data Analysis*, New York: Wiley.

The default tuning constant for each M-estimator is as follows:

Estimator	Default Tuning Constant
Tukey's Biweight	9
Huber's	1.5
Andrew's Wave	6.283185

You can enter a different value in the Robust Tuning properties.

Class Imputation Statistics

Missing values for class variables can be replaced with one of the following statistics:

- Count — missing values are replaced with the modal value.
- Default Constant — all replacements are equal to a fixed constant value. You configure the fixed constant value using the Default Character Value and the Default Number Value properties.

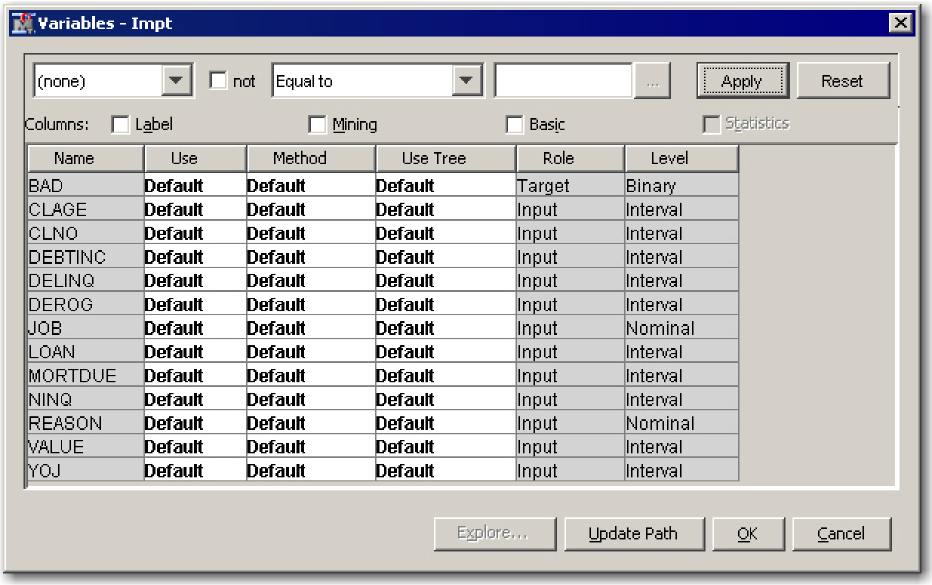
- Distribution — replacement values are calculated based on the random percentiles of the variable's distribution. In this case, the assignment of values is based on the probability distribution of the non-missing observations. This imputation method typically does not change the distribution of the data very much.
- Tree — replacement values are estimated by analyzing each input as a target, and the remaining input and rejected variables are used as predictors. Use the "Impute Node Variables Table" on page 637 to edit the status of the input variables. Variables that have a model role of target cannot be used to impute the data. Because the imputed value for each input variable is based on the other input variables, this imputation technique may be more accurate than simply using the variable mean or median to replace the missing tree values.
- Tree Surrogate — the same as Tree Imputation, except with the addition of surrogate splitting rules. A surrogate rule is a back-up to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all the surrogates from applying to an observation, the main rule assigns the observation to the branch that is assigned to receive missing values.
- None — replacement values are not imputed, and are left as missing.

If several values occur with the same frequency and Count is the class imputation statistic, the smallest value is used as the Count. If the Count is a missing value, then the next most frequently occurring non-missing value is used for data replacement.

Impute Node Variables Table

Use the Variables table of the Impute node to specify which individual variables you want to impute values for, the method that you want to use to impute each variable, and whether to use the variable when tree based imputation is performed. There are also read-only fields in the Variables table that display column meta information such as the variable Role, Level, Type, Order, Label, Format, Informat, Length, Lower and Upper Limits, Distribution, Family, Report, Creator, and Comment.

The default imputation methods that you set in the Class Variables and Interval Variables properties are used if Use is set to Default.



Name	Use	Method	Use Tree	Role	Level
BAD	Default	Default	Default	Target	Binary
CLAGE	Default	Default	Default	Input	Interval
CLNO	Default	Default	Default	Input	Interval
DEBTINC	Default	Default	Default	Input	Interval
DELINQ	Default	Default	Default	Input	Interval
DEROG	Default	Default	Default	Input	Interval
JOB	Default	Default	Default	Input	Nominal
LOAN	Default	Default	Default	Input	Interval
MORTDUE	Default	Default	Default	Input	Interval
NINQ	Default	Default	Default	Input	Interval
REASON	Default	Default	Default	Input	Nominal
VALUE	Default	Default	Default	Input	Interval
YOJ	Default	Default	Default	Input	Interval

The Use column settings are **Yes** (replace missing values using the specified imputation method), **No** (don't replace variable values), or **Default**. All input variables have a default status of Use. By default Target and Rejected variables are ignored.

Click the Method cell of the variable to which you want to assign a new imputation method.

You can select from the following transformation methods:

- **Andrew's Wave**
- **Constant**
- **Count**
- **Default**
- **Distribution**
- **Huber**
- **Mean**
- **Median**
- **Mid-Minimum Spacing**
- **Mid-Range**
- **None**
- **Tree**
- **Tree Surrogate**
- **Tukey's Biweight**

If you select a tree imputation method, then you can set the variables that you want to use in the Use Tree column. To set the Use Tree status, click the cell in the Use Tree column that corresponds to the variable, and select **Default**, **Yes**, or **No**. All input variables are used by default whereas Rejected are ignored.

Impute Node Results

After the Impute node successfully runs, you can open the Results — Impute window by right-clicking the node in the Diagram Workspace and selecting Results from the pop-up menu. The Results — Impute window contains an imputation summary table and a window displaying the node's output. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu of the Impute Results window to view the following information:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Impute node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Impute node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of variables property of the node.
  - **Train Code** — the code that Enterprise Miner used to train the node.

  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Impute node run.
  - **Output** — the SAS output of the Impute node run. The SAS output includes a variable summary, a distribution of missing observations in training data table, and an imputation summary by variable.
  - **Flow Code** — the SAS code used to produce the output that the Impute node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Impute node does not generate PMML code.
- **Imputation Summary** — a table of that shows a summary of the imputation run. The Imputation Summary table displays the variable name, the imputation method, the imputed variable name, the variable role, the variable level, the variable type (numeric or character), the variable label (if any), and the number of missing values for the train data set, the validation data set, and the test data set. The Imputation Summary table is a subset of the SAS output from the Impute node run.
- **Distribution of Incomplete Observations** — a bar chart that shows the distribution of missing values in the training data set. It shows how many observations had 0 variables imputed, 1 variable imputed, 2 variables imputed, and so on. The Distribution of Incomplete Observations menu item will be dimmed and unavailable unless the Impute Report property Distribution of Missing is set to Yes prior to running the node.
- **Table** — open the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart plotting wizard so that you can plot the data in the table that you have in focus.

References

Hoaglin, D.C., Mosteller, M., and Tukey, J.W, ed. 1983. *Understanding Robust and Exploratory Data Analysis*. New York, NY: Wiley.

Chapter 43
Interactive Binning Node

Interactive Binning Node

Overview of the Interactive Binning Node

In the SAS Enterprise Miner SEMMA data mining methodology, the **Interactive Binning** node belongs to the Modify category. The **Interactive Binning** node is an interactive grouping tool that you use to model non-linear functions of multiple modes of continuous distributions. The interactive tool computes initial bins by quantiles. Then you can interactively split and combine the initial bins.

You use the **Interactive Binning** node to create bins or buckets or classes of all input variables, which include both class and interval input variables. You can create bins in order to reduce the number of unique levels as well as attempt to improve the predictive power of each input.

The predictive power of a characteristic, its ability to separate high-risk applicants from low-risk ones, is assessed by its Gini statistic.

- First, sort the data by the descending order of the proportion of events in an attribute. Suppose a characteristic has m attributes. Then, the sorted attributes are expressed as groups 1, 2, ..., m. Each group corresponds to an attribute, and group 1 has the highest proportion of events.
- Then, for each of these sorted groups, calculate the number of events(n_i^{event}) and nonevents (n_i^{event}) in group i. Then, compute the Gini statistic.

$$Gini = \left(1 - \frac{2\sum_{i=2}^{m}(n_i^{event} * \sum_{1}^{i-1} n_i^{nonevent}) + \sum_{i}^{m}(n_i^{event} * n_i^{nonevent})}{N^{event} * N^{nonevent}}\right) * 100$$

Here, N^{event} and $N^{nonevent}$ are the total numbers of events and nonevents in the data, respectively.

Use the **Interactive Binning** node to select strong characteristics based on the Gini statistic and to group the selected characteristics based on business considerations. The node is helpful in shaping the data to represent risk ranking trends rather than modeling quirks, which might lead to overfitting. In some cases, binning attributes also lead to stronger predictive power.

Data Set Requirements for the Interactive Binning Node

The **Interactive Binning** node requires a binary or interval target variable. The target variable must be defined in the SAS Enterprise Miner data source.

Interactive Binning Node Properties

Interactive Binning Node General Properties

The following general properties are associated with the **Interactive Binning** node:

- **Node ID** — displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type.
- **Imported Data** — provides access to the Imported Data — Interactive Binning window. The Imported Data — Interactive Binning window contains a list of the ports that provide data sources to the **Interactive Binning** node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and do the following:

  - Click **Browse** to open a window where you can browse the data set.
  - Click **Explore** to open the Explore window, where you can sample and plot the data.
  - Click **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.
- **Exported Data** — provides access to the Exported Data — Interactive Binning window. The Exported Data — Interactive Binning window contains a list of the output data ports that the **Interactive Binning** node creates data for when it runs. Click the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and do the following:

  - Click **Browse** to open a window where you can browse the data set.
  - Click **Explore** to open the Explore window, where you can sample and plot the data.

  - Click **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Click the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Interactive Binning Node Train Properties

The following train properties are associated with the **Interactive Binning** node:

- **Variables** — specifies the properties of each variable that you want to use in your data source. Click the ... button to the right of the Variables property to open a variables table. In the table, you can set the variable status to either **Use** or **Don't Use**, and you can select which variables you want included in reports.
- **Interactive Binning** — launches the Interactive Binning window. Click the ... button to the right of the Interactive Binning property to open the Interactive Binning window.
- **Treat Missing as Level** — indicates whether missing values should be treated as levels when creating groupings. If this property is set to **Yes**, then a group value will be assigned to missing levels of the variable. If it is set to **No**, then the group value will be set to missing for each variable. The default value is **Yes**.
- **Use Frozen Groupings** — specifies whether frozen groupings should be used or if new groupings should be created during training. When the Use Frozen Groupings property is set to **Yes** and you run the **Interactive Binning** node, automatic grouping is not performed on variables where grouping definitions have already been created. When this property is set to **No**, the **Interactive Binning** node ignores the groupings that were previously defined, and creates new groupings based on the existing settings of the node properties.

Interactive Binning Node Train Properties: Interval Target Options

- **Binary Transformation** — specifies the binary transformation method that is used. You can choose from the following options:
  - **Mean Cutoff** — the mean value of the interval target variable. It is used as the cutoff when the node creates a new binary target variable. If a frequency variable exists, then that variable is used when determining the mean value.
  - **User-Specified Cutoff** — the value that you specify in the **Cutoff Value** property. It is used as the cutoff when the node creates a new binary target variable.
- **Cutoff Value** — specifies the cutoff value that is used when **User-Specified** is selected as the **Binary Transformation** method.

Interactive Binning Node Train Properties: Interval Variable Options

- **Apply Level Rule** — specifies whether the number of unique levels of an interval input should be taken into consideration when determining how the variable should be treated. If the Apply Level Rule property is set to **Yes**, the number of unique levels is compared to the number of levels in the Number of Bins property. If the number of unique levels is less than the number of levels specified in the Number of Bins property, the input variable is treated as an ordinal input. If the number of unique levels is greater than the number of levels specified in the Number of Bins property, the input variable is treated as an interval input. If the Apply Level Rule property is set to **No**, all interval input variables will be treated as interval, regardless of the number of unique levels.

- **Method** — specifies the method to use for pre-binning of interval input variables.

  Choose one of the following methods:

  - **Quantile** — generates groups that are formed by ranked quantities with approximately the same frequency in each group.
  - **Bucket** — generates groups by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
- **Number of Groups** — specifies the number of nonmissing groups to create for the interval input variables.

Interactive Binning Node Train Properties: Class Variable Options

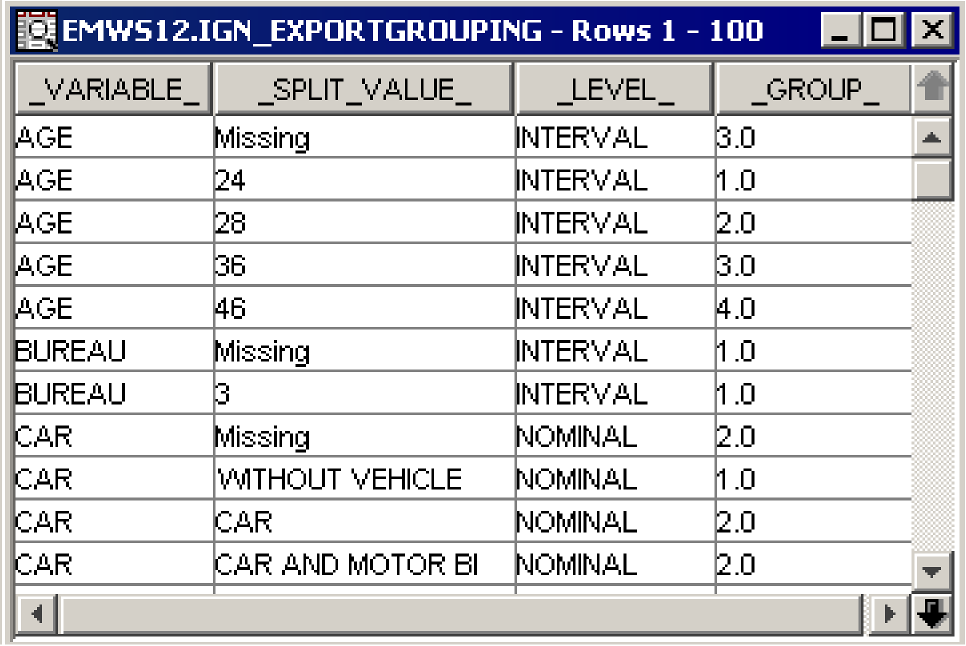
- **Group Rare Levels** — indicates whether rare levels should be grouped together. If this property is set to **Yes**, all values of the class variable that occur less than the cutoff percentage will be placed in the same group. If set to **No**, all values will be assigned a unique group value.
- **Cutoff Value Percentage** — indicates the cutoff percentage used to determine which levels should be grouped together if **Group Rare Levels** is set to **Yes**.

Interactive Binning Node Train Properties: Import/Export Options

- **Import Grouping Data** — specifies whether to import pre-defined grouping information. If this is set to **Yes**, the grouping definitions found in this data set will be used instead of those created by running the node.
- **Import Data Set** — specifies the name of the pre-defined grouping data set when the Import Grouping Data property is set to Yes. Click the ... button to the right of the **Import Data Set** property to open a Select a SAS Table window. You can either import a SAS data set that was generated by another Interactive Binning node, or create your own SAS data set to import. This grouping data set must be in the following format.

Variable Name	Type	Format	Description
VARIABLE	Character	$32	variable name
_SPLIT_VALUE_	Character	$200	the upper bound of the values in a group if the variable is interval; individual values if the variable is categorical (ordinal, nominal, or binary)
LEVEL	Character	$8	measure level of the variable, such as interval, ordinal, nominal, or binary
GROUP	Interval	8	group ID

The following display is an example of the grouping data set that is generated by the **Interactive Binning** node. The name of the data set is EMWS12.IGN_EXPORTGROUPING.



EMWS12.IGN_EXPORTGROUPING - Rows 1 - 100

VARIABLE	_SPLIT_VALUE_	_LEVEL_	_GROUP_
AGE	Missing	INTERVAL	3.0
AGE	24	INTERVAL	1.0
AGE	28	INTERVAL	2.0
AGE	36	INTERVAL	3.0
AGE	46	INTERVAL	4.0
BUREAU	Missing	INTERVAL	1.0
BUREAU	3	INTERVAL	1.0
CAR	Missing	NOMINAL	2.0
CAR	WITHOUT VEHICLE	NOMINAL	1.0
CAR	CAR	NOMINAL	2.0
CAR	CAR AND MOTOR BI	NOMINAL	2.0

In this example, four groups are created for the interval variable AGE, and two groups are created for the categorical variable CAR as follows:

- For AGE, group 1 contains the applicants who are younger than 24.
- For AGE, group 2 contains the applicants who are at least 24 but younger than 28.
- For AGE, group 3 contains the applicants who are at least 28 but younger than 36 and the applicants whose age information is missing.
- For AGE, group 4 contains the applicants who are at least 36 but younger than 46.
- For AGE, group 5 contains the applicants who are at least 46.
- For CAR, group 1 contains the applicants who have no cars.
- For CAR, group 2 contains the applicants who have cars, applicants who have both cars and motorcycles, and applicants for whom information about cars is not available.

Interactive Binning Node Score Properties

The following score properties are associated with the **Interactive Binning** node:

- **Group Level** — specifies the level assigned to the exported Group variables. Possible values are Ordinal (default) and Nominal.
- **Variable Selection Method** — specifies the criteria for variable selection. Possible values are **Gini Statistic** and **None**.
- **Gini Cutoff** — specifies a minimum cutoff value for Gini Statistic. Variables. A Gini statistic greater than the specified cutoff will be selected as input to successor nodes. Otherwise, the variables will have a role of "Rejected." The default setting for the Gini Cutoff property is 20.0.
- **Reject Interval Target** — specifies if the role interval target variable is set to **Rejected** and the transformed binary variable is used in the exported data sets. If you specify **No**, then the original target variable is exported and the binary target variable is set to **Rejected**.

Interactive Binning Node Report Properties

The following report properties are associated with the **Interactive Binning** node:

- **Create Grouping Data** — specifies whether to export the grouping data set.
- **Create Method** — specifies whether to append or to overwrite the saved information of groupings for the grouped variables. You can set this property when the Create Grouping Data property is set to **Yes**.
- **Number of Variables** — specifies the maximum number of variables to be plotted in the Event Rate Plot. Only the variables that exceed the Gini cutoff are plotted.

Interactive Binning Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

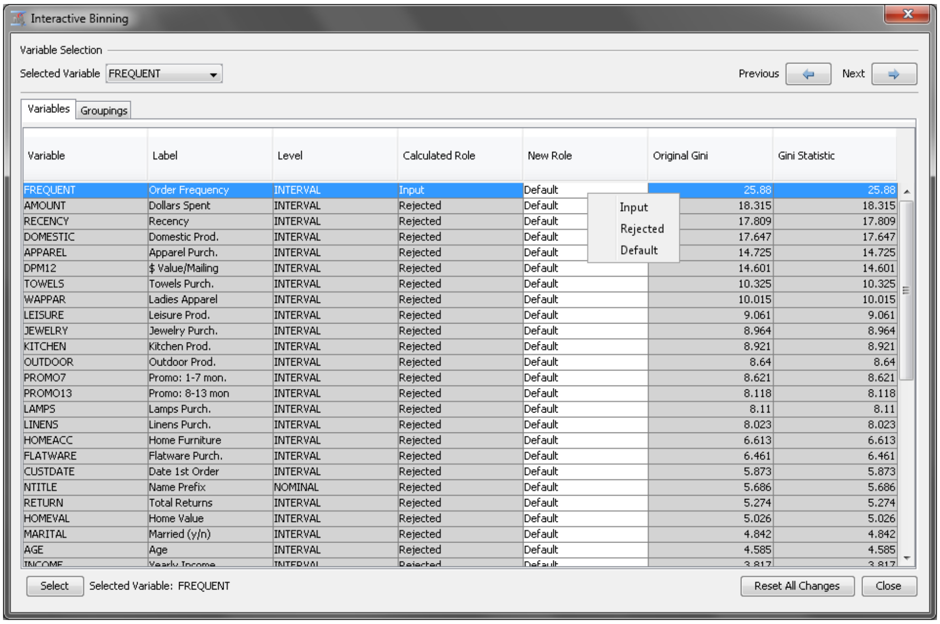
Interactive Binning Application

Overview

To launch the Interactive Binning application, click the ... button to the right of the Interactive Binning property.

Variables Tab

The **Variables** tab displays a table with the variable name, level, role, and Gini statistic. You can select the variable that is plotted in the **Groupings** tab by selecting that variable and clicking the **Select** button in the lower left corner of the window. You can interactively override each variable’s role, either **Input**, **Rejected**, or **Default**, when you perform interactive training as opposed to having to change variable roles in successor nodes. To change a variable’s role, select that variable and right-click anywhere in the **New Role** column. This action opens a drop-down menu that enables you to select the variable’s new role.



Variable Selection

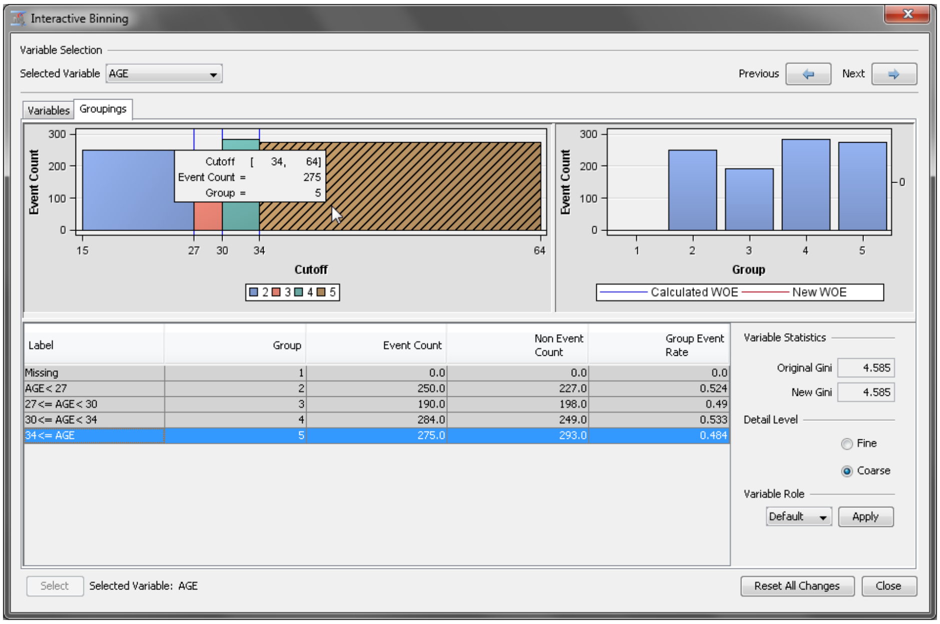
Selected Variable: FREQUENT

Variable	Label	Level	Calculated Role	New Role	Original Gini	Gini Statistic
FREQUENT	Order Frequency	INTERVAL	Input	Default	25.88	25.88
AMOUNT	Dollars Spent	INTERVAL	Rejected	Default	18.315	18.315
RECENCY	Recency	INTERVAL	Rejected	Default	17.809	17.809
DOMESTIC	Domestic Prod.	INTERVAL	Rejected	Default	17.647	17.647
APPAREL	Apparel Purch.	INTERVAL	Rejected	Default	14.725	14.725
DPM12	$ Value/Mailing	INTERVAL	Rejected	Default	14.601	14.601
TOWELS	Towels Purch.	INTERVAL	Rejected	Default	10.325	10.325
WAPPAR	Ladies Apparel	INTERVAL	Rejected	Default	10.015	10.015
LEISURE	Leisure Prod.	INTERVAL	Rejected	Default	9.061	9.061
JEWELRY	Jewelry Purch.	INTERVAL	Rejected	Default	8.964	8.964
KITCHEN	Kitchen Prod.	INTERVAL	Rejected	Default	8.921	8.921
OUTDOOR	Outdoor Prod.	INTERVAL	Rejected	Default	8.64	8.64
PROMO7	Promo: 1-7 mon.	INTERVAL	Rejected	Default	8.621	8.621
PROMO13	Promo: 8-13 mon	INTERVAL	Rejected	Default	8.118	8.118
LAMPS	Lamps Purch.	INTERVAL	Rejected	Default	8.11	8.11
LINENS	Linens Purch.	INTERVAL	Rejected	Default	8.023	8.023
HOMEACC	Home Furniture	INTERVAL	Rejected	Default	6.613	6.613
FLATWARE	Flatware Purch.	INTERVAL	Rejected	Default	6.461	6.461
CUSTDATE	Date 1st Order	INTERVAL	Rejected	Default	5.873	5.873
NTITLE	Name Prefix	NOMINAL	Rejected	Default	5.686	5.686
RETURN	Total Returns	INTERVAL	Rejected	Default	5.274	5.274
HOMEVAL	Home Value	INTERVAL	Rejected	Default	5.026	5.026
MARITAL	Married (y/n)	INTERVAL	Rejected	Default	4.842	4.842
AGE	Age	INTERVAL	Rejected	Default	4.585	4.585
INCOME	Yearly Income	INTERVAL	Rejected	Default	3.817	3.817

Selected Variable: FREQUENT

Groupings Tab: Coarse Detail Setting

After you select the **Groupings** tab of the Interactive Binning window, click the **Coarse** button in the lower right corner for a coarse detail view of your variable groupings.



Variable Selection

Selected Variable: AGE

Label	Group	Event Count	Non Event Count	Group Event Rate
Missing	1	0.0	0.0	0.0
AGE< 27	2	250.0	227.0	0.524
27<= AGE< 30	3	190.0	198.0	0.49
30<= AGE< 34	4	284.0	249.0	0.533
34<= AGE	5	275.0	293.0	0.484

Variable Statistics

Original Gini: 4.585

New Gini: 4.585

Detail Level

○ Fine

● Coarse

Variable Role: Default

Selected Variable: AGE

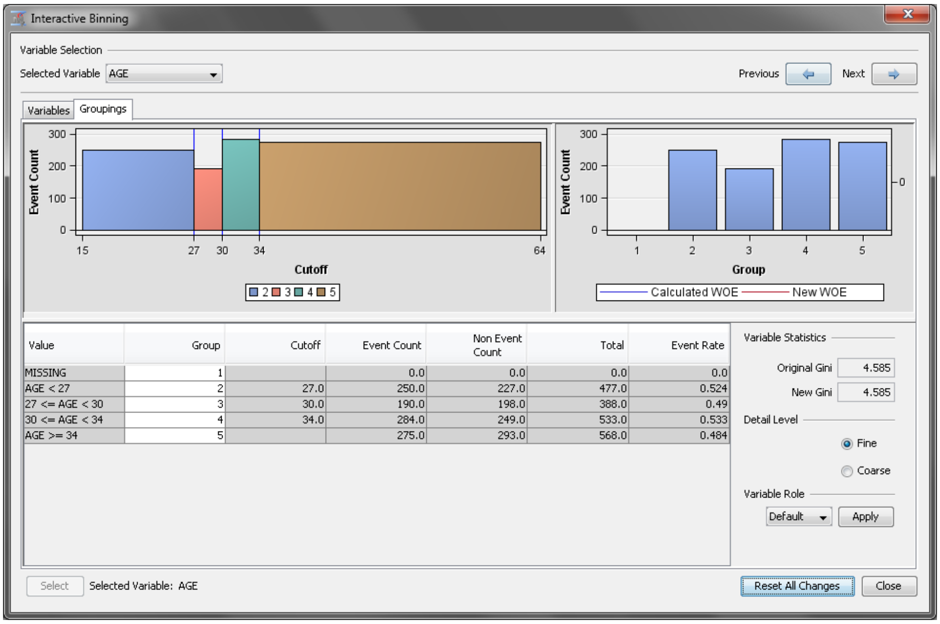
The coarse-detail level displays tables and plots for one variable at a time. The coarse detail view displays a variable grouping table and variable group plots by event count, as well as original and new Gini statistic values.

To change the number of bins, or to modify bin cutoff points, go to the fine detail display by clicking the **Fine** button in the lower right corner.

Groupings Tab: Fine Detail Setting

After you select the **Groupings** tab of the Interactive Binning window, click the **Fine** button in the lower right corner for a fine-detail view of your variable groupings.

Like the coarse-detail level, the fine-detail level displays tables and plots for one variable at a time. You use the variable grouping table in the fine-detail level display to make changes to the number of bins, or to modify bin cutoff points.



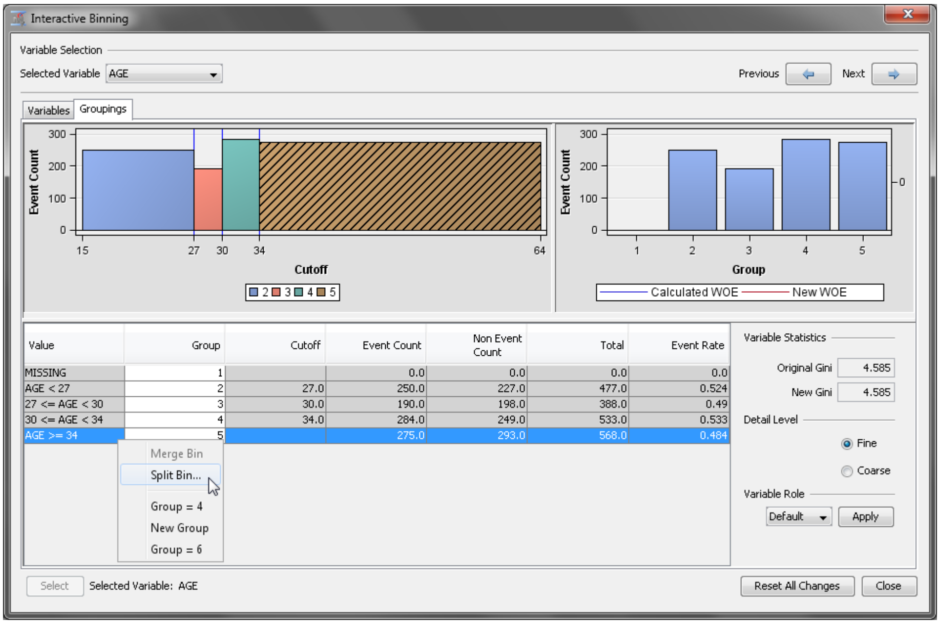
Selected Variable: AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	4	34.0	284.0	249.0	533.0	0.533
AGE >= 34	5		275.0	293.0	568.0	0.484

Original Gini 4.585
New Gini 4.585
● Fine
○ Coarse
Variable Role: Default
Selected Variable: AGE

In the fine-detail variable groups table, you can interactively modify bin cutoff values, add and remove bins, and re-assign group values. You can do these actions one variable at a time. You can interactively use the fine detail split points to construct your final bins. If a class variable is being examined, each level of the variable appears. If an interval variable is being examined, the fine-detail information is based on the binned values of the variable (either quantiles, or equally spaced buckets).

Suppose you want to split the bin for Group 5.0 (AGE>=34) that has cutoff limits of 34 and 64 into two bins by lowering the upper cutoff for Group 5.0 to less than 55 (34<=AGE<55), and then create a new Group 6.0 (AGE>=55) for ages 55 and up. To do this, you right-click **Group 5.0** in the fine-detail group table, and click **Split Bin** on the menu.

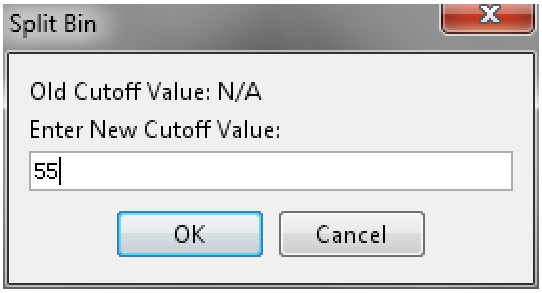


Selected Variable: AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	4	34.0	284.0	249.0	533.0	0.533
AGE >= 34	5		275.0	293.0	568.0	0.484

Original Gini 4.585
New Gini 4.585
● Fine
○ Coarse
Variable Role: Default
Selected Variable: AGE

The Split Bin window appears.

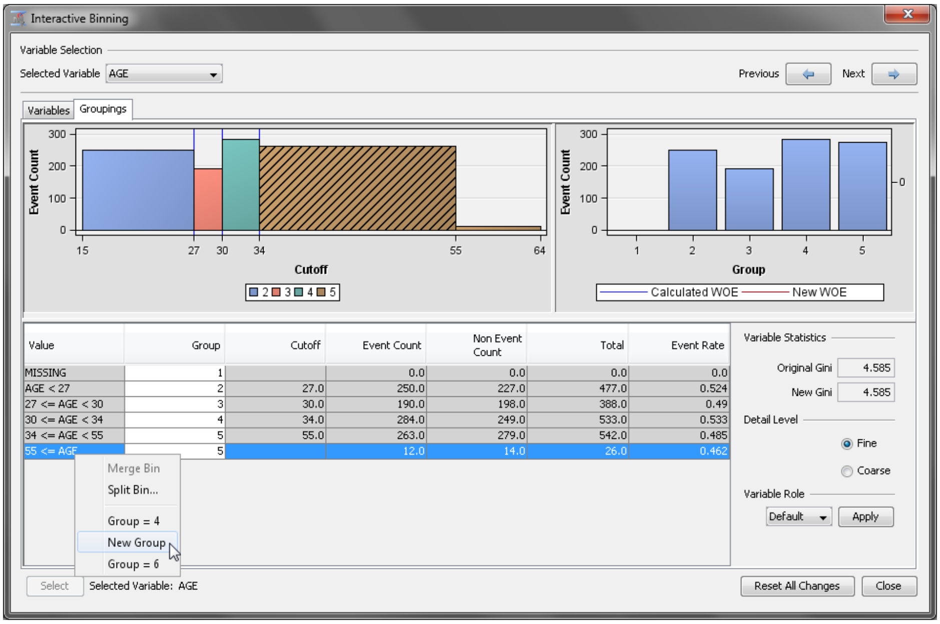


Old Cutoff Value: N/A

Enter New Cutoff Value: 55

Enter the new cutoff point (55) in the Split Bin window and click **OK**.

The bin is now split in the updated Fine Detail plot view. The group statistics table is updated for the event count and event rate columns. However, both the main plot and the variable group table still show both parts of the split bin as members of Group 5.0.



Selected Variable: AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	4	34.0	284.0	249.0	533.0	0.533
34 <= AGE < 55	5	55.0	263.0	279.0	542.0	0.485
55 <= AGE	5		12.0	14.0	26.0	0.462

Variable Statistics

Original Gini 4.585

New Gini 4.585

Detail Level

● Fine

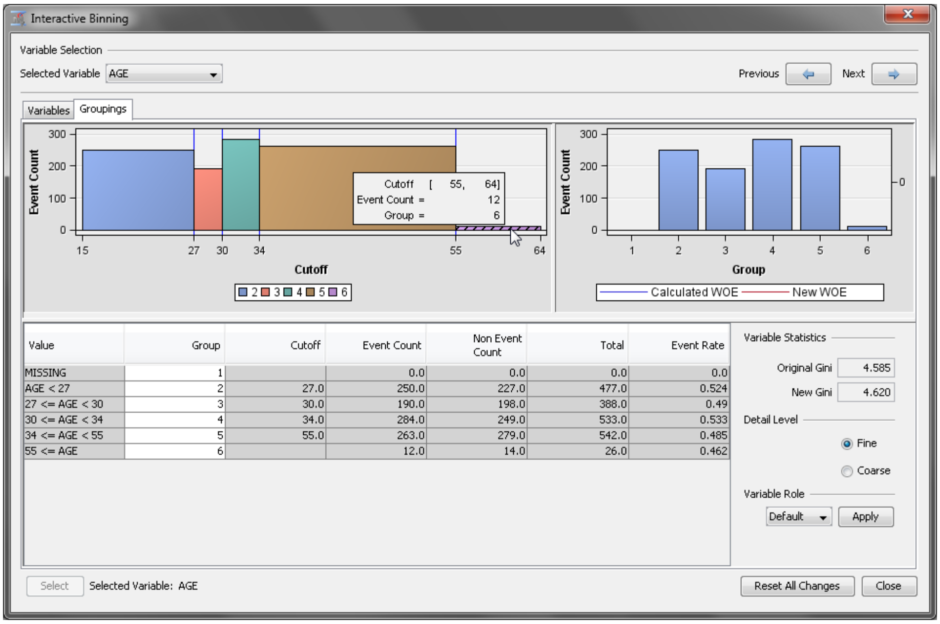
○ Coarse

Variable Role

Default

Selected Variable: AGE

To convert the newly split bin to a new group, right-click the new row in the table, and click **New Group** on the menu. SAS Enterprise Miner updates the plots and data tables to reflect your change.



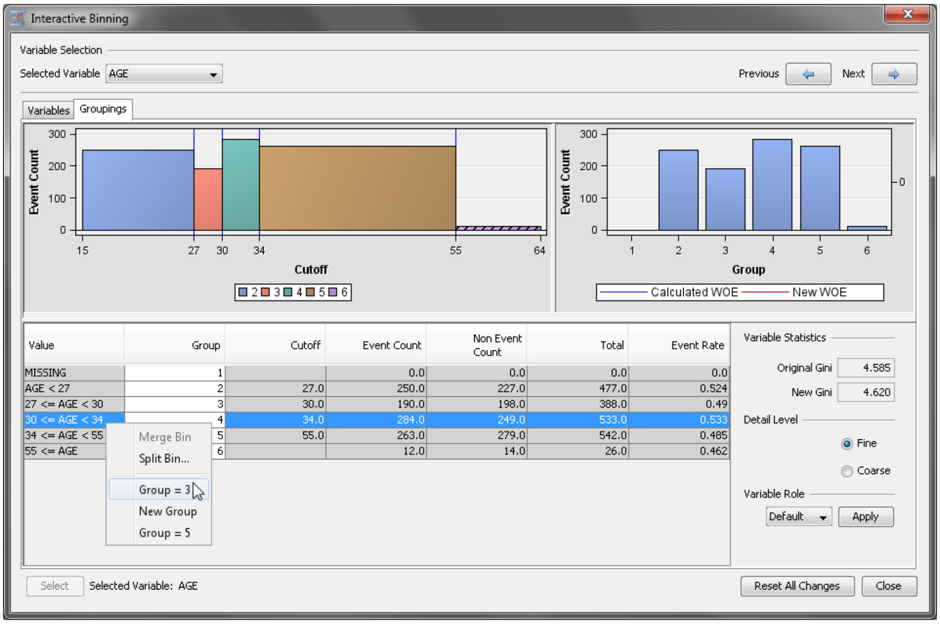
Selected Variable: AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	4	34.0	284.0	249.0	533.0	0.533
34 <= AGE < 55	5	55.0	263.0	279.0	542.0	0.485
55 <= AGE	6		12.0	14.0	26.0	0.462

Original Gini 4.585
New Gini 4.620
Detail Level: ● Fine ○ Coarse
Variable Role: Default

Now, both plots and the group table in the fine-detail level reflect information that is based on your newly binned sixth group. You can compare the **Original Gini** and **New Gini** scores to assess whether your changes are likely to be favorable to the model. Generally, a binning change that increases the value of the original Gini statistic is likely to improve the predictive power of a model. Splitting Group 5.0 into Group 5.0 and Group 6.0 improved the Gini score from 4.585 to 4.620, which could represent a marginal improvement to the predictive capability of the model.

Suppose that you look at Group 3.0 and Group 4.0 for the variable AGE, and wonder if the split between them was arbitrary. Would merging the two groups into one improve the model?

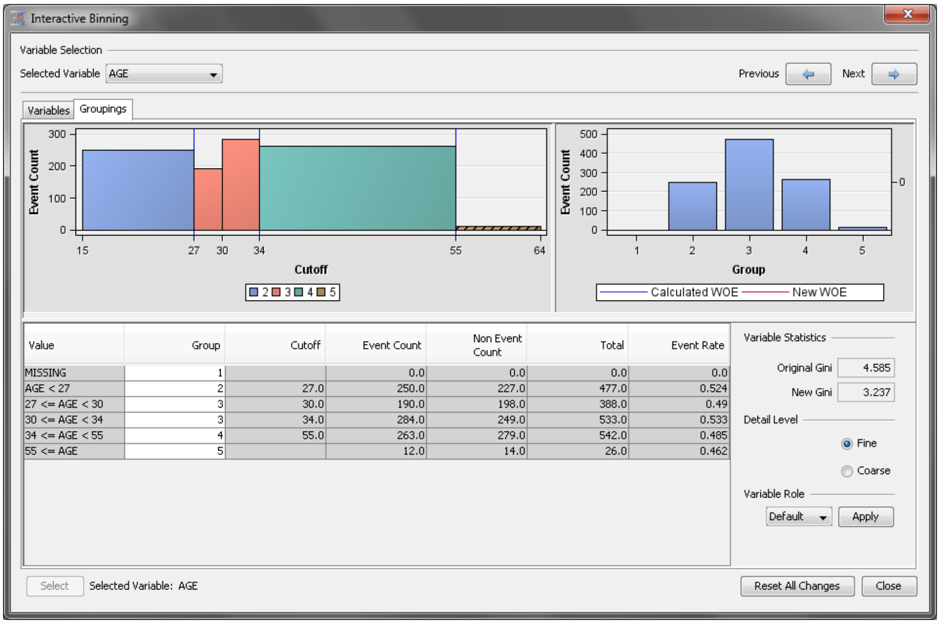


Selected Variable: AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	4	34.0	284.0	249.0	533.0	0.533
34 <= AGE < 55	5	55.0	263.0	279.0	542.0	0.485
55 <= AGE	6		12.0	14.0	26.0	0.462

Original Gini 4.585
New Gini 4.620
Detail Level: ● Fine ○ Coarse
Variable Role: Default

In the fine-detail level table, right-click **Group 4.0** and click **Group 3.0** on the menu. (You could also right-click **Group 3.0** and choose **Group 4.0** from the menu, if you prefer.) SAS Enterprise Miner updates the window plots and group table, but two Group 3.0 bins remain in the main plot and group table.



Selected Variable AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE < 34	3	34.0	284.0	249.0	533.0	0.533
34 <= AGE < 55	4	55.0	263.0	279.0	542.0	0.485
55 <= AGE	5		12.0	14.0	26.0	0.462

Original Gini 4.585

New Gini 3.237

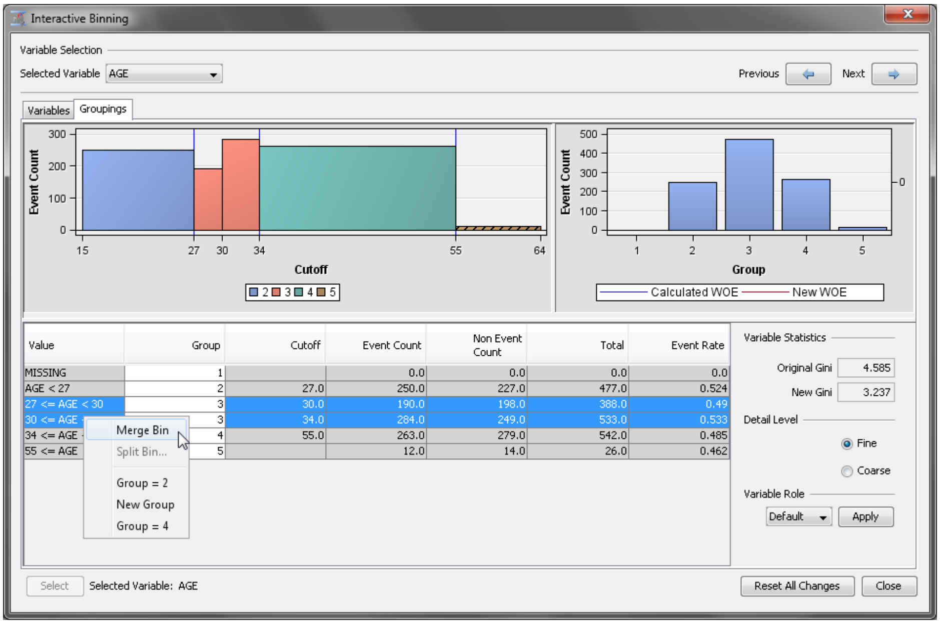
● Fine

○ Coarse

Default

Selected Variable: AGE

To complete the merge, select both of the Group 3.0 rows in the table, right-click, and click **Merge Bin** on the menu.



Selected Variable AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 30	3	30.0	190.0	198.0	388.0	0.49
30 <= AGE	3	34.0	284.0	249.0	533.0	0.533
34 <= AGE	4	55.0	263.0	279.0	542.0	0.485
55 <= AGE	5		12.0	14.0	26.0	0.462

Original Gini 4.585

New Gini 3.237

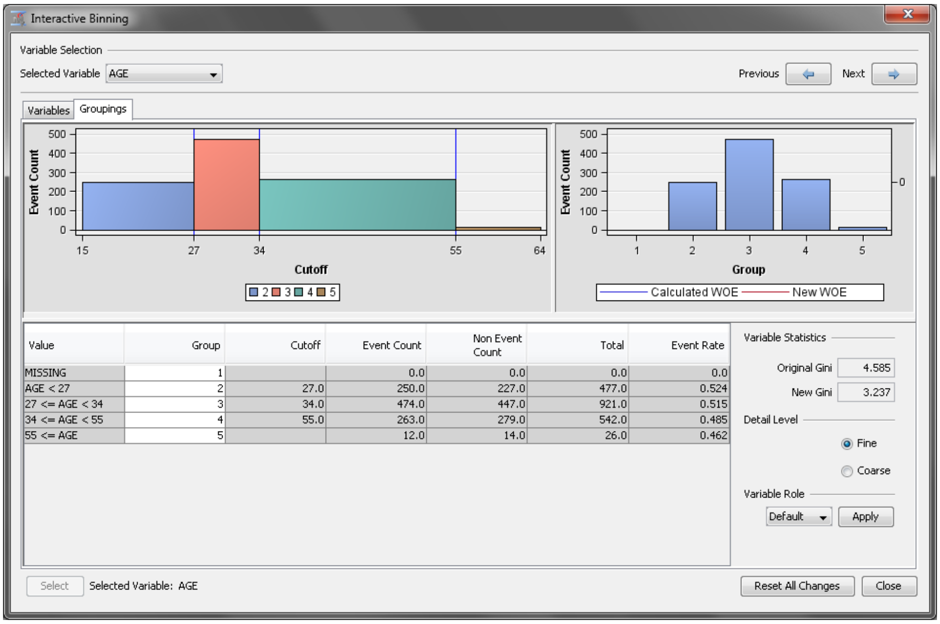
● Fine

○ Coarse

Default

Selected Variable: AGE

SAS Enterprise Miner merges the two bins into one, and displays the net result of your changes:



Selected Variable AGE

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate
MISSING	1		0.0	0.0	0.0	0.0
AGE < 27	2	27.0	250.0	227.0	477.0	0.524
27 <= AGE < 34	3	34.0	474.0	447.0	921.0	0.515
34 <= AGE < 55	4	55.0	263.0	279.0	542.0	0.485
55 <= AGE	5		12.0	14.0	26.0	0.462

Variable Statistics

Original Gini 4.585

New Gini 3.237

● Fine

○ Coarse

Variable Role: Default — Apply

Selected Variable: AGE

Combining the bins for Group 3.0 and Group 4.0 does not appear to enhance the prediction power of the model. The **New Gini** score is now reduced to 3.237 (by the cumulative effects of merging Group 4.0 with Group 3.0, after splitting Group 5.0 into Group 5.0 and Group 6.0).

The **Variable Role** section enables you to modify the role of the current variable. Use the drop-down menu to set the variable's role without switching back to the **Variables** tab.

Click **Reset All Changes** to return all of the model's bins and cutoff points to their original settings. Clicking **Close** saves your interactive binning changes and closes the Interactive Binning window.

Interactive Binning Node Results

You can open the Results window of the **Interactive Binning** node by right-clicking the node and clicking **Results** on the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

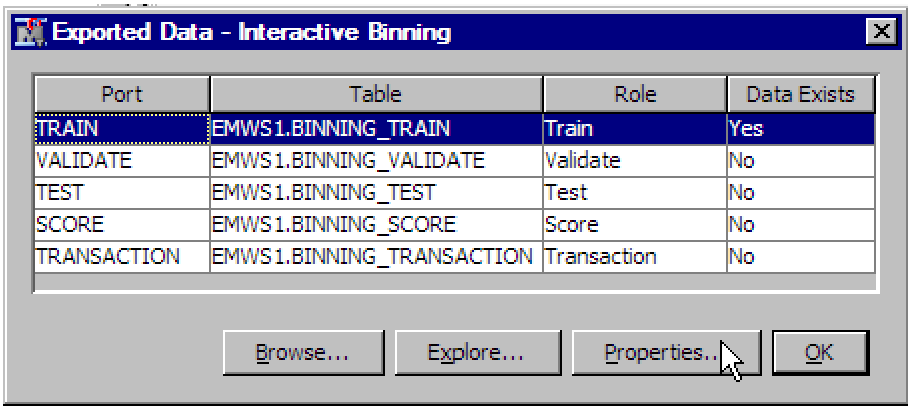
Click **View** on the main menu to view the following information in the Results window:

- **Properties**
  - **Settings** — displays a window that has a read-only table of the **Interactive Binning** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **Interactive Binning** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — displays a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays the information that is entered by the user for the node.

- **SAS Results**
    - **Log** — the SAS log of the **Interactive Binning** node run.
    - **Output** — the SAS output of the **Interactive Binning** node run.
    - **Flow Code** — the SAS code used to produce the output that the **Interactive Binning** node passes on to the next node in the process flow diagram.
- **Scoring**
    - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications. SAS Code contains Interactive Binning values of input variables only. This includes Interactive Binning of unknown levels (if specified) and Interactive Binning of specific levels. If no Interactive Binning values were generated, the **SAS Code** menu item is dimmed and unavailable.
    - **PMML Code** — the Interactive Binning Node does not generate PMML code.
- **Model**
    - **Event Rate Statistics** — displays a table of all group values for all input variables. For each variable and group value combination, columns provide information about variable role, group number, and event count.
    - **Output Variables** — displays the columns of the Output Variables table with Gini Statistic, Exported role, Level, Variable Labels, and Gini Ordering information for each of the input variables. Variables that have a Gini Statistic of less than 20.0 are assigned the role "Rejected."
    - **Statistics Plot** — displays the value of the Gini statistic for each of the input variables. Position your pointer over a vertical bar to see the variable name, Gini statistic, and exported role.
    - **Event Rate Plot** — a lattice plot that shows the event rates across group values for each of the input variables. Each lattice represents the group values for an input variable.
- **Table** — enables you to open a table of the data that is associated with the graph that you have open. Data table column headings display variable labels by default. To display the variable names used in SAS code as column headings, right-click in any table cell and click **Show Variable Names**. To view labels in the column headings again, right-click in any table cell and click **Show Variable Labels**.
- **Plot** — enables you to create or modify a graph of the data in the table that you have open.

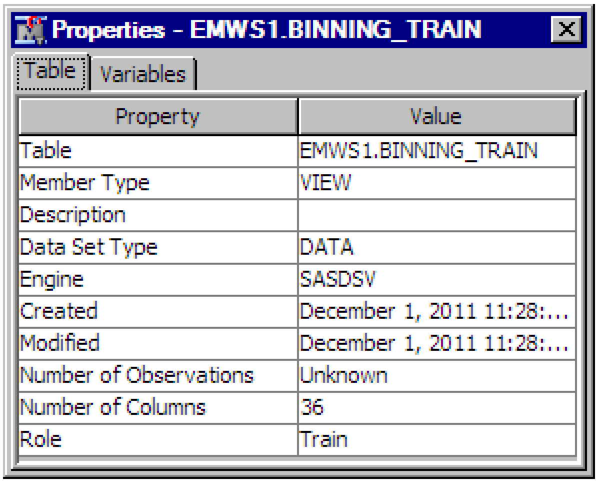
Interactive Binning Node Output Data

You can view the **Interactive Binning** node output data after you successfully run the node. Select the node in the Diagram Workspace, and then click the ... button to the right of the Exported Data property in the Properties panel to open the exported data table for the **Interactive Binning** node.



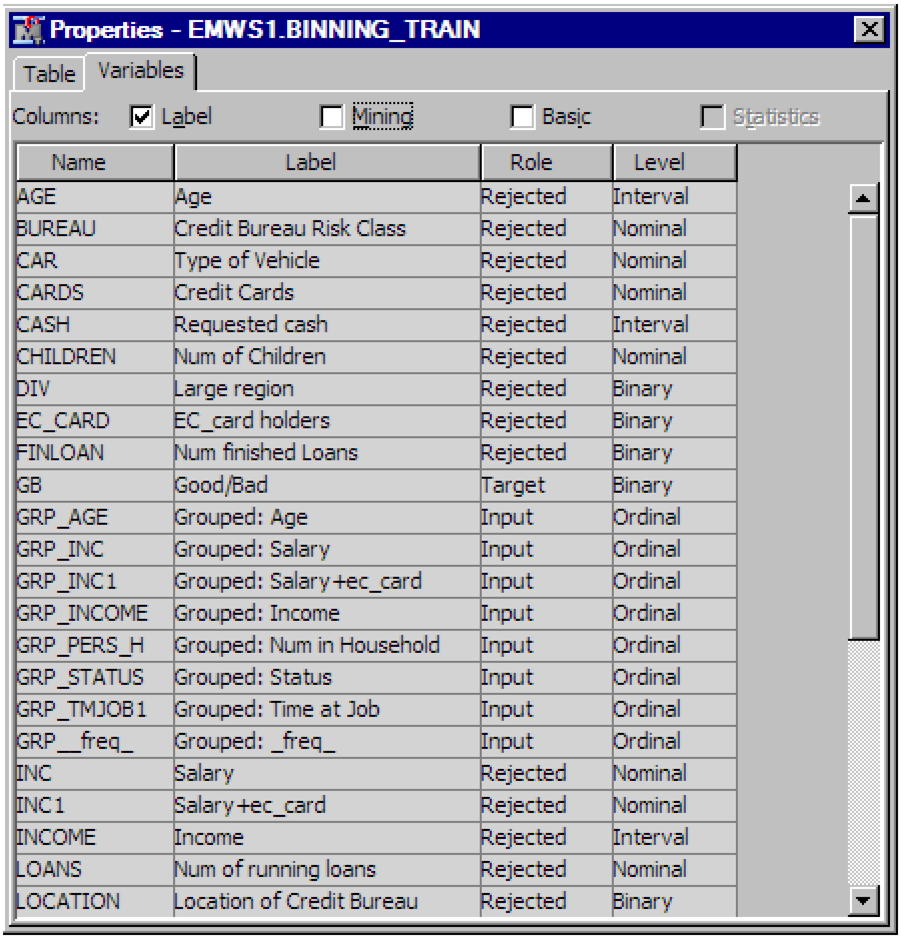
Port	Table	Role	Data Exists
TRAIN	EMWS1.BINNING_TRAIN	Train	Yes
VALIDATE	EMWS1.BINNING_VALIDATE	Validate	No
TEST	EMWS1.BINNING_TEST	Test	No
SCORE	EMWS1.BINNING_SCORE	Score	No
TRANSACTION	EMWS1.BINNING_TRANSACTION	Transaction	No

The Exported Data — Interactive Binning window contains a table that lists the exported Interactive Binning data sets. This example was not partitioned and contains only training data. To view a summary of the exported training data, highlight the data set in the window and click **Properties**. This opens the Properties — EMWS1.BINNING_TRAIN window, which contains summary information about the output data set.



Property	Value
Table	EMWS1.BINNING_TRAIN
Member Type	VIEW
Description	
Data Set Type	DATA
Engine	SASDSV
Created	December 1, 2011 11:28:...
Modified	December 1, 2011 11:28:...
Number of Observations	Unknown
Number of Columns	36
Role	Train

The properties window for the interactive binning training data opens by default to the **Table** tab. The table tabs provides summary information about the training table. Clicking the **Variables** tab provides a variable summary of the output data:



Columns: ☑ Label □ Mining □ Basic □ Statistics

Name	Label	Role	Level
AGE	Age	Rejected	Interval
BUREAU	Credit Bureau Risk Class	Rejected	Nominal
CAR	Type of Vehicle	Rejected	Nominal
CARDS	Credit Cards	Rejected	Nominal
CASH	Requested cash	Rejected	Interval
CHILDREN	Num of Children	Rejected	Nominal
DIV	Large region	Rejected	Binary
EC_CARD	EC_card holders	Rejected	Binary
FINLOAN	Num finished Loans	Rejected	Binary
GB	Good/Bad	Target	Binary
GRP_AGE	Grouped: Age	Input	Ordinal
GRP_INC	Grouped: Salary	Input	Ordinal
GRP_INC1	Grouped: Salary+ec_card	Input	Ordinal
GRP_INCOME	Grouped: Income	Input	Ordinal
GRP_PERS_H	Grouped: Num in Household	Input	Ordinal
GRP_STATUS	Grouped: Status	Input	Ordinal
GRP_TMJOB1	Grouped: Time at Job	Input	Ordinal
GRP__freq_	Grouped: _freq_	Input	Ordinal
INC	Salary	Rejected	Nominal
INC1	Salary+ec_card	Rejected	Nominal
INCOME	Income	Rejected	Interval
LOANS	Num of running loans	Rejected	Nominal
LOCATION	Location of Credit Bureau	Rejected	Binary

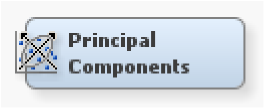
The **Label**, **Mining**, and **Basic** check boxes toggle the display of extra table columns that provide additional metadata information about the variables. You can choose the level of display that you like best.

The output data sets from the **Interactive Binning** node contain the original input variables, plus a **GRP_<variable-name>** variable, which contains the identification number of the group that an observation belongs to.

The roles of all of the input and output variables (except the target, frequency, and GRP variables for those inputs that meet the variable selection criterion) are set to **Rejected**.

Chapter 44
Principal Components Node

Principal Components Node



Overview of the Principal Components Node

The Principal Components node belongs to the Modify category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA). The Principal Components node calculates eigenvalues and eigenvectors from the uncorrected covariance matrix, corrected covariance matrix, or the correlation matrix of input variables. Principal components are calculated from the eigenvectors and can be used as inputs for successor modeling nodes in the process flow diagram. Since interpreting principal components is often problematic or impossible, it is much safer to view them simply as a mathematical transformation of the set of original variables.

A principal components analysis is useful for data interpretation and data dimension reduction. It is usually an intermediate step in the data mining process. Principal components are uncorrelated linear combinations of the original input variables; they depend on the covariance matrix or the correlation matrix of the original input variables. Principal components are usually treated as the new set of input variables for successor modeling nodes.

In the Principal Components node, a set of dummy variables is created for each class of the categorical variables. Instead of original class variables, the dummy variables are used as interval input variables in the principal components analysis.

Principal Components Node Properties

Principal Components Node General Properties

The following general properties are associated with the Principal Components node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Principal Components node added to a diagram will have a Node ID of PRINCOMP. The second Principal Components node added to a diagram will have a Node ID of PRINCOMP2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — Principal Components window. The Imported Data — Principal Components window contains a list of the ports that provide data sources to the Principal Components node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — the Exported Data property provides access to the Exported Data — Principal Components window. The Exported Data — Principal Components window contains a list of the output data ports that the Principal Components node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Principal Components Node Train Properties

The following train properties are associated with the Principal Components node:

- **Variables** — Select the ... button to open the Variables — Principal Components table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Report, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Eigenvalue Source** — Use the Eigenvalue Source property of the Principal Components node to specify the type of source matrix that you want to use to calculate eigenvalues and eigenvectors. You can choose from
  - **Covariance** — Use the covariance matrix to calculate eigenvalues and eigenvectors.
  - **Correlation** (default) — Use the correlation matrix to calculate eigenvalues and eigenvectors.
  - **Uncorrected** — Use the uncorrected matrix to calculate eigenvalues and eigenvectors.
- **Interactive Selection** — Once the Principal Components node has successfully run, you can interactively view and select the Principal Components that you want to export to the successor node. Select the ... button to the right of the Interactive Selection property to open the Interactive Principal Components Selection table. You use the Plot list to select the y-axis Eigenvalue display. The y-axis display options are
  - **Eigenvalue**
  - **Proportional Eigenvalue**
  - **Cumulative Proportional Eigenvalue**
  - **Log Eigenvalue**
  - **Eigenvalue Table**

  After you select the y-axis display, you can use the spin box to change the value of the Number of Principal Components to be exported. If you change the value in the box, the chart and the yellow Principal Component threshold line update to reflect your new Principal Component selection value.
- **Print Eigenvalue Source** — Use the Print Eigenvalue Source property of the Principal Components node to specify whether to print the covariance or correlation matrix that is used to calculate eigenvalues. The printed output depends on the value that you specify in the Eigenvalue Source property. The default setting for the Print Covariance or Correlation Matrix property is **No**.

Principal Components Node Score Properties

The following score properties are associated with the Principal Components node:

- **Princomp Prefix** — Use the Princomp Prefix property to specify the prefix that will be used to create SAS variable names for the exported Principal Component variables. The default prefix setting for the Princomp Prefix property is PC. String values are acceptable values for the Princomp Prefix property.

Principal Components Node Score Properties: Eigenvalue Cutoff

- **Cumulative** — Use the Cumulative property of the Principal Components node to specify the cutoff criterion of the cumulative proportion of the total variance that is attributable to principal components. Principal components that have a cumulative proportional variance greater than the cutoff value are not passed to successor nodes. The default value for the Cumulative property is 0.99. The acceptable values for the Cumulative property are 0.7, 0.75, 0.80, 0.85, 0.875, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97. 0.98, 0.99, 0.999, 0.9999, 1.0.

- **Increment** — Use the Increment property of the Principal Components node to specify the cutoff criterion of the proportional increment for the total variance that is attributable to principal components. After the cumulative proportional variance reaches 0.9, the principal components that have a proportional increment less than the cutoff value are not passed on to successor nodes. The default value for the Increment property is 0. The acceptable values for the Increment property are 0.000001, 0.00001, 0.00005, 0.0001, 0.00025, 0.0005, 0.00075, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.09, 0.1.

Principal Components Node Score Properties: Max Number Cutoff

- **Apply Maximum Number** — Use the Apply Maximum Number property of the Principal Components node to indicate whether to apply the upper threshold for the maximum number of principal components that are passed to successor nodes. When the Apply Maximum Number property is set to **Yes**, you must specify the threshold value in the Maximum Number property. The default setting of the Apply Maximum Number property is **Yes**.

- **Maximum Number** — Use the Maximum Number property of the Principal Components node to specify a value for the maximum number of principal components to be passed to successor node, when the Apply Maximum Number property is set to **Yes**. Permissible values are nonnegative integers. The default value of the Maximum Number property is 20.

- **Reject Original Input Variables** — Use the Reject Original Input Variables property of the Principal Components node to indicate whether the original inputs should be excluded when the principal components are passed. The default setting for the Reject Original Input Variables property is **Yes**, which suppresses the original input variables from being passed to successor nodes.

- **Hide Rejected Variables** — Set the Hide Rejected Variables property of the Principal Components node to **No** if you want to keep the original variables in your transformed data set. The default setting for the Hide Rejected Variables property is **Yes**. When set to **Yes**, the original variables will be removed from the exported metadata that is sent to successor nodes, but the original variables are not removed from the exported data sets and data views.

Principal Components Node Status Properties

The following status properties are associated with this node:

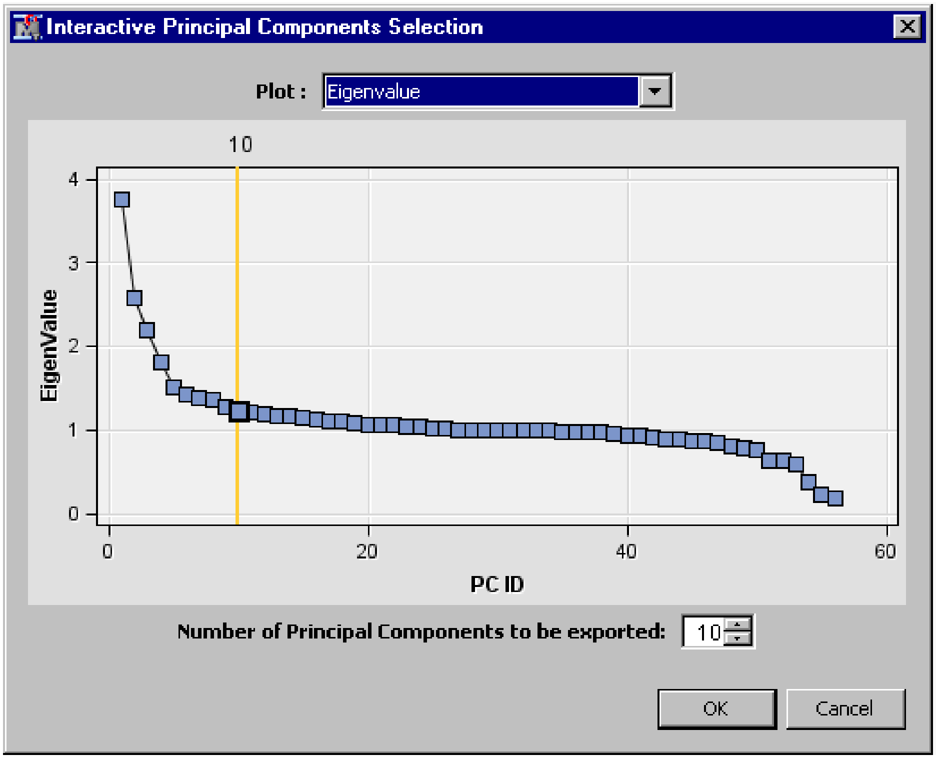
- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Interactive Principal Components Selection Window

The Interactive Principal Components Selection window enables you to change the number of principal components that are exported from the node after the node results have been generated. To open the Interactive Principal Components Selection window, you must run the node and select the ... button to the right of the Interactive Selection property in the Principal Components node Properties panel.

The following is an example of the Interactive Principal Components Selection window. Ten principal components are exported to successor nodes in the chart below. Selecting five principal components might be a better choice, based on the slope of the plot.



Plot : Eigenvalue

Number of Principal Components to be exported: 10

You can display a line plot for the Eigenvalue, Proportional Eigenvalue, Cumulative Proportional Eigenvalue, or Log Eigenvalue. To select the plotting variable, select an item from the **Plot** list. You also can display the eigenvalues in a table by selecting **Eigenvalue Table** from the list. The position of the reference line determines the number of principal components that are passed to successor nodes.

You can use one of the following methods to change the number of principal components that are exported to successor nodes:

- Click a data point in the plot.
- Use the **Number of Principal Components to be exported** spin box to change the value.
- Select a row in the Eigenvalue Table. For example, if you select the fifth row in the table, which corresponds to the fifth principal components, then the first five principal components (PC_1 - PC_5) will then be exported.

After you manually change the number of principal components that are exported from the Principal Components node, click OK to apply the changes, and to regenerate the results and score code.

Principal Components Node Results

You can open the Results window of the Principal Components node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following information in the Principal Components node Results window:

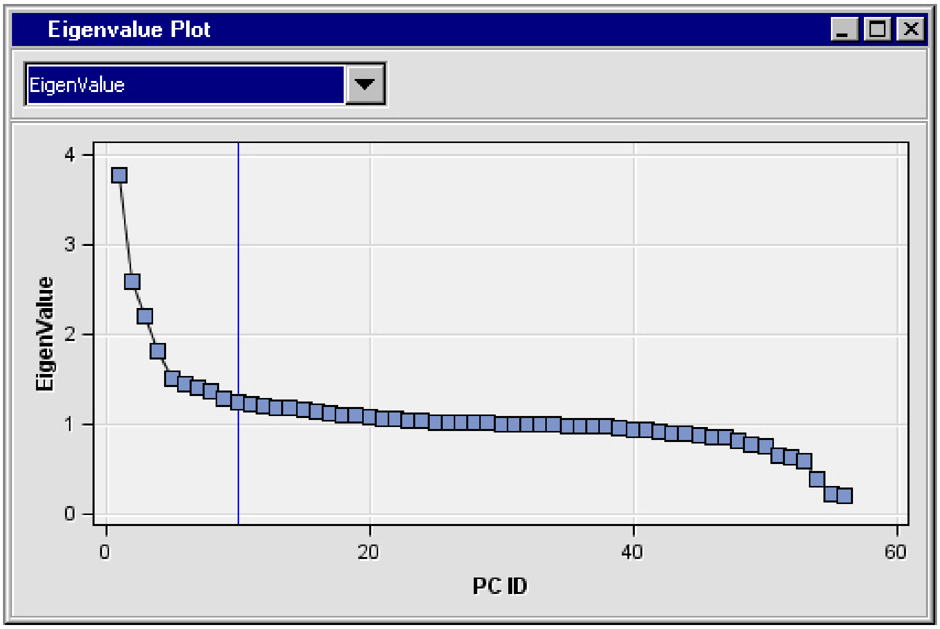
- **Properties**
  - **Settings** — displays a window with a read-only table of the Principal Components node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Principal Components node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Principal Components node run.
  - **Output** — the SAS output of the Principal Components node run. The Output window displays the mean and standard deviations for interval input variables and for each level of categorical input variables. Eigenvalues, differences between consecutive eigenvalues, and the (noncumulative and cumulative) proportion to the total variance are listed. At the end of the SAS output, summaries of the criteria for exporting principal components are displayed.

    The summary includes the following information:

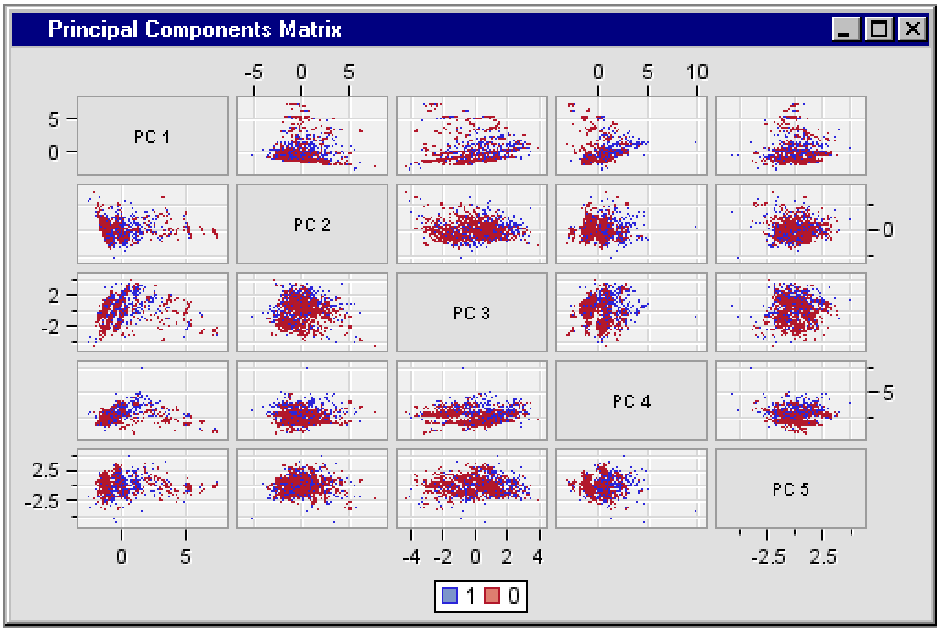
    - total number of input variables that are used in the principal components analysis
    - cutoff value for the maximum number of principal components
    - cutoff value for the cumulative proportional eigenvalue
    - cutoff value for the increment of proportional eigenvalue
    - number of exported principal components

    - percentage of total variance that is attributable to the exported principal components.
  - **Flow Code** — the SAS scoring code that was produced for the Principal Components analysis to be passed on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Principal Components node does not generate PMML code.
- **Plots**
  - **Eigenvalue Plot** — You can plot the following values in the Eigenvalue Plot in the results of the Principal Components node. By default, eigenvalues are plotted. Select an item from drop-down menu to change the plotting value.
    - **Eigenvalue** — the eigenvalues for successive principal component. This plot is also called the Scree plot. This graphical method was first proposed by Cattell (1966). His method finds the sudden drop point of the smooth decrease of eigenvalues.
    - **Proportional Eigenvalue** — the proportional eigenvalue for each principal component.
    - **Cumulative Proportional Eigenvalue** — the cumulative proportional eigenvalues for the principal components.
    - **Log Eigenvalue** — the logarithms of eigenvalues for successive principal components.

    The vertical reference line indicates the number of principal components that will be output to the successor node.

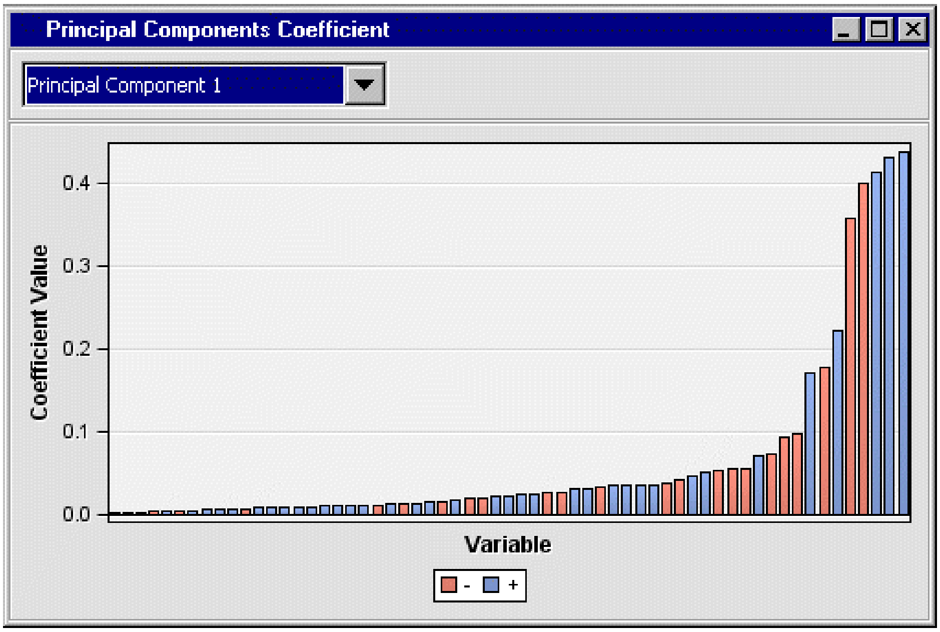


  - **Principal Components Matrix** — The Principal Components Matrix plot displays scatter plots for all pairs of the selected principal components. By default, the initial matrix plot shows the scatter plots for the first five principal components. The following example shows the Principal Components Matrix plot for the binary target variable BAD. The data points are color coded, depending on the value of the target variable.



To change the number of principal components that are included in the Principal Components Matrix plot, right-click in the plot, select Data Options, and change the role to None for the variables that you want to exclude. In the Data Options dialog box, an interval target variable has the role of Group, and a categorical target variable has the role of Color. If there is more than one target variable in the input data source, the first one will be used to create the matrix plot. If there is no target variable in the input data source, there will be no Group or Color variable, and data points will have the same color. The variables (principal components) that are included in the plot have the role of MatrixVar. If only one variable has the role of MatrixVar, the Principal Components Matrix plot will display a histogram for the principal component.

- **Principal Components Coefficient** — The Principal Components Coefficient plot displays the input variable coefficients for each of the principal components. Select a principal component from the drop-down menu.



The histogram bar height corresponds to the coefficient that was applied to the successive input variables. Blue bars represent positive coefficients, and red bars represent negative coefficients.

- **Table** — opens the data table that corresponds to the graph that is currently in focus.
- **Plot** — opens the Select a Chart Type plotting wizard so that you can plot the data in the table that is currently in focus.

Principal Components Node Output Data Sources

The data sets that the Principal Components node creates as output can be examined after the node runs. With the Principal Components node selected in the Diagram Workspace, select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

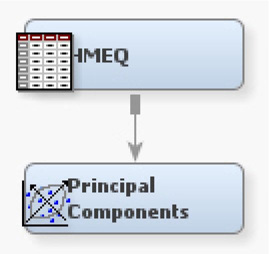
The Principal Components node can create the following data sources to be passed to the successor node. The naming convention of data sources is given in parentheses.

- Train (<libref>.PRINCOMP_TRAIN) — the scored training data set that contains all the variables in the training data set and the principal components.
- Validate (<libref>.PRINCOMP_VALIDATE) — the scored validation data set that contains all the variables in the validation data set and the principal components.
- Test (<libref>.PRINCOMP_TEST) — the scored test data set that contains all the variables in the test data set and the principal components.

- Eigen (<libref>.PRINCOMP_EIGEN) — The eigen data source contains the following variables for each principal component.
  - **PC ID** — the ID number of a principal component.
  - **Eigenvalue** — the eigenvalue of a principal component.
  - **Difference** — the difference of eigenvalues between two consecutive principal components.
  - **Proportional Eigenvalue** — the noncumulative proportion to the total variance.
  - **Cumulative Proportional Eigenvalue** — the cumulative proportion to the total variance.
  - **LogEigenvalue** — the logarithm of the eigenvalue.
  - **Exported** — the export status of a principal component.
- Eigenvector (<libref>.PRINCOMP_EIGENVECTOR) — As described in the Overview section, a set of dummy variables is created for each class of the categorical variables. Instead of original categorical variables, the dummy variables are used as interval input variables in the principal components analysis. The eigenvector data source contains the following information about the interval input variables and the dummy variables that are created from the categorical variables.
  - **Variable** — lists the names of original interval variables and the dummy variables. For example, the dummy variable names for a categorical variable AGE that has three levels are AGE_1_, AGE_2_, and AGE_3_.
  - **Label** — the class of categorical input variables.
  - **PC_1,PC_2**, ... — the eigenvectors.

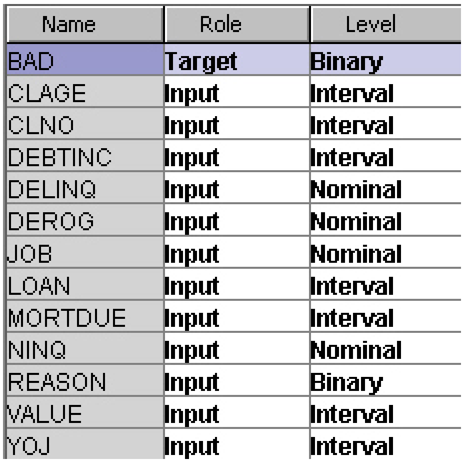
Principal Components Node Example

This example uses a mortgage scoring data set (SAMPSIO.HMEQ). The target variable (BAD) is binary and it indicates whether an applicant defaulted on or was ever seriously delinquent in making payments. The principal components analysis uses the other 12 variables as inputs and the generated principal components are passed to a successor modeling node.



Follow these steps to create the above example flow:

1. Use the Create a Data Source wizard to create a raw data source named HMEQ from the SAS sample table SAMPSIO.HMEQ. Use the Advanced Metadata Advisor option to set the role and level values for each variable as shown below:

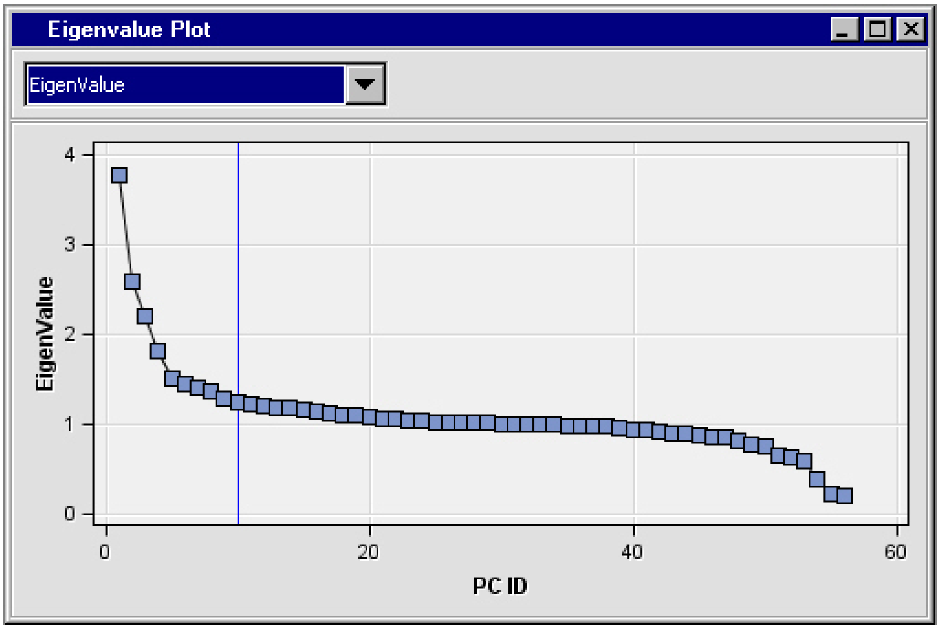


Name	Role	Level
BAD	Target	Binary
CLAGE	Input	Interval
CLNO	Input	Interval
DEBTINC	Input	Interval
DELINQ	Input	Nominal
DEROG	Input	Nominal
JOB	Input	Nominal
LOAN	Input	Interval
MORTDUE	Input	Interval
NINQ	Input	Nominal
REASON	Input	Binary
VALUE	Input	Interval
YOJ	Input	Interval

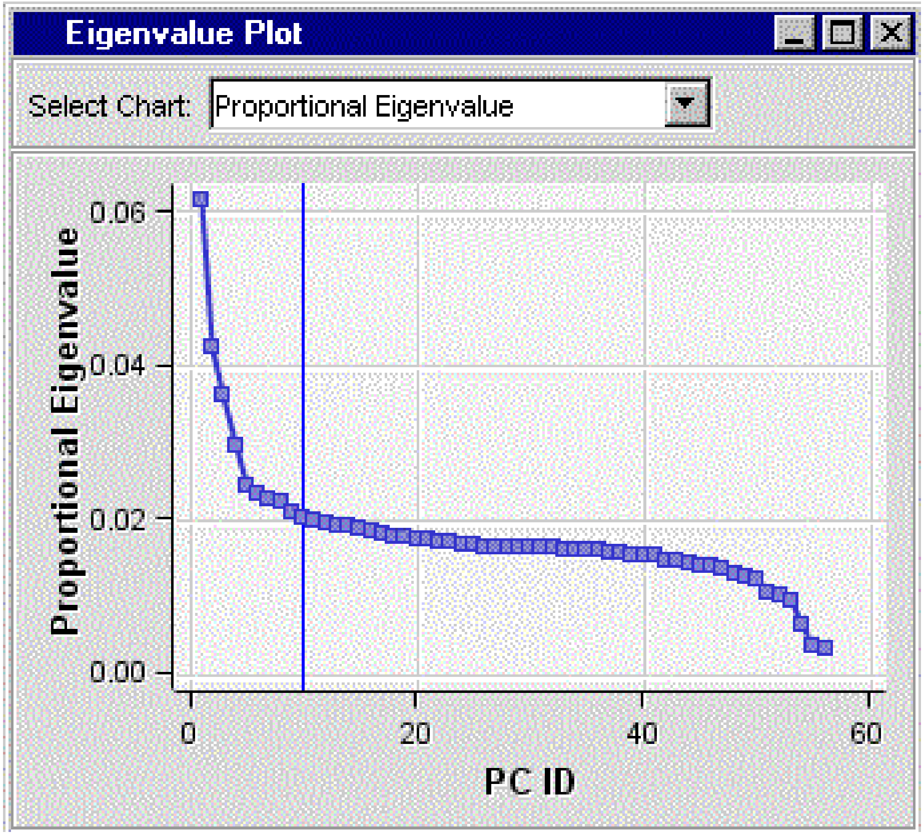
2. Drag the SAMPSIO.HMEQ data source from the Data Sources list to the Diagram Workspace.
3. Drag and drop a Principal Components node from the Modify tool folder to the Diagram Workspace. Connect the SAMPSIO.HMEQ Input Data node to the Principal Components node.
4. Click the Principal Components node in the Diagram Workspace to select it, then in the node Properties Panel, change the value of the Cumulative property in the Eigenvalue Cutoff group to 0.97, and set the Maximum Number property in the Max Number Cutoff group to 10.
5. Run the Principal Components node and view the results.

The Principal Components node uses several graphical displays to show the eigenvalues of all principal components. The available graphical displays are Eigenvalue, Proportional Eigenvalue, Cumulative Proportional Eigenvalue, Log Eigenvalue, Principal Components Coefficients plots, and Principal Components Matrix.

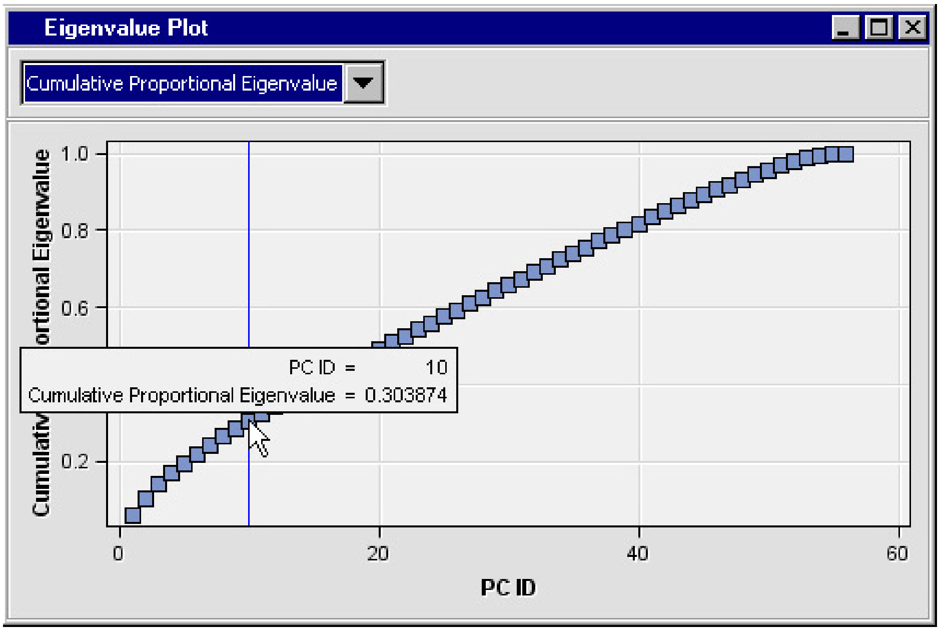
The Eigenvalue plots displays the associated eigenvalues:



The plotted values in the Proportional Eigenvalue and Cumulative Proportional Eigenvalue plots represent the noncumulative or cumulative contribution of the eigenvalue for each principal component to the total variance. The cumulative proportional eigenvalues show a nondecreasing curve. Use the Proportional Eigenvalue and Cumulative Proportional Eigenvalue plots to decide the number of principal components to be exported for successive analysis. The following display shows the Proportional Eigenvalue and Cumulative Proportional Eigenvalue plots for this example:



Select Chart: Proportional Eigenvalue

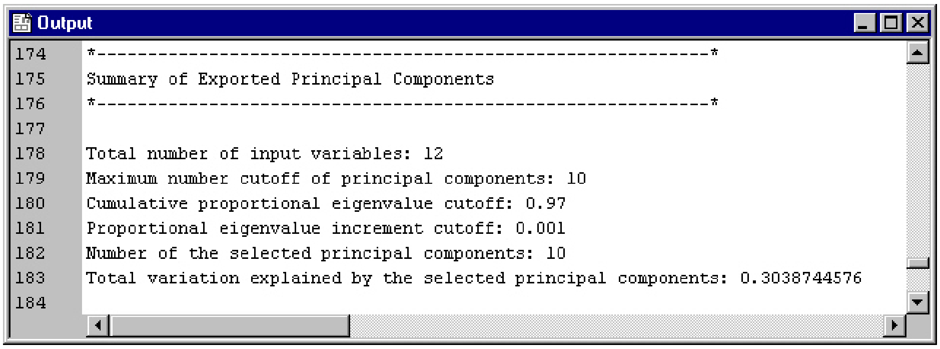


Cumulative Proportional Eigenvalue

PC ID = 10
Cumulative Proportional Eigenvalue = 0.303874

When you move your mouse pointer over a principal component on the curve in the Cumulative Proportional Eigenvalue plot, a ToolTip displays the cumulative proportion of total variance up to the selected principal component. In this example, the first ten principal components represent more than 30.3% of the total variance.

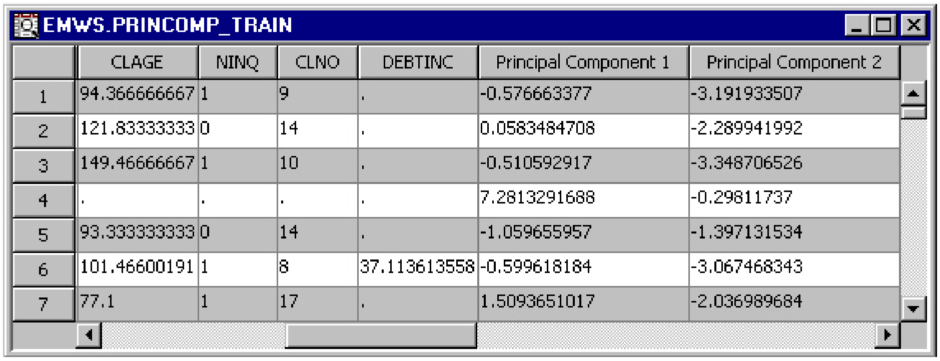
The Output window displays standard SAS output from running the Principal Components node. Here is a display that shows the criterion summary of the exported principal components:



```
*------------------------------------------------------------*
Summary of Exported Principal Components
*------------------------------------------------------------*

Total number of input variables: 12
Maximum number cutoff of principal components: 10
Cumulative proportional eigenvalue cutoff: 0.97
Proportional eigenvalue increment cutoff: 0.001
Number of the selected principal components: 10
Total variation explained by the selected principal components: 0.3038744576
```

In this example, the Principal Components node recommends that you export ten principal components based on the node properties that were specified before running the node. The variables that are associated with the principal components are in the scored training data set. To view the scored training data set, select the ... button to the right of the Exported Data property in the properties panel, choose the training data set from the Exported Data window, and then click **Browse**.

The scored training data set has all the original variables and the variables that represent the principal components. The following display lists the contents of the scored training data set (EMWS.PRINCOMP_TRAIN) for this example:



	CLAGE	NINQ	CLNO	DEBTINC	Principal Component 1	Principal Component 2
1	94.366666667	1	9	.	-0.576663377	-3.191933507
2	121.83333333	0	14	.	0.0583484708	-2.289941992
3	149.46666667	1	10	.	-0.510592917	-3.348706526
4	.	.	.	.	7.2813291688	-0.29811737
5	93.333333333	0	14	.	-1.059655957	-1.397131534
6	101.46600191	1	8	37.113613558	-0.599618184	-3.067468343
7	77.1	1	17	.	1.5093651017	-2.036989684

The values in the columns Principal Component 1 through Principal Component 10 are the principal component variable values that are exported to successor nodes. The principal component variables can be used as inputs for predictive modeling.

References

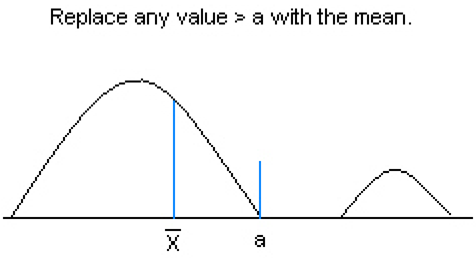
Cattell, R.B "The Scree Test for the Number of Factors." 1966. *Multivariate Behavioral Research* 1: 245–276.

Chapter 45
Replacement Node

Replacement Node

Overview of the Replacement Node

The Replacement node belongs to the Modify category of the SAS SEMMA (Sample, Explore, Modify, Model, and Assess) data mining process. You use the SAS Enterprise Miner Replacement node to generate score code to process unknown levels when scoring and also to interactively specify replacement values for class and interval levels. In some cases you may want to reassign specified nonmissing values (trim your variable's distribution) before performing imputation calculations for the missing values. For example, assume you are working with the bimodal distribution below:



You may want to trim the right side of the distribution to create a tighter, more centralized distribution to be used for missing variable imputation. In this example, you might replace all values in the second "hump" of the distribution (values > a) with the distribution mean, Xbar. To trim the distribution, or to replace specific nonmissing values before you impute missing values, use a Replacement node before an Impute node.

The Replacement node must follow a node that exports a data set, such as a Data Source, Sample, or Data Partition node.

Replacement Node Properties

Replacement Node General Properties

The following general properties are associated with the Replacement node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Replacement node added to a diagram will have a Node ID of Repl. The second Replacement node added to a diagram will have a Node ID of Repl2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — Replacement window. The Imported Data — Replacement window contains a list of the ports that provide data sources to the Replacement node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — Use the Exported Data property to view a table that lists the data set or data sets that are exported by the Replacement node to a successor node. Select the ... button to open the Exported Data — Replacement table. The Exported Data — Replacement table lists the originating port of the exported data, the name of the table that contains the exported data, and the assigned role of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Replacement Node Train Properties: Interval Variables

- **Replacement Editor** — Click the ... button to open the "Interactive Replacement Interval Filter Window" on page 676 for the associated data set. Use the Replacement Editor window to specify new values for specific interval levels of variables in the data imported to the Replacement node. Predecessor nodes to the Replacement node must be run before opening the Replacement Editor.

The Replacement Editor is a table that lists all interval variables. The interactive window is especially useful if you wish to specify different limits methods for individual variables instead of applying the default limits method to all interval variables. You can use the columns in the table to specify the following replacement parameters for individual variables: Use, Report, Limit Method, Lower Limit, Upper Limit, Replace Method, Lower Replacement Value, and Upper Replacement Value. You can use the Interactive Replacement Interval Filter table on page 676 to filter an interval variable for a single value by setting both the Upper Limit and Lower Limit columns for that variable to the desired variable value.

- **Default Limits Method** — Use the Default Limits Method property to specify the default method to determine the range limits for interval variables. Use any of the methods below.
  - **Mean Absolute Deviation (MAD)** — The Mean Absolute Deviation method eliminates values that are more than n deviations from the median. You specify the threshold value for the number of deviations, n, in the Cutoff for MAD property.
  - **User-Specified Limits** — The User-Specified Limits method specifies a filter for observations that is based on the interval values that are displayed in the Lower Limit and Upper Limit columns of your data table. You specify these limits in the Interactive Replacement Interval Filter window.
  - **Metadata Limits** — Metadata Limits are the lower and upper limit attributes that you can specify when you create a data source or when you are modifying the Variables table of an Input Data node on the diagram workspace.
  - **Extreme Percentiles** — The Extreme Percentiles method filters values that are in the top and bottom pth percentiles of an interval variable's distribution. You specify the upper and lower threshold value for p in the Cutoff Percentiles for Extreme Percentiles property.
  - **Modal Center** — The Modal Center method eliminates values that are more than n spacings from the modal center. You specify the threshold value for the number of spacings, n, in the Cutoff for Modal Center property.
  - **Standard Deviations from the Mean** — (default setting) The Standard Deviations from the Mean method filters values that are greater than or equal to n standard deviations from the mean. You must use the Cutoff for Standard Deviation property to specify the threshold value that you want to use for n.
  - **None** — Do not filter interval variables.
- **Cutoff Values** — Click the ... button to the right of the Cutoff Values property to open the Cutoff Values window. You use the Cutoff Values window to modify the cutoff values for the various limit methods available in the Default Limits Method property.
  - **MAD** — When you specify Mean Absolute Deviation as your Default Limits Method, you must use the MAD property of the Replacement node to quantify n, the threshold value for the number of deviations from the median value. Specify the number of deviations from the median to be used as cutoff value. That is, values that are that many mean absolute deviations away from the median will be used as the limit values. When set to User-Specified the values specified using the Interval Editor are used. When set to Missing, blanks or missing values are used as the replacement values. Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
  - **Percentiles for Extreme Percentiles** — When you specify Extreme Percentiles as your Default Limits Method, you must use the Percentiles for Extreme

Percentiles property to specify p, the threshold value used to quantify the top and bottom pth percentiles. Permissible values are percentages greater than or equal to 0 and less than 50. (P specifies upper and lower thresholds, 50% + 50% = 100%.) The default value is 0.5, or 0.5%.

  - **Modal Center** — When you specify Modal Center as your Default Limits Method, you must use the Modal Center property to specify the threshold number of spaces n. That is, values that are that many spacings away from the model center will be used as the limit values Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
  - **Standard Deviation** — Use the Standard Deviation property to quantify n, the threshold for number of standard deviations from the mean. That is, values that are that many standard deviations away from the mean will be used as the limit values. Permissible values are real numbers greater than or equal to zero. The default value is 3.0.

Replacement Node Train Properties: Class Variables

- **Replacement Editor** — Use the “Interactive Class Variables Replacement Editor Window” on page 675 to specify new values for specific class levels of variables in the data imported to the Replacement node. Predecessor nodes to the Replacement node must be run before opening the Replacement Editor. Click the ... button to open the Replacement Editor window for the associated data set. The Replacement Editor is a table that lists the levels of all class variables. The level "_UNKNOWN_" enables you to specify a replacement value to use at scoring time when encountering levels not in the training data. The Replacement Editor contains read-only columns for Variable (name), Level, Frequency, Type (character, numeric), Character Raw Value, and Numeric Raw Value. The Replacement Value column can be edited.
- **Unknown Levels** — Use the Unknown Levels property to specify the way that unknown levels for a variable should be handled during scoring. Unknown levels are levels that do not occur in the training data set.

  The Unknown Levels property has the following options:

  - **Ignore** — variable observations that contain unknown levels are not modified. This is the default setting.
  - **Missing Value** — variable observations that contain unknown levels are replaced with SAS missing value notations.
  - **Mode** — variable observations that contain unknown levels are replaced by the most frequent class level, or mode.

Replacement Node Score Properties

The following score properties are associated with the Replacement node:

- **Replacement Values** — Use the Replacement Values property to specify which values should be used as replacement values.
  - **Computed** — replace the variable value with the result of a mathematical computation.
  - **User-Specified** — replace the variable value with a value defined by the user.
  - **Missing** — replace the variable value with the SAS missing value notation.
- **Hide** — Use the Hide property to specify whether original variables should be dropped from the metadata exported by the node.

Replacement Node Report Properties

The following report property is associated with the Replacement node:

- **Replacement Report** — Use the Replacement Report property to specify whether to count the number of observations replaced for each variable. The report is calculated for the training, validation, and test data sets when applicable.

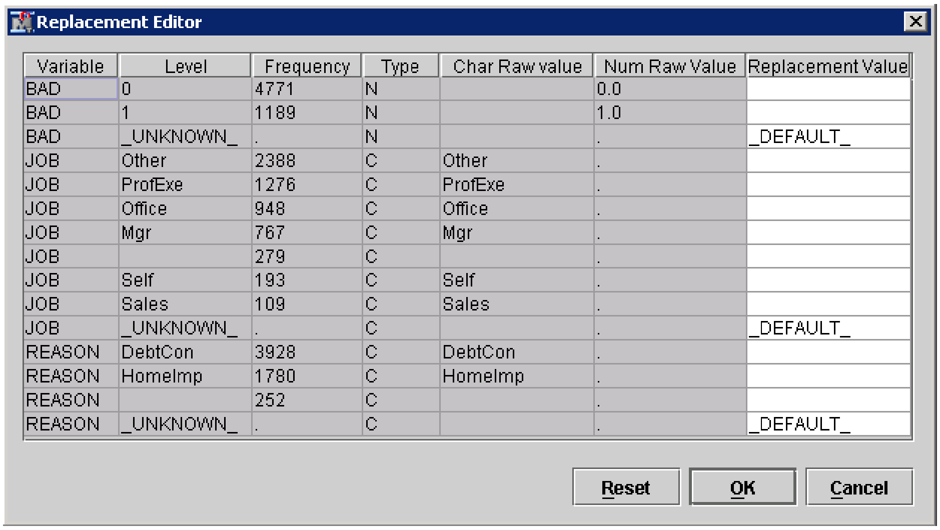
Replacement Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Interactive Class Variables Replacement Editor Window

Use the Replacement Editor to specify replacement values for class variables on a variable-by-variable basis.



Variable	Level	Frequency	Type	Char Raw value	Num Raw Value	Replacement Value
BAD	0	4771	N		0.0	
BAD	1	1189	N		1.0	
BAD	_UNKNOWN_	.	N		.	_DEFAULT_
JOB	Other	2388	C	Other	.	
JOB	ProfExe	1276	C	ProfExe	.	
JOB	Office	948	C	Office	.	
JOB	Mgr	767	C	Mgr	.	
JOB		279	C		.	
JOB	Self	193	C	Self	.	
JOB	Sales	109	C	Sales	.	
JOB	_UNKNOWN_	.	C		.	_DEFAULT_
REASON	DebtCon	3928	C	DebtCon	.	
REASON	HomeImp	1780	C	HomeImp	.	
REASON		252	C		.	
REASON	_UNKNOWN_	.	C		.	_DEFAULT_

You use the Replacement column to enter the replacement value or statistic that you want to use for specific levels of a class variable. By default, unknown levels are replaced by the value specified in the Unknown Levels property.

You can use the following keywords as replacement values in the Replacement column:

- **_DEFAULT_** — use the default value that is defined in the Unknown Levels property.
- **_MISSING_** — replace the variable value with the SAS missing value notation.

- **_MODE_** — replace the variable value with the most frequent class level, or mode.
- **<blank>** — a blank space in the Replacement column indicates that no replacement will occur for the level.

Interactive Replacement Interval Filter Window

Overview

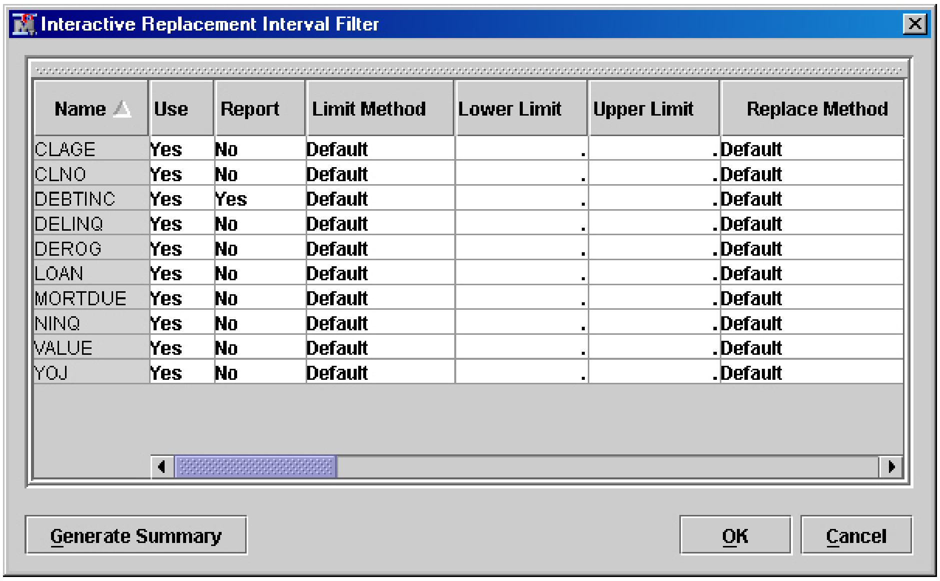
Use the editor to specify replacement values for interval variables on a variable-by-variable basis. You can configure and modify the Use, Report, Limit Method, Lower Limit, Upper Limit, Replace Method, Lower Replacement Value (), and Upper Replacement Value (>) attributes.

Note: The Default value of the Limit Method column is the value specified in the Default Limits Method property. The Default value under the Replace Method column is the value specified in the Replacement Report property.

- **Lower Limit** — a numeric field that can be used to define the lower limit of a variable filter.
- **Upper Limit** — a numeric field that can be used to define the upper limit of a variable filter.

With SAS Enterprise Miner 12.3, you can use the Interactive Interval Filter table to keep only records for a specified single value of an interval variable. To keep only records for a specified value of an interval variable, set both the Upper Limit and Lower Limit columns for that variable to the desired variable value.

The following are read-only attributes: Name, Role, Level, Type, Order, Label, Format, Informat, Length, Lower Limit, Upper Limit, Distribution, Family, Creator, Report, and Comment.

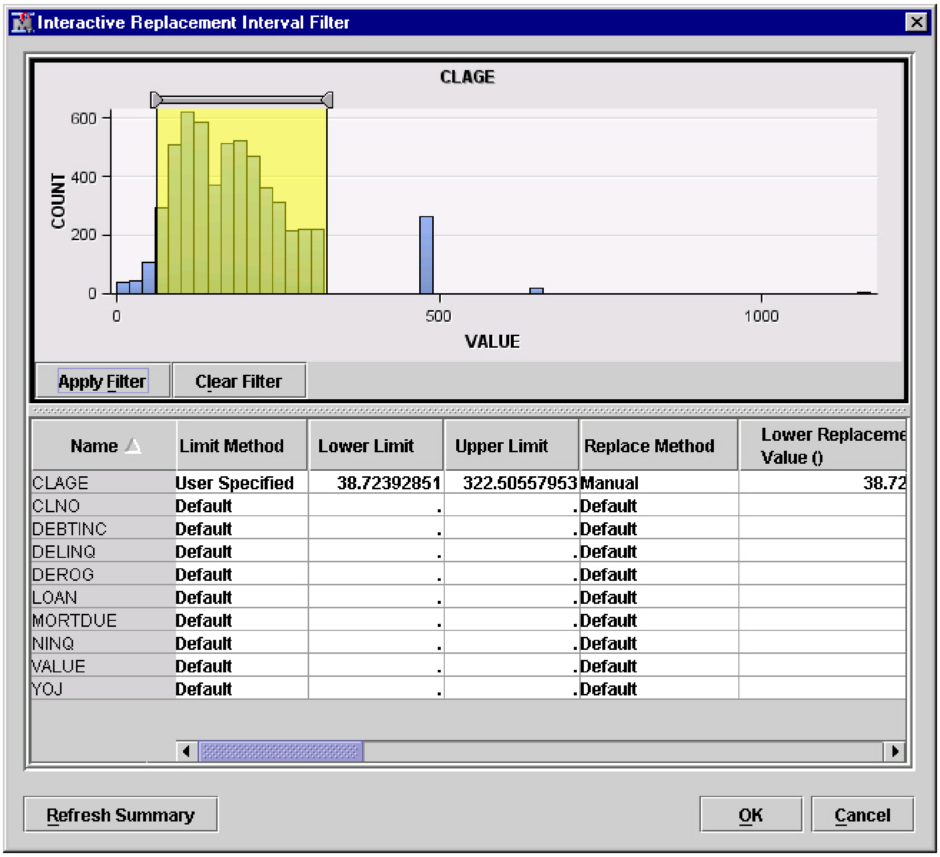


Name	Use	Report	Limit Method	Lower Limit	Upper Limit	Replace Method
CLAGE	Yes	No	Default	.	.	Default
CLNO	Yes	No	Default	.	.	Default
DEBTINC	Yes	Yes	Default	.	.	Default
DELINQ	Yes	No	Default	.	.	Default
DEROG	Yes	No	Default	.	.	Default
LOAN	Yes	No	Default	.	.	Default
MORTDUE	Yes	No	Default	.	.	Default
NINQ	Yes	No	Default	.	.	Default
VALUE	Yes	No	Default	.	.	Default
YOJ	Yes	No	Default	.	.	Default

Interactive Filtering

In addition to the standard limit methods, the Interactive Replacement Interval Filter window provides the additional limit method of User Specified. To interactively select a limit range for a variable, select a variable in the table.

- The variable distribution displayed is based on a summary data set. To generate this data set, click the Generate Summary button.
- The shaded area identifies the range of values to be kept. To modify the filter limits, you can move the sliders at the top of the graph or enter values directly in the Lower Limit and/or Upper Limit fields.
- Click the Apply Filter button to confirm your choices, the selected variable's Limit Method field should change to User Specified, the Replace Method field should change to Manual and the Lower Limit, Upper Limit, Lower Replacement Value (), Upper Replacement Value fields should change to the values corresponding to the movable reference lines on the graph.
- Click on the Clear Filter button, to clear the filter limits.



Name	Limit Method	Lower Limit	Upper Limit	Replace Method	Lower Replaceme Value ()
CLAGE	User Specified	38.72392851	322.50557953	Manual	38.72
CLNO	Default	.	.	Default	
DEBTINC	Default	.	.	Default	
DELINQ	Default	.	.	Default	
DEROG	Default	.	.	Default	
LOAN	Default	.	.	Default	
MORTDUE	Default	.	.	Default	
NINQ	Default	.	.	Default	
VALUE	Default	.	.	Default	
YOJ	Default	.	.	Default	

Replacement Node Results

After a successful node run, you can open the Results window of the Replacement node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Replacement node properties configuration when the node was last run.

- **Run Status** — indicates the status of the Replacement node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
- **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
- **Train Code** — the code that Enterprise Miner used to train the node.
- **Notes** — allows users to read text that was entered via the Replacement node Notes property before the node was run.

- **SAS Results**
  - **Log** — the SAS log of the Replacement node run.
  - **Output** — the SAS output of the Replacement node run. The SAS output displays a variable summary and replacement count statistics for train, validate, and test data sets.
  - **Flow Code** — the SAS code used to produce the output that the Replacement node passes on to the next node in the process flow diagram. Flow code contains replacement values for input and target variables. This includes replacement of unknown levels (if specified) and replacement of specific levels. If no replacement values were generated, the Flow Code menu item is dimmed and unavailable. Replacement of unknown levels will not affect scoring the training data set in the flow, but could affect the scoring of a validation or test data set.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications. SAS Code contains replacement values of input variables only. This includes replacement of unknown levels (if specified) and replacement of specific levels and limits. If no replacement values were generated, the SAS Code menu item is dimmed and unavailable.
  - **PMML Code** — the Replacement node does not generate PMML code.
- **Model**
  - **Interval Variables** — displays a read-only table summarizing the replaced variable levels and the specified replacement settings. If no replacement values were generated, the Replacement Levels menu item is dimmed and unavailable.
  - **Total Replacement Counts** — displays a read-only table summarizing the variable replacement counts for train, validate, and test data sets. If no replacement values were generated, the Replacement Counts menu item is dimmed and unavailable.
- **Plot** — opens the Select a Chart plotting wizard so that you can plot the data in the table that you have in focus.

Chapter 46
Rules Builder Node

Rules Builder Node

Overview of the Rules Builder Node

The **Rules Builder** node is on the **Modify** tab of the SAS Enterprise Miner toolbar. The node accesses the Rules Builder window so that you can create ad hoc sets of rules with user-definable outcomes. You can interactively define the values of the outcome variable and the paths to the outcome. This is useful in ad hoc rule creation such as applying logic for posterior probabilities and scorecard values.

For example, you can create two outcomes named "Approve" and "Deny" and rules such as the following:

```
If Age < 20 then outcome="Deny";
```

You can also create nested IF statements. These rules are stored in the node's exported data sets. The rules can also be saved as SAS files and used to build other scoring processes.

Any Input Data Source data set can be used as an input to the **Rules Builder** node. Rules are defined using charts and histograms based on a sample of the data. The sample is created when you run the **Rules Builder** node, so you must run the node before defining your rules.

In some instances, rules might become out-of-sync if a variable that is used in the rules no longer exists. In that case, you must modify the rules and remove that variable or the **Rules Builder** node will fail when you run it.

Rules Builder Node Properties

Rules Builder Node General Properties

The following general properties are associated with the **Rules Builder** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Rules Builder** node that is added to a diagram has a Node ID of Rule. The second **Rules Builder** node that is added to a diagram has a Node ID of Rule2, and so on.
- **Imported Data** — accesses the Imported Data —Rules Builder window. The Imported Data — Rules Builder window contains information about the imported data's port, source, table name, data role, and whether data exists. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click to select a task:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — Rules Builder window. The Exported Data — Rules Builder window contains information about the exported data's port, table, role, and whether there is data available after the **Rules Builder** node is run. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click as follows:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Rules Builder Node Train Properties

The following train properties are associated with the **Rules Builder** node:

- **Variables** — Use the Variables window to see the status for individual variables. Select the ... button to open a window that contains the variables table. You can specify the Use or Report properties for a variable, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.

  The Variables window enables you to configure metadata, and view a variable's sampling information, observation values, or a plot of variable distribution.

You can use the following buttons to accomplish these tasks:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Rules** — Use the Rules Builder property to create manual rules. Click the ... button to the right of the Rules property to open the Rules Builder window. See "Using the Rules Builder Window" on page 682 for more information.

Rules Builder Node Score Properties

The following score property is associated with the **Rules Builder** node:

- **Outcome Variable Role** — sets the model role that is assigned to the outcome variable. The default outcome identifier is Segment.
  - **Segment** — is a variable that identifies the segment.
  - **ID** — is an indicator variable for every observation in the data set.
  - **Input** — is an independent variable that is used to predict the target.
  - **Target** — is the dependent or response variable.

Rules Builder Node Status Properties

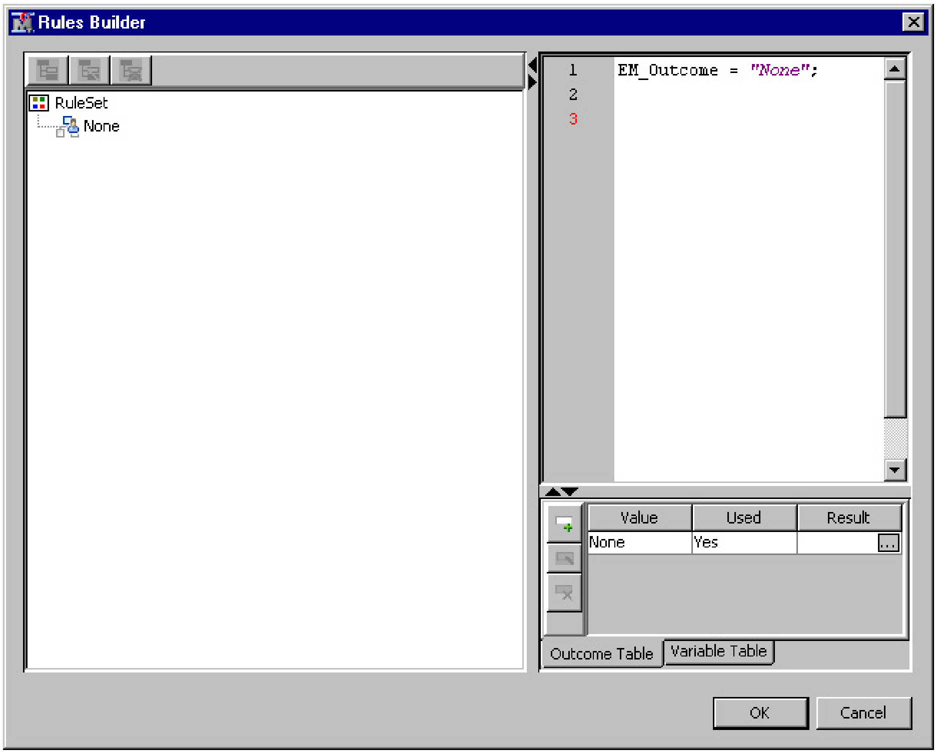
The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Using the Rules Builder Window

Overview

The Rules Builder window enables you to create customized rules from the input variables. Rules are defined using charts and histograms based on a sample of the data. The sample is created when you run the **Rules Builder** node, so you must run the node first before defining your rules. To open the Rules Builder window, click the ... button for the Rules property in the Properties panel. Here is an example display of the Rules Builder window.



```
EM_Outcome = "None";
```

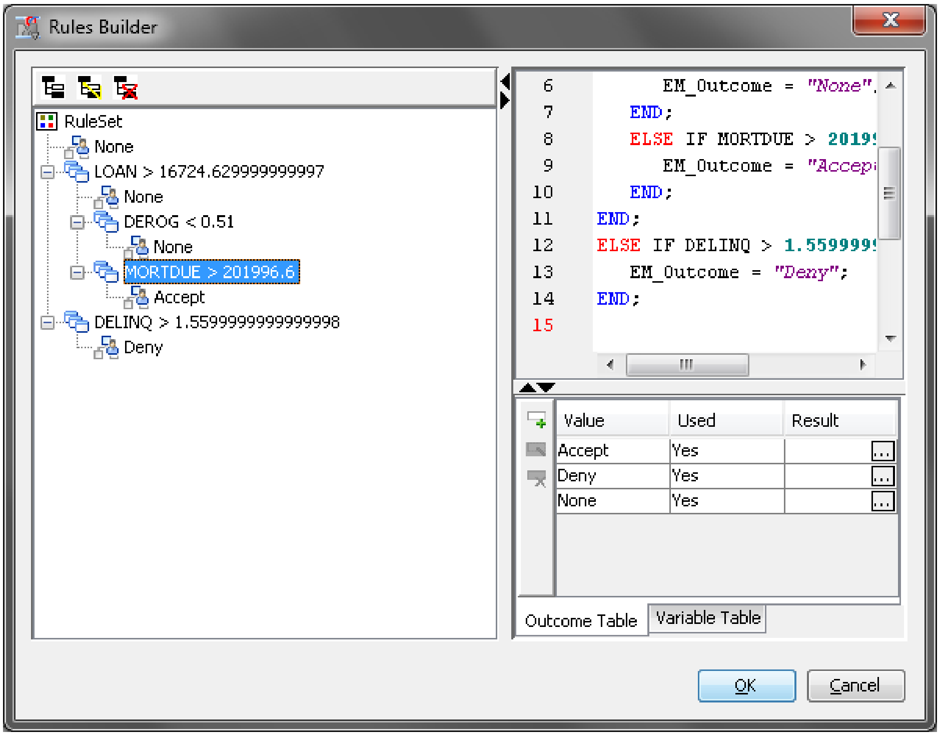
Value	Used	Result
None	Yes	...

Rules Tree

The left pane of the Rules Builder window displays a tree of the rules. The root of the rules tree is named RuleSet.

- To create a new rule, select the RuleSet root node and click the **Add** icon ( ) on top of the table. The Rules Builder wizard opens. To add a nested IF statement, select the parent rule on the left pane and click the **Add** icon ( ) on top of the table. See "Using the Rules Builder Wizard" on page 687 .
- To modify an existing rule, select the node that corresponds to the rule and click the **Edit** icon ( ) on top of the table. The Edit Outcome window appears. See "Using the Rules Builder Wizard" on page 687 .
- To delete an existing rule, select the rule and click the **Delete** icon ( ) on top of the table.

You can drag the rules in the **Rules Tree** in order to adjust your rule set. For example, consider the **Rules Tree** given below.

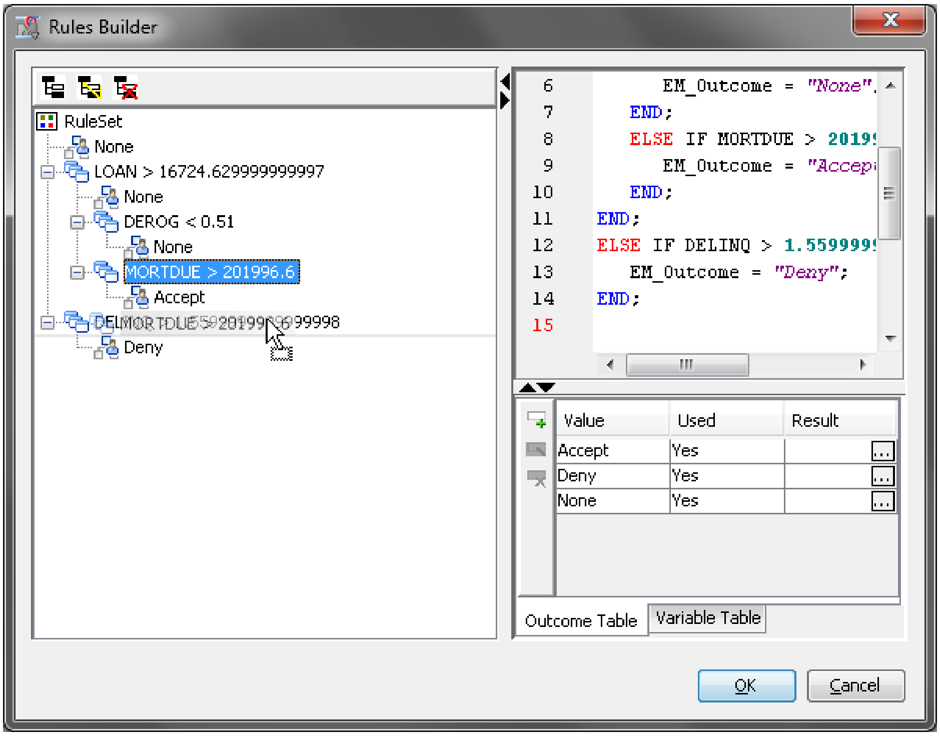


- RuleSet
  - None
  - LOAN > 16724.629999999997
    - None
    - DEROG < 0.51
      - None
    - MORTDUE > 201996.6
      - Accept
  - DELINQ > 1.5599999999999998
    - Deny

```
    EM_Outcome = "None".
  END;
  ELSE IF MORTDUE > 2019!
    EM_Outcome = "Accep
  END;
END;
ELSE IF DELINQ > 1.559999!
  EM_Outcome = "Deny";
END;
```

Value	Used	Result
Accept	Yes	...
Deny	Yes	...
None	Yes	...

Suppose that you want to move the **MORTDUE** rule out of the **LOAN** branch and into the **DELINQ** branch. To do so, left-click the **MORTDUE** rule and drag it to the **DELINQ** rule as shown in the image below. Release the mouse button when you are placing your mouse pointer over the **DELINQ** rule.

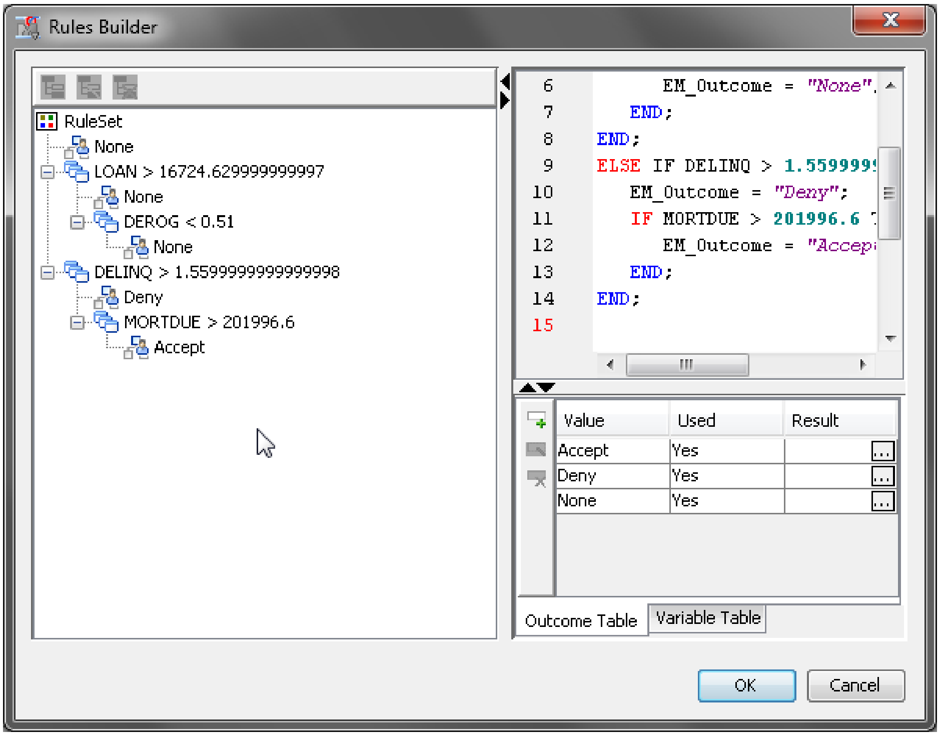


- RuleSet
  - None
  - LOAN > 16724.629999999997
    - None
    - DEROG < 0.51
      - None
    - MORTDUE > 201996.6
      - Accept
  - DELINQ > 1.5599999999999998
    - Deny

```
    EM_Outcome = "None".
  END;
  ELSE IF MORTDUE > 2019!
    EM_Outcome = "Accep
  END;
END;
ELSE IF DELINQ > 1.559999!
  EM_Outcome = "Deny";
END;
```

Value	Used	Result
Accept	Yes	...
Deny	Yes	...
None	Yes	...

The final result is that the **MORTDUE** rule has been moved to the **DELINQ** branch, as shown below.



- RuleSet
  - None
  - LOAN > 16724.629999999997
    - None
    - DEROG < 0.51
      - None
  - DELINQ > 1.5599999999999998
    - Deny
    - MORTDUE > 201996.6
      - Accept

```
        EM_Outcome = "None"
     END;
END;
ELSE IF DELINQ > 1.559999
     EM_Outcome = "Deny";
     IF MORTDUE > 201996.6
        EM_Outcome = "Accep
     END;
END;
```

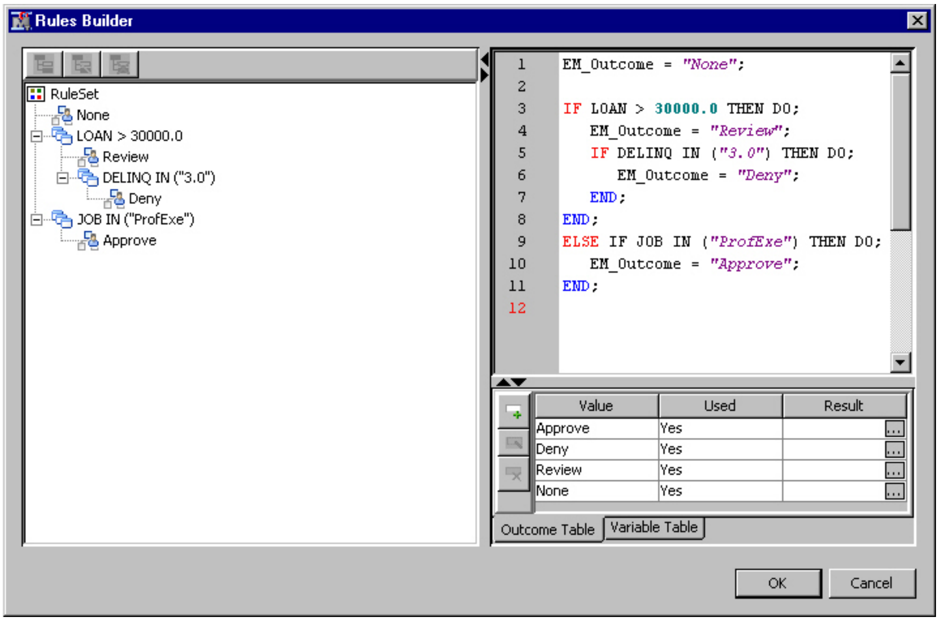
Value	Used	Result
Accept	Yes	...
Deny	Yes	...
None	Yes	...

In order to move a rule to the root of the **Rules Tree**, you must drag that rule to the **RuleSet** root node.

Because the **Rules Builder** node does not permit duplicate rules within a branch, you receive an error if you move a rule into a branch that already contains an identical rule.

Code Pane

The code editor in the upper right pane displays the textual representation of the rules.

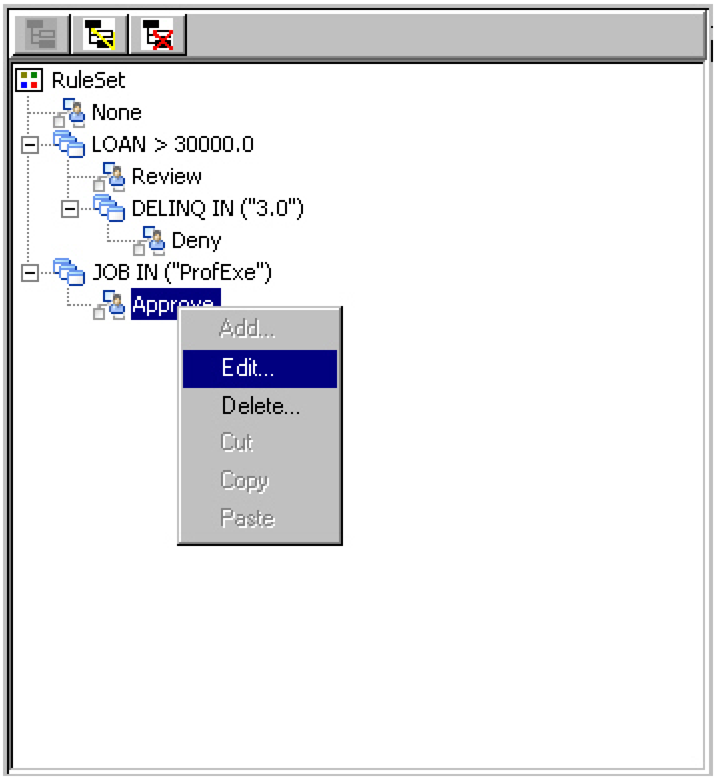


- RuleSet
  - None
  - LOAN > 30000.0
    - Review
    - DELINQ IN ("3.0")
      - Deny
  - JOB IN ("ProfExe")
    - Approve

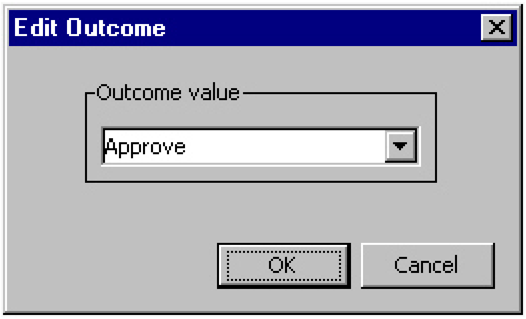
```
EM_Outcome = "None";

IF LOAN > 30000.0 THEN DO;
   EM_Outcome = "Review";
   IF DELINQ IN ("3.0") THEN DO;
      EM_Outcome = "Deny";
   END;
END;
ELSE IF JOB IN ("ProfExe") THEN DO;
   EM_Outcome = "Approve";
END;
```

Value	Used	Result
Approve	Yes	...
Deny	Yes	...
Review	Yes	...
None	Yes	...

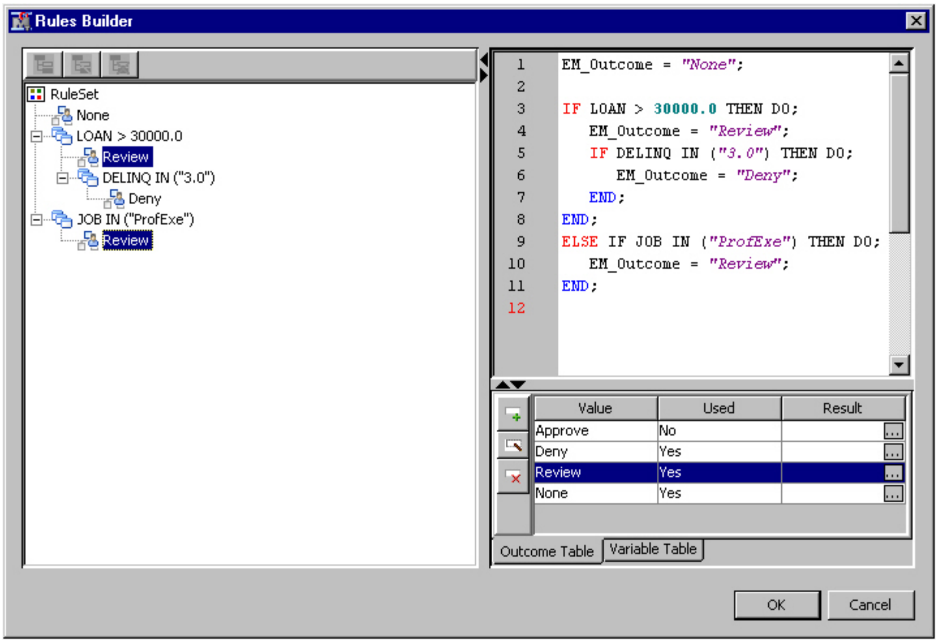
You can quickly change the outcome with a rule. For example, to change the **Approve** rule, select the **Approve** rule from the RuleSet tree on the left pane. Right-click **Approve** and select **Edit** from the popup menu.



In the Edit Outcome window, select the Outcome value and click **OK**.



The new outcome is reflected in the RulesTree and the Rules Code pane. The following illustrates a selection of Review from the drop-down menu in the **Edit Outcome** window.



RuleSet
- None
- LOAN > 30000.0
  - Review
  - DELINQ IN ("3.0")
    - Deny
- JOB IN ("ProfExe")
  - Review

```
EM_Outcome = "None";

IF LOAN > 30000.0 THEN DO;
   EM_Outcome = "Review";
   IF DELINQ IN ("3.0") THEN DO;
      EM_Outcome = "Deny";
   END;
END;
ELSE IF JOB IN ("ProfExe") THEN DO;
   EM_Outcome = "Review";
END;
```

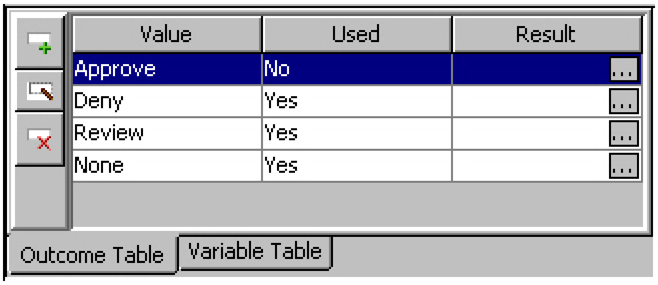
Value	Used	Result
Approve	No	...
Deny	Yes	...
Review	Yes	...
None	Yes	...

Outcome Table

The Outcome Table displays the values that are defined for the EM_Outcome variable.

The table has the following columns:

- Value — the value of the rule.
- Used — indicates whether the value is used or not.
- Result — a list of RuleSet names where the outcome is used. Select the ... button to open a window that contains the Used Values dialog box that lists the name of the rule nodes that use the value.



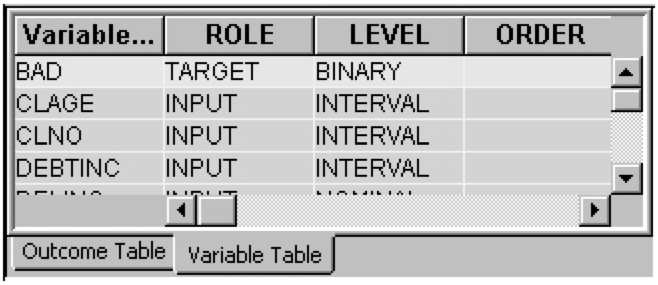
Value	Used	Result
Approve	No	...
Deny	Yes	...
Review	Yes	...
None	Yes	...

You can add, edit, or delete an outcome.

- To create a new outcome, click the **Add outcome** icon ( ) on the top left of the table. The **Add Outcome** window appears. Specify a value in the **Outcome value** field and click **OK**. The new value appears in the Outcome Table.
- To modify an existing outcome, select the row that corresponds to the outcome and click the **Edit outcome** icon ( ) on the left side of the table. The **Edit Outcome** window appears. Modify the value in the **Outcome value** field and click **OK**. The modified value appears in the Outcome table.
- To delete an outcome, select the row that corresponds to the outcome and click the **Delete outcome** icon ( ).

Variable Table

The Variable Table shows the columns of the input data source that is connected to the **Rules Builder** node. The columns are read-only.

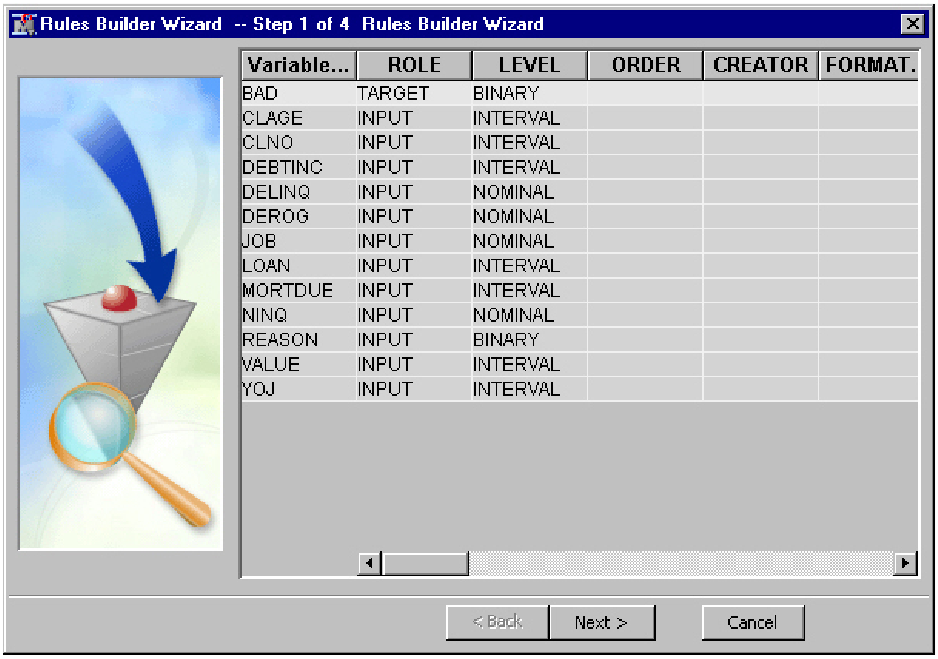


Variable...	ROLE	LEVEL	ORDER
BAD	TARGET	BINARY	
CLAGE	INPUT	INTERVAL	
CLNO	INPUT	INTERVAL	
DEBTINC	INPUT	INTERVAL	

Using the Rules Builder Wizard

Use the Rules Builder wizard to create or edit rules. To open the Rules Builder wizard, click **Add** or **Edit** icons from the Rules Builder window.

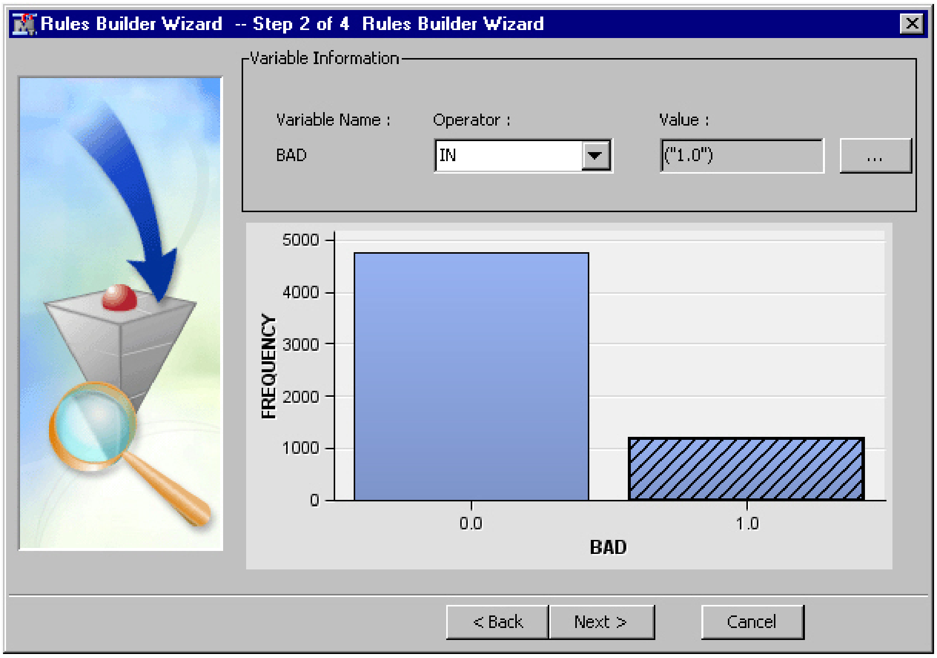
1. Select a single variable from the table and click **Next**.



Rules Builder Wizard -- Step 1 of 4 Rules Builder Wizard

Variable...	ROLE	LEVEL	ORDER	CREATOR	FORMAT.
BAD	TARGET	BINARY			
CLAGE	INPUT	INTERVAL			
CLNO	INPUT	INTERVAL			
DEBTINC	INPUT	INTERVAL			
DELINQ	INPUT	NOMINAL			
DEROG	INPUT	NOMINAL			
JOB	INPUT	NOMINAL			
LOAN	INPUT	INTERVAL			
MORTDUE	INPUT	INTERVAL			
NINQ	INPUT	NOMINAL			
REASON	INPUT	BINARY			
VALUE	INPUT	INTERVAL			
YOJ	INPUT	INTERVAL			

2. Select the value for the displayed variable and select the appropriate operator. For a nominal or binary variable, you can set the value by clicking the ... button to the right of the **Value** field and selecting a value from the Values Table. Alternatively, you can select a bar from the histogram.

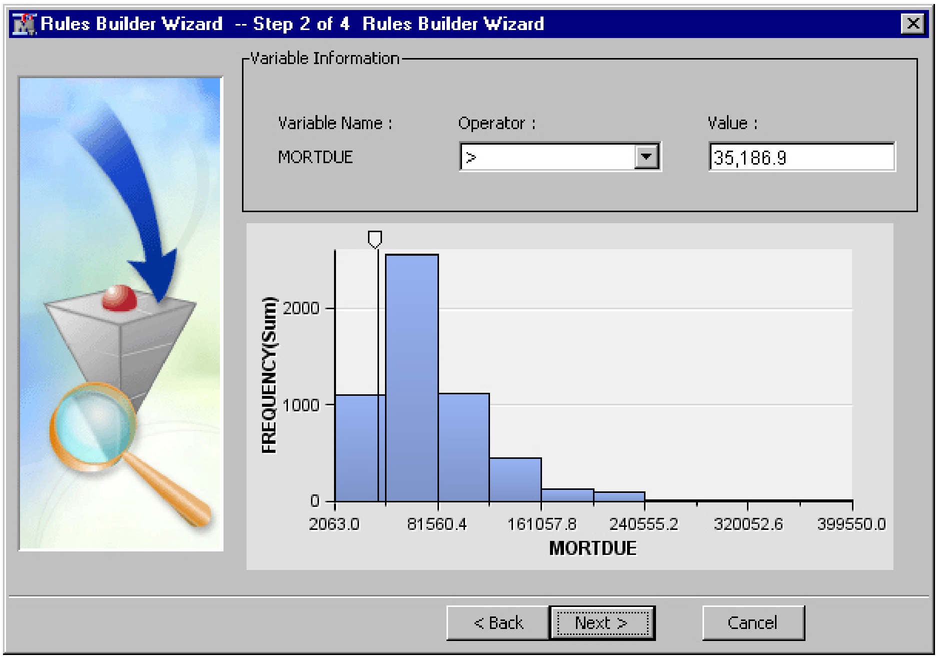


Variable Name : BAD

Operator : IN

Value : ("1.0")

For an interval variable, you can set the **Value** field by entering it or by dragging the movable reference line button on top of the histogram.



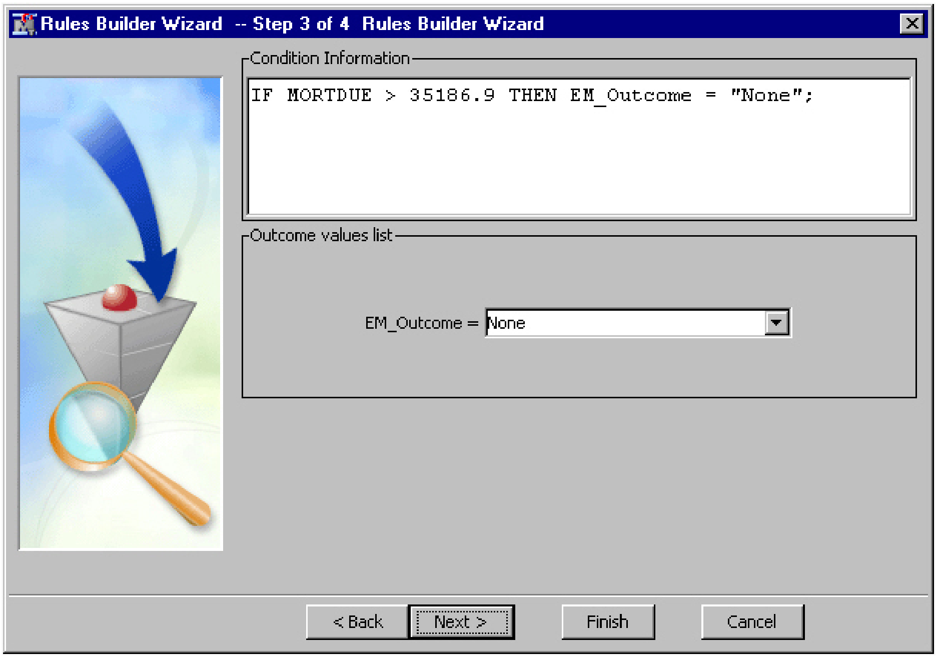
Variable Name : MORTDUE

Operator : >

Value : 35,186.9

Click **Next**.

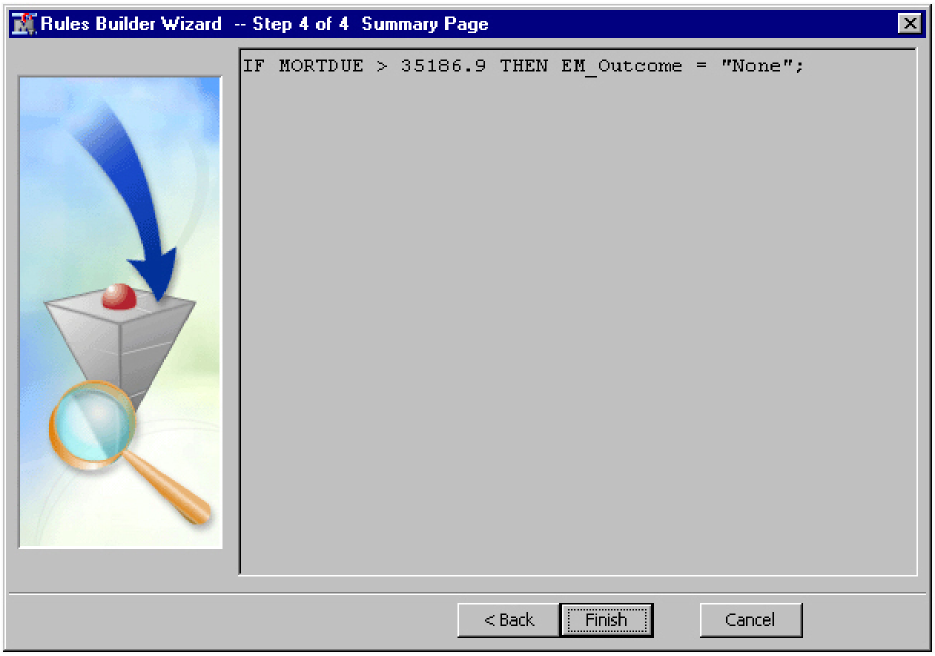
3. The rule is displayed in the Condition Information area. Select the value for EM_Outcome from the drop-down list. Click **Next**.



```
IF MORTDUE > 35186.9 THEN EM_Outcome = "None";
```

EM_Outcome = None

4. The rule is displayed in the Summary Page window. Click **Finish**. The new or edited rule appears in the Rules Builder window.



```
IF MORTDUE > 35186.9 THEN EM_Outcome = "None";
```

Rules Builder Node Results

You can open the Results window of the **Rules Builder** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

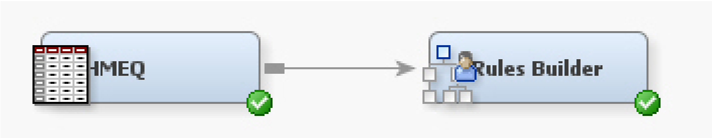
- **Properties**

  - **Settings** — displays a window with a read-only table of the configuration information in the Rules Builder Node Properties panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the **Rules Builder** node run. Information about whether the run completed successfully, the Run Start Time, Run Duration, and the Run ID are displayed in this window.
  - **Variables** — a read-only table of variable metadata information about the data set submitted to the **Rules Builder** node . The table includes columns to see the variable name, the variable role, the variable level, and other statistical information.
  - **Train Code** — the code that SAS Enterprise Miner uses to train the node.
  - **Notes** — enables you to read the notes associated with this node.
- **SAS Results**
  - **Log** — the SAS log of the **Rules Builder** node's run.
  - **Output** — the SAS output of the **Rules Builder** node's run.
  - **Flow Code** — the SAS code used to produce the output that the **Rules Builder** node passes to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **Rules Builder** node does not generate PMML code.
- **Custom Reports**
  - **Scored Sample** — a table of the scored sample from the input data source's exported table. The table contains the EM_OUTCOME column. From the Exported Data property of the **Rules Builder** node, the table name of the scored sample is EMWS.RULE_TRAIN.
  - **Distribution of EM_Outcome** — a bar chart of the occurrence of outcomes.

Rules Builder Node Example

This example assumes that you have already created and opened both a project and a diagram to contain this example. For information on how to create a project, see Creating a New Project on page 226 . For information on how to create a diagram, see Create a New Project Diagram on page 230 .

1. From the SAS Enterprise Miner menu, select **Help** ⇨ **Generate Sample Data Sources** to create sample data sources that you will use for the examples below. In the Generate Sample Data Sources window, deselect everything except the **Home Equity** data set and click **OK**. The example data is located in the SAS sample library SAMPSIO.
2. To create a new process flow diagram, drag the HMEQ data source into the Diagram Workspace. Drag a **Rules Builder** node from the **Modify** tab of the node toolbar and connect this node to the HMEQ data source node. Right-click the **Rules Builder** node and select **Run**. In the Confirmation window, select **Yes**. After a successful run of the **Rules Builder** node, select **OK** in the Run Status window.



3. Select the **Rules Builder** node in the diagram and click the ... button to the right of the **Rules** property to open the Rules Builder window.

4. Select the **Outcome Table** tab on the Rules Builder window and click the **Add outcome** icon ( ) at the top left of the Outcome Table. The Add Outcome window appears. Enter **`Review`** in the **Outcome value** field and click **OK**. The new Review outcome value appears in the Outcome Table. In a similar fashion, add **`Deny`** and **`Approve`** outcomes.

5. In the left pane of the Rules Builder window, select the RuleSet root and click the **Add** icon ( ) on top of the table. The **Rules Builder** wizard opens.

   a. Select the LOAN variable from the variable table of the Rules Builder — Step 1 window. Click **Next**.

   b. Select the greater-than operator (>) and enter **`30000`** in the **Value** field of the Rules Builder Wizard — Step 2 window. Click **Next**.

   c. Select the **Review** item from the EM_Outcome **Outcome values** list. The Condition Information view displays the following code:

   ```
   IF LOAN > 30000 THEN EM_Outcome = "Review";
   ```

   Click **Next**.

   d. The Rules Builder Summary page opens. Click **Finish**. The rule LOAN > 30000 appears under RuleSet in the Rules Builder window.

6. Select the LOAN > 30000 node and click the **Add** icon ( ). The **Rules Builder** wizard opens.

   a. Select the DELINQ variable from the variable table of the Rules Builder Wizard — Step 1 window. Click **Next**.

   b. Select the greater-than operator (>) and enter **`3`** in the **Value** field of the Rules Builder Wizard — Step 2 window. Click **Next**.

   c. Select the **Deny** item from the EM_Outcome **Outcome values** list. The Condition Information view displays the following code:

   ```
   IF DELINQ > 3.0 THEN EM_Outcome = "Deny";
   ```

   Click **Next**.

   d. The Rules Builder Summary page opens. Click **Finish**. The rule DELINQ > 3.0 appears under the Loan Rule in the Rules Builder window.

7. Select the RuleSet root and click the **Add** icon ( ) on top of the table. The **Rules Builder** wizard opens.

   a. Select the JOB variable from the variable table of the Rules Builder Wizard — Step 1 window. Click **Next**.

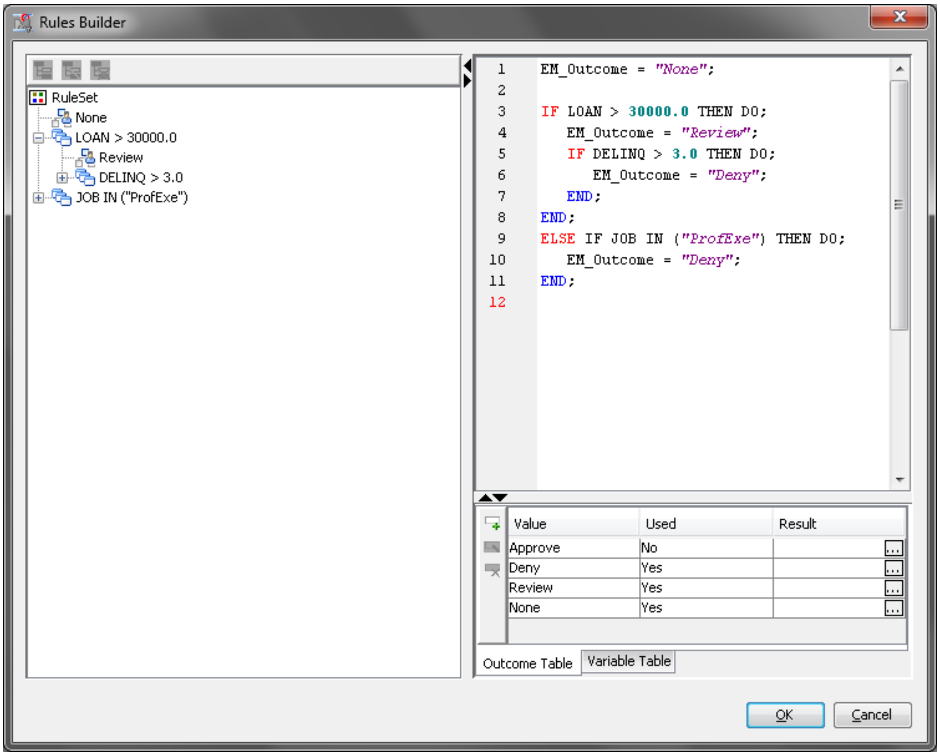
   b. Select the ... button by the Value textbox to open the Values Table window. Select **ProfExe** and click **OK**. Click **Next**.

c. Select the **Approve** item from the EM_Outcome **Outcome values** list. The Condition Information view displays the following code:

```
IF JOB IN ("ProfExe") THEN EM_Outcome = "Approve";
```

Click **Next**.

d. The Rules Builder Summary page opens. Click **Finish**. The rule JOB IN ("ProfExe") appears under RuleSet in the Rules Builder window.



```
RuleSet
    None
    LOAN > 30000.0
        Review
        DELINQ > 3.0
    JOB IN ("ProfExe")
```

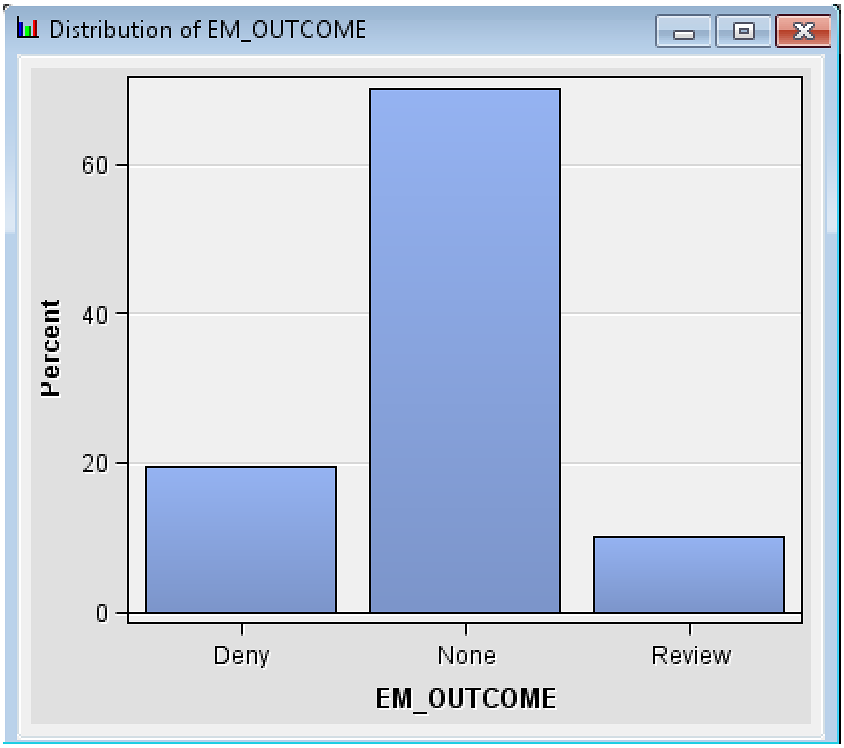
```
EM_Outcome = "None";

IF LOAN > 30000.0 THEN DO;
   EM_Outcome = "Review";
   IF DELINQ > 3.0 THEN DO;
      EM_Outcome = "Deny";
   END;
END;
ELSE IF JOB IN ("ProfExe") THEN DO;
   EM_Outcome = "Deny";
END;
```

Value	Used	Result
Approve	No	...
Deny	Yes	...
Review	Yes	...
None	Yes	...

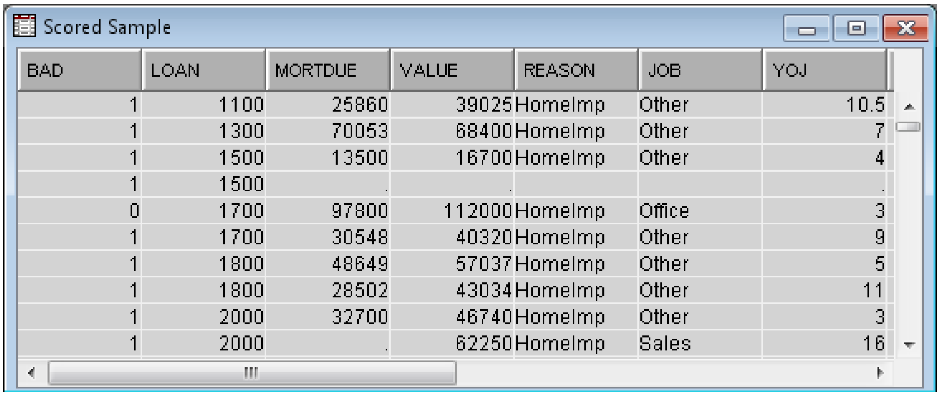
In the Rules Builder window, click **OK**.

8. Right-click the **Rules Builder** node, and select **Run** to run the process flow diagram. When the Run Status window indicates that the run is completed, click **Results** to open the Rules Builder Results window.

   - Examine the distribution of EM_OUTCOME.

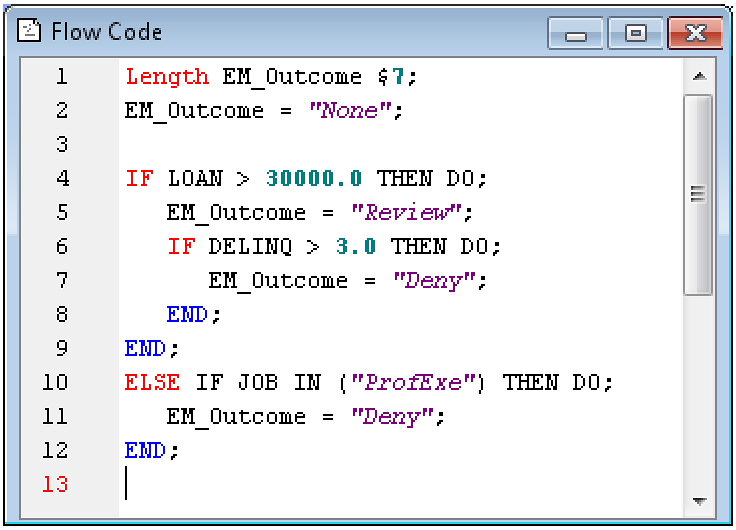


- Examine the Scored Sample table. The scored sample can be used to create ad hoc reports using the Select a Chart Type window by selecting the Scored Sample table and selecting **View** ⇨ **Plot** from the Results window.



BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ
1	1100	25860	39025	HomeImp	Other	10.5
1	1300	70053	68400	HomeImp	Other	7
1	1500	13500	16700	HomeImp	Other	4
1	1500	.	.			.
0	1700	97800	112000	HomeImp	Office	3
1	1700	30548	40320	HomeImp	Other	9
1	1800	48649	57037	HomeImp	Other	5
1	1800	28502	43034	HomeImp	Other	11
1	2000	32700	46740	HomeImp	Other	3
1	2000	.	62250	HomeImp	Sales	16

- Select **View** ⇨ **SAS Results** ⇨ **Flow Code** to examine the flow code.



```
Length EM_Outcome $7;
EM_Outcome = "None";

IF LOAN > 30000.0 THEN DO;
   EM_Outcome = "Review";
   IF DELINQ > 3.0 THEN DO;
      EM_Outcome = "Deny";
   END;
END;
ELSE IF JOB IN ("ProfExe") THEN DO;
   EM_Outcome = "Deny";
END;

```

Chapter 47

Transform Variables Node

Transform Variables Node

Overview of the Transform Variables Node

The Transform Variables node enables you to create new variables or new variables that are transformations of existing variables in your data. The Transform Variables node also enables you to transform class variables and to create interaction variables. Transformations are useful when you want to improve the fit of a model to the data. For example, transformations can be used to stabilize variances, remove nonlinearity, improve additivity, and correct nonnormality in variables. The “References” on page 716 section provides a list of documents that discuss many of the reasons for making transformations and the methods that are used to choose the appropriate transformation.

The Transform Variables node can be connected to any predecessor node that exports a Raw or Train data set. To transform a transformed variable, you should add another Transform Variables node to the process flow. For example, you may want to transform the target and then transform selected input variables to maximize their relationship with the transformed target.

If you have several missing values for one or more variables, you may want to impute missing values with either the Impute node or the Cluster node, before you transform variables.

Transform Variables Node Properties

Transform Variables Node General Properties

The following general properties are associated with the Transform Variables node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Transform Variable node added to a diagram will have a Node ID of Trans. The second Transform Variable node added to the diagram will have a Node ID of Trans2.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Transform Variables window. The Imported Data — Transform Variables window contains a list of the ports that provide data sources to the Transform node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Transform Variables window. The Exported Data — Transform Variables window contains a list of the output data ports that the Transform node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Transform Variables Node Train Properties

The following train properties are associated with the Transform Variables node:

- **Variables** — Use the Variables property to specify the properties of each variable that you want to use in the data source. Select the ... button to open a variables table. You set the transformation method in the Variables table. For more detailed information, see "Setting the Transformation Method" on page 706 .
- **Formulas** — Use the Formulas property to create customized transformations. Select the ... button to open the Formula Builder on page 708 .

- **Interactions** — Use the Interactions property to create interaction variables on page 713 . Select the ... button to open the Terms window.

- **SAS Code** — Select the ... button to open the SAS Code window. You enter SAS code statements to create your own custom variable transformation in the SAS Code window. If you want to use code from a SAS catalog or external file, use the SAS Code window to submit a filename statement and a %include statement.

Transform Variables Node Train Properties: Default Methods

The following default method properties are associated with the Transform Variables node. See "Transformation Methods" on page 701 for more information about the transformation methods.

- **Interval Inputs** — Use the Interval Inputs property to specify the default transformation method that you want to apply to interval input variables. Each variable in the Variables table that uses the Default method will use the method that you specify in this property. The available methods are

  - **Best** — performs several transformations and uses the transformation that has the best Chi Squared test for the target.
  - **Multiple** — creates several transformations intended for use in successor Variable Selection nodes.
  - **Log** — transformed using the logarithm of the variable.
  - **Log 10** — transformed using the base-10 logarithm of the variable.
  - **Square Root** — transformed using the square root of the variable.
  - **Inverse** — transformed using the inverse of the variable.
  - **Square** — transformed using the square of the variable.
  - **Exponential** — transformed using the exponential logarithm of the variable.
  - **Centering** — centers variable values by subtracting the mean from each variable.
  - **Standardize** — standardizes the variable by subtracting the mean and dividing by the standard deviation.
  - **Range** — transformed using a scaled value of a variable equal to (x - min) / (max - min), where x is current variable value, min is the minimum value for that variable, and max is the maximum value for that variable.
  - **Bucket** — buckets are created by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
  - **Quantile** — data is divided into groups with approximately the same frequency in groups.
  - **Optimal Binning** — data is binned in order to maximize the relationship to the target.
  - **Maximum Normal** — a best power transformation to maximize normality.
  - **Maximum Correlation** — a best power transformation to maximize correlation to the target. No transformation occurs if used on data sets with non-interval targets.
  - **Equalize** — a best power transformation to equalize spread with target levels.
  - **Optimal Max. Equalize** — an optimized best power transformation to equalize spread with target levels.

  - **None** — (default setting) No transformation is performed.

- **Interval Targets** — Use the Interval Targets property to specify the default transformation method that you want to use for interval target variables. Each interval target variable in the Variables table that uses the Default method will use the method that you specify in this property. The available methods are
  - **Best** — tries several transformations and selects the one that has the highest Chi Squared test with the target.
  - **Log** — transformed using the logarithm of the variable.
  - **Log 10** — transformed using the base-10 logarithm of the variable.
  - **Square Root** — transformed using the square root of the variable.
  - **Inverse** — transformed using the inverse of the variable.
  - **Square** — transformed using the square of the variable.
  - **Exponential** — transformed using the exponential logarithm of the variable.
  - **Centering** — centers variable values by subtracting the mean from each variable.
  - **Standardize** — standardizes the variable by subtracting the mean and dividing by the standard deviation.
  - **Range** — transformed using a scaled value of a variable equal to (x - min) / (max - min), where x is current variable value, min is the minimum value for that variable, and max is the maximum value for that variable.
  - **Bucket** — buckets are created by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
  - **Quantile** — data is divided into groups with approximately the same frequency in groups.
  - **Optimal Binning** — data is binned in order to maximize the relationship to the target.
  - **None** — (default setting) No transformation is performed.

- **Class Inputs** — Use the Class Inputs property to specify the default transformation method that you want to use for class input variables. The available methods are
  - **Group rare levels** — transformed using rare levels.
  - **Dummy Indicators** — transformed using dummy variables. Dummy variable coding is applied to the categorical variables from highest class value to lowest class value. For a variable x that contains values from 1 to 7, the transformation dummy variables that correspond with x=1 are named ti_x7, transformation dummy variables that correspond with x=2 are named ti_x6, and transformation dummy variables that correspond with x=7 are named ti_x1.
  - **None** — (default) No transformation is performed.

- **Class Targets** — Use the Class Targets property to specify the default transformation method that you want to use for class target variables.
  - **Group rare levels** — transformed using rare levels.
  - **Dummy Indicators** — transformed using dummy variables.
  - **None** — (default) no transformation is performed

- **Treat Missing as Level** — Use the Treat Missing as Level property to indicate whether missing values should be treated as levels when creating groupings. If the

Treat Missing as Level property is set to Yes, a group value will be assigned to missing values of the variable. If the Treat Missing as Level property is set to No, the group value will be set to missing for each variable.

Transform Variables Node Train Properties: Sample Properties

The Transform Variables node uses a sample to display the graphical distribution of variables in the Formula Builder. Use the following properties to specify how the sample is created.

- **Method** — Use the Method property to specify the sampling method that is used to create a sample.

  The following methods are available:

  - **First N** — When First N is selected, the top n observations are selected from the input data set for the sample. (Default)
  - **Random** — When random sampling is selected, each observation in the data set (population) has the same probability of being selected for the sample, independently of the other observations that happen to fall into the sample.
- **Size** — Use the Size property to specify the number of observations that are used to generate the plot.
  - **Default** — The default value depends on the record length.
  - **Max** — The Max value uses the maximum number of observations that are downloadable. To change the Size value from Default to Max, the Method property must be set to Random.
- **Random Seed** — Use the Random Seed property to specify the seed that is used to generate the random sample. The default value is 12345.

Transform Variables Node Train Properties: Optimal Binning

- **Number of Bins** — Use the Number of Bins property to specify the number of bins to use when performing optimal binning transformations.
- **Missing Values** — Use the Missing property to specify how to handle missing values when you use a optimal binning transformation. Select from the any of the available missing value policies.
  - **Separate Branch** — assigns missing values to its own separate branch.
  - **Use in Search** — uses missing values during the split search.
  - **Largest Branch** — assigns the observations that contain missing values to the branch that has the largest number of training observations.
  - **Branch with Smallest Residual** — assigns the observations that contain missing values to the branch that has the smallest value of residual sum of squares.

Transform Variables Node Train Properties: Grouping Method

- **Cutoff Value** — Group the levels with a very small occurrence (less than the specified cutoff percentage) in an _OTHER_ category. The default setting is 0.1, meaning all levels that occurs less than 10% will be grouped into the _OTHER_ category.
- **Group Missing** — Indicates whether or not missing values should be grouped into the _OTHER_ category with rare levels. The default setting is **No**.
- **Number of Bins** — Use the Number of Bins property to specify the number of bins to use when performing bucket or quantile transformations.

- **Add Minimum Value to Offset Value** — The Add Minimum Value to Offset Value property is useful when transforming data sets that contain negative variable values. Negative variable values are incompatible with some transformation functions and can result in missing value entries in the output transformed data. If the data set to be transformed contains negative values, use the Add Minimum Value to Offset Value property to shift input data values by the absolute value of the most negative variable value.

  In its default configuration of **Yes**, the Add Minimum Value to Offset Value property shifts the most negative variable value in a pre-transformed data set from a negative number to 0.0. When the Offset Value property is configured in its default setting of 1.0, the two properties together will shift variable values so that the smallest variable value in the data to be transformed is 1.0. This results in transformations with the smallest possible number of preventable missing variable entries. When the Add Minimum Value to Offset Value property is set to **No**, the pre-transformation data is only shifted by the value that is specified in the Offset Value property.

- **Offset Value** — The Offset Value property is useful when transforming data sets that contain negative variable values. Negative variable values are incompatible with some transformation functions and can result in missing value entries in the output transformed data. If the data set to be transformed contains negative values, use the Offset Value property to shift input data values. The Offset Value property and the Add Minimum Value to Offset Value property are additive when used together.

  If Add Minimum Value to Offset Value property is set to Yes and the Offset Value property is set to 1.0, the pre-transform data will be shifted so that the smallest variable value to be transformed is 1.0. If Add Minimum Value to Offset Value property is set to Yes and the Offset Value property is set to 2.0, the pre-transform data will be shifted so that the smallest variable value to be transformed is 2.0. If the Add Minimum Value to Offset Value property is set to No, the pre-transform data is only shifted by the amount specified in the Offset Value property.

Transform Variables Node Score Properties

The following score properties are associated with the Transform Variables node:

- **Use Meta Transformation** — Use the Use Meta Transformation property to specify whether to use transformations that are defined in the Enterprise Miner Meta library.
- **Hide** — Use the Hide property to specify how to handle the original variables after a transformation is performed. Setting the Hide property to **No** if you want to keep the original variables in the exported metadata from your transformed data set. The default setting for the Hide property is **Yes**. When this property is set to **Yes**, the original variables are only removed from the exported metadata, and not removed from the exported data sets and data views.
- **Reject** — Use the Reject property to specify whether the model role of the original variables should be changed to Rejected or not. The default value for this property is **Yes**. To change the Reject value from **Yes** to **No**, you must set the Hide property value to **No**.

Transform Variables Node Report Properties

The following report property is associated with the Transform Variables node:

- **Summary Statistics** — Use the Summary Variables property to specify which variables have summary statistics computed. The default setting of **Yes** generates summary statistics on the transformed and new variables in the data set. The **No** setting does not generate any summary statistics.

Transform Variables Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Transformation Methods

Overview

The Transform Variables node supports various computed transformation methods. The available methods depend on the type and the role of a variable.

- For interval variables:
  - "Simple Transformations" on page 701
  - "Binning Transformations" on page 702
  - "Best Power Transformations" on page 702
- For class variables:
  - "Group Rare Levels Transformations" on page 706
  - "Dummy Indicators Transformation" on page 706

Simple Transformations

You can choose from the following simple transformations in the Transform Variables node:

- Log — Variable is transformed by taking the natural log of the variable.
- Square Root — Variable is transformed by taking the square root of the variable.
- Inverse — Variable is transformed by using the inverse of the variable.
- Square — Variable is transformed by using the square of the variable.
- Exponential — Variable is transformed by using the exponential logarithm of the variable.
- Standardize — Variable is standardized by subtracting the mean and dividing by the standard deviation.

Note: If the minimum value of a variable to be transformed is less than or equal to zero, then you should configure the node to add an offset value to the variable before transforming it. When you add an offset value to a variable, you specify a constant value that is added to every observation for that variable. The offset shift prevents illegal mathematic operations during transformation, such as dividing by zero or taking the square root of a negative number. When the node attempts to transform a variable with illegal values, the operation creates a missing variable value in your output transformed data set. Use a properly configured offset value to avoid preventable missing values in your transformed data output.

In most cases, you should offset variable values in an untransformed data set so that the smallest value of a variable is equal to 1. If the data set to be transformed contains negative variable values, you can use the Offset Value and Add Minimum Value properties to configure the Transform node for the range of values in your data. Use the default Offset Value property setting of 1.0 and the default Add Minimum Value setting of Yes to minimize instances of preventable "missing value" data in your transformed data set.

Binning Transformations

Binning transformations enable you to collapse an interval variable, such as debt to income ratio, into an ordinal grouping variable. There are three types of binning transformations.

- Bucket on page 703 — Buckets are created by dividing the data values into equally spaced intervals based on the difference between the maximum and the minimum values.
- Quantile on page 703 — Data is divided into groups that have approximately the same frequency in each group.
- “Optimal Binning for Relationship to Target Transformation” on page 703 — Data is binned in order to optimize the relationship to the target.

Best Power Transformations

The best power transformations are a subset of the general class of transformations that are known as Box-Cox transformations.

The Transform Variables node supports the following best power transformations:

- “Maximize Normality Power Transformation” on page 703 — This method chooses the transformation that yields sample quantiles that are closest to the theoretical quantiles of a normal distribution. This method requires an interval target.
- “Maximize Correlation with Target Power Transformation” on page 704 — This method chooses the transformation that has the best squared correlation with the target. This method requires an interval target. If the maximize correlation transformation is attempted with a non-interval target, the data will not be transformed.
- “Equalize Spread with Target Levels Power Transformation” on page 705 — This method chooses the transformation that has the smallest variance of the variances between the target levels. This method requires a class target.
- Optimal Maximum Equalize Spread with Target Level on page 705 — This method chooses the transformation that equalizes spread with target levels. This method requires a class target.

All of the Best Power transformations evaluate the transformation subset listed below, and choose the transformation that has the best results for the specified criterion. In the list below, x represents the transformed variable.

- x
- log(x)
- sqrt(x)

- e^x
- $x^{1/4}$
- x^2
- x^4

Note: Variables are scaled before they are transformed by the power transformations. Variables are scaled so that their values range from 0 to 1. The scaled value of the variable is equal to (x - min) / (max - min).

Bucket and Quantile Transformations

You have the flexibility to divide the data values into n equally spaced classes. The quantile transformation is useful when you want to create groups with approximately the same frequency in each group. For example, you may want to divide the variable values into 10 uniform groups (10th, 20th, 30th,.... percentiles).

Buckets are created by dividing the data values into n equally spaced intervals based on the difference between the minimum and maximum values. Unlike a quantile transformation, the number of observations in each bucket is typically unequal.

Optimal Binning for Relationship to Target Transformation

The **Optimal Binning for Relationship to Target** transformation optimally splits a variable into n groups with regards to a target variable. This binning transformation is useful when there is a nonlinear relationship between the input variable and the target. A nominal measurement level is assigned to the transformed variable.

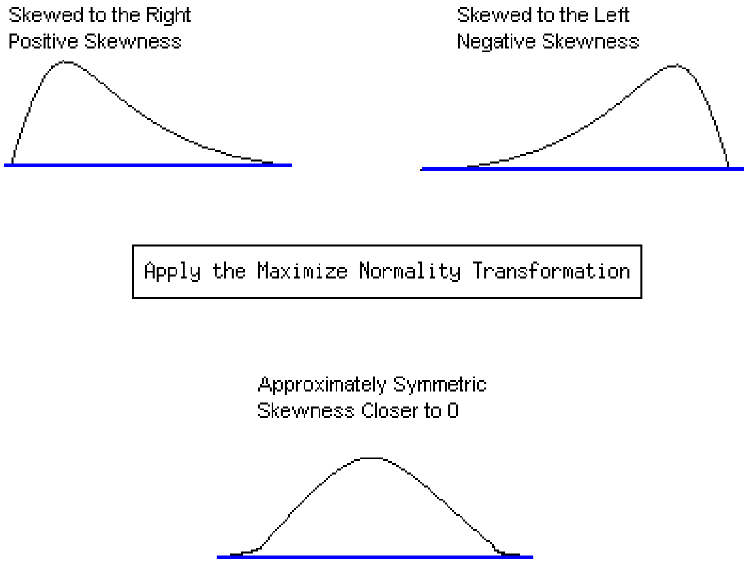
To create the n optimal groups, the node uses a decision tree of depth 1. The Missing Value property specifies how missing values are handled. Missing values are either used as a value assuring the split search or are assigned to a node based on the selected transformation. For example, if Largest Branch is selected, all of the observations that have missing values for the splitting rule are placed in the branch with the largest number of training observations.

Maximize Normality Power Transformation

The **Maximize Normality** power transformation is useful when you want to normalize a variable that has a skewed distribution or an overly peaked or flat distribution. Skewness measures the deviation of the distribution from symmetry. A skewed distribution has a heavy tail or extreme values located on one side of the distribution. If the skewness statistic for a variable is clearly different from 0, then the distribution is asymmetrical, while normal distributions are perfectly symmetrical (bell shaped). After you run the Transform Variables node, you can see the skewness and kurtosis values in the Results - Transform Variables window. The skewness values for each variable are listed in the Skewness column of the Transformations Statistics table. The degree of flatness is measured by the value in the Kurtosis column, where a normal distribution has a value of 1.

Note: Click Results from the Transform Variables pop-up menu to see the window.

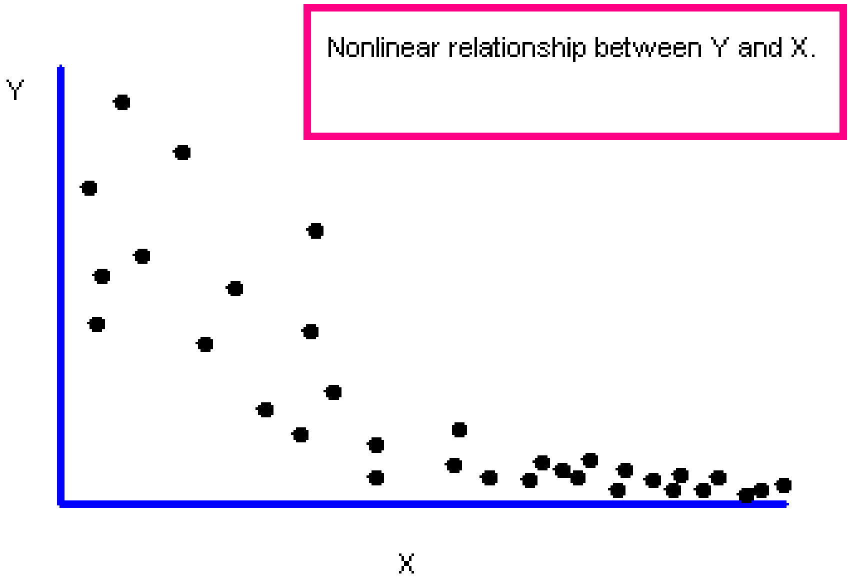
The degree of normality obtained by applying the Maximize Normality transformation to a skewed variable is naturally data dependent.



Maximize Correlation with Target Power Transformation

The **Maximize Correlation with Target** power transformation enables you to straighten (linearize) the relationship between an interval target and an interval input variable. If the Maximize correlation with target setting is used with a non-interval target, the data will not be transformed when the node runs. Although neural networks and decision tree methods are quite capable of predicting nonlinear relationships, increasing the correlation with the target tends to simplify the relationship between the target and the input variable, making it more easily understood and communicable to others. The overall model fit is also often improved by linearizing the data.

In the following scatter plot, notice that the relationship between Y and X is nonlinear. An additional problem is also evident in that the variance in Y also decreases as the value of X increases. The Maximize correlation with target transformation may help to straighten the relationship between Y and X and in turn stabilize the variance in Y across the different levels of X.

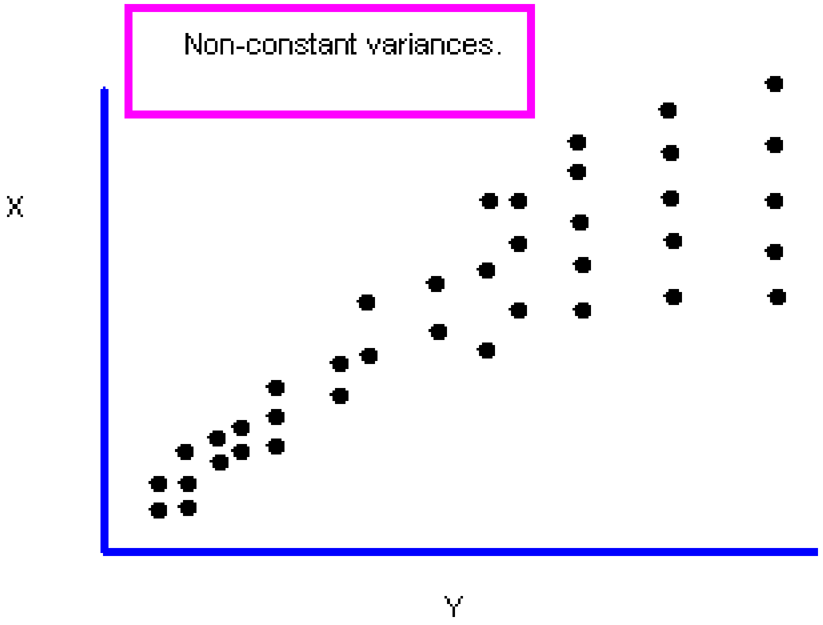


Equalize Spread with Target Levels Power Transformation

The **Equalize Spread with Target Levels** transformation helps to stabilize the variance in the interval input across the different levels of a binary, nominal, or ordinal target. Ordinary least squares estimation assumes that the errors are additive, and that they are normally independent random variables with a common variance. When the assumption of independence and common variance hold, least squares estimators have the desirable property of being the best (minimum variance) among all possible linear unbiased estimators.

The assumption of common variance implies that every observation on the target contains the same amount of information. Consequently, all observations in ordinary least squares receive the same weight. Unfortunately, equal weighting does not give the minimum variance estimates of the parameters if the variances are not equal. The direct impact of heterogeneous variances is a loss of precision in the parameter estimates compared to the precision that would have been realized if the heterogeneous variances had been stabilized using an appropriate transformation.

In the following scatter plot, although there is a linear relationship between Y and X, the variance increases as the values of the target Y increase. The variance is not equal across the different levels of the target Y. The Equalize Spread with Target Levels transformation may help to reduce the spread among the target levels and in turn provide more stable parameter estimates.



Group Rare Levels Transformations

The Group Rare Levels transformation is available for class variables only. This method combines the rare levels (if any) into a separate group, _OTHER_. You use the **Cutoff Value** property to specify the cutoff percentage. Rare levels are the variable values that occur less than the specified cutoff value.

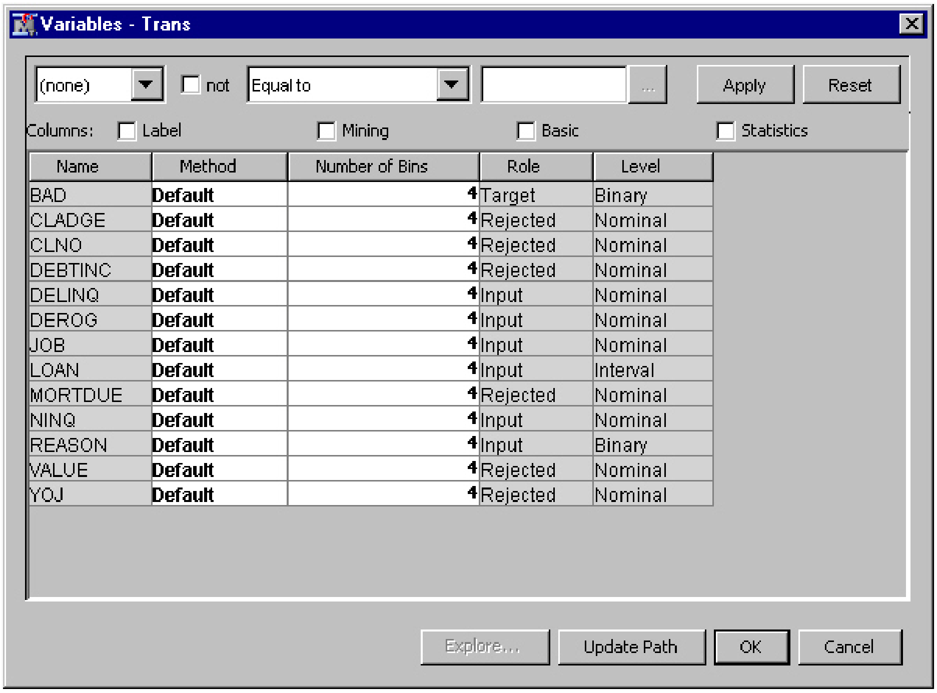
Dummy Indicators Transformation

The Dummy Indicators Transformation is available for class variables only. This method creates a dummy indicator variable (0 or 1) for each level of the class variable.

Setting the Transformation Method

Configuring the Variables Window

You configure the transformation method that you want to use for your data set using the table in the Variables window. You open the Variables window by selecting the ... button by the Variables property in the Enterprise Miner properties panel. The following is a display of the Variables window.



Columns: □ Label □ Mining □ Basic □ Statistics

Name	Method	Number of Bins	Role	Level
BAD	**Default**	4	Target	Binary
CLADGE	**Default**	4	Rejected	Nominal
CLNO	**Default**	4	Rejected	Nominal
DEBTINC	**Default**	4	Rejected	Nominal
DELINQ	**Default**	4	Input	Nominal
DEROG	**Default**	4	Input	Nominal
JOB	**Default**	4	Input	Nominal
LOAN	**Default**	4	Input	Interval
MORTDUE	**Default**	4	Rejected	Nominal
NINQ	**Default**	4	Input	Nominal
REASON	**Default**	4	Input	Binary
VALUE	**Default**	4	Rejected	Nominal
YOJ	**Default**	4	Rejected	Nominal

The table in the Variables window has the following columns. You can edit only the Method and Number of Bins columns.

- **Name** — name of the variable
- **Method** — transformation method assigned to variable
- **Number of Bins** — number of bins for binning interval variables when the use methods such as Bucket or Quantile. For more information, see “Setting the Number of Bins” on page 708 .
- **Role** — variable role
- **Level** — level for class variables

You can add additional columns by selecting one or more check boxes.

Select the **Label** checkbox to add the following columns:

- **Label** — adds a column for a label for each variable.

Select the **Mining** checkbox to add the following columns:

- **Order** — variable order
- **Report** — Boolean setting for creating reports.
- **Lower Limit** — minimum value for variable
- **Upper Limit** — maximum value for variable
- **Creator** — user who created the process flow diagram
- **Comment** — user-supplied comments about a variable
- **Format Type** — variable format

Select the **Basic** checkbox to add the following columns:

- **Type** — variable type
- **Format** — SAS Format for variable

- **Informat** — SAS Informat for variable
- **Length** — Length of the variable

Select the **Statistics** checkbox to add the following columns:

- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percent of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for numeric variables only. For character variables, . is displayed.
- **Maximum** — displays the maximum values for numeric variables only. For character variables, . is displayed.
- **Mean** — displays the arithmetic mean values for numeric variables only. For character variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for numeric variables only. For character variables, . is displayed.
- **Skewness** — displays the skewness for numeric variables only. For character variables, . is displayed.
- **Kurtosis** — displays the kurtosis values for numeric variables only. For character variables, . is displayed.

Setting the Number of Bins

If you select a binning transformation such as **bucket**, **quantile**, or **optimal**, then you can set the maximum number of bins in the Number of Bins column. To set the maximum number of bins, select the field in the Number of Bins column that corresponds to the variable, and type the number of bins that you want to use . You must hit the ENTER key after changing the number of bins in a cell for Enterprise Miner to accept the new value. The default value for the maximum number of bins is 4. With some data sets, Enterprise Miner may find less than the specified maximum number of bins.

Using the Formula Builder

Overview

The Formula Builder enables you to create customized transformations from the input variables. To open the Formulas window, select the ... button for the Formulas property in the properties panel.

The Formulas window contains the following tabs:

- “Inputs Tab” on page 709
- “Outputs Tab” on page 710
- “Sample Tab” on page 711
- “Log Tab” on page 711

Note: The term, a *new variable*, refers to a customized transformation of the original variables in the input data source.

You can customize the layout of these tabs. Select the right arrow on the tab to view the contents of a tab in its own pane to the right of other tabs. Select the down arrow on the tab to view the contents of a tab in its own pane below other tabs.

Inputs Tab

The top portion of the Formulas window displays the distribution of the variable that is selected in the table below.

The bottom portion of the window displays the following information about the original variables:

- **Name** — name of the variable
- **Method** — transformation method assigned to variable
- **Number of Bins** — number of bins for binning interval variables when the use methods such as Bucket or Quantile.
- **Role** — variable role
- **Level** — level for class variables

You can add additional columns by selecting one or more check boxes. Select the Label check box to add a text label to be used for the variable in graphs and output.

Select the **Mining** checkbox to add the following columns:

- **Order** — variable order
- **Report** — Boolean setting for creating reports.
- **Lower Limit** — minimum value for variable
- **Upper Limit** — maximum value for variable
- **Creator** — user who created the process flow diagram
- **Comment** — user-supplied comments about a variable
- **Format Type** — variable format

Select the **Basic** checkbox to add the following columns:

- **Type** — variable type
- **Format** — SAS Format for variable
- **Informat** — SAS Informat for variable
- **Length** — Length of the variable

Select the **Statistics** checkbox to add the following columns:

- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percent of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for numeric variables only. For character variables, . is displayed.
- **Maximum** — displays the maximum values for numeric variables only. For character variables, . is displayed.
- **Mean** — displays the arithmetic mean values for numeric variables only. For character variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for numeric variables only. For character variables, . is displayed.
- **Skewness** — displays the skewness for numeric variables only. For character variables, . is displayed.

- **Kurtosis** — displays the kurtosis values for numeric variables only. For character variables, . is displayed.

Outputs Tab

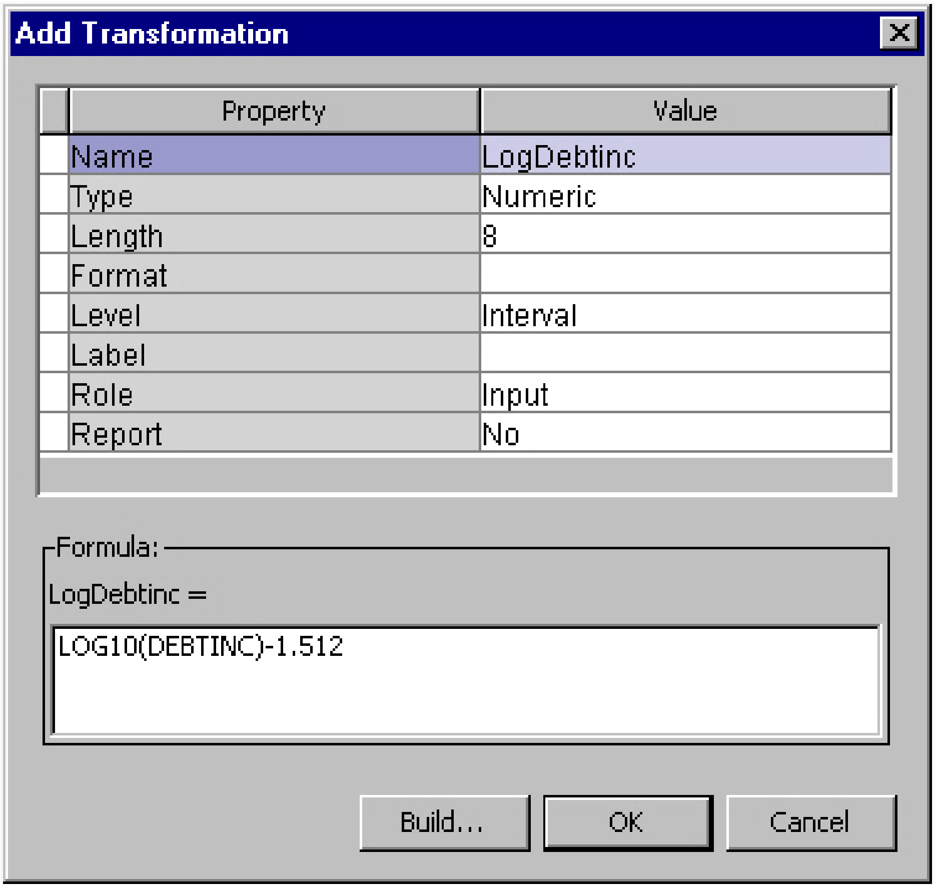
Here is an example display of the Outputs tab.

- To view the distribution of a new variable, select a variable in the table and click **Preview**.

  Note: A sample of the input data source is used to generate the distribution chart. Use the "Transform Variables Node Train Properties: Sample Properties" on page 699 to specify the settings that are used to generate the sample.

- To create a customized transformation, click the Create icon () at the top of the Formulas window and the Add Transformation window opens. Specify a variable name, type (Numeric or Character with drop-down menu), variable length, format, level, label, role, and report setting. Specify an expression for the new variable either by entering the formula in the Formula box or by "Using the Expression Builder" on page 711 . To open the Expression Builder, click Build.

  In the following example, the variable that is called LogDebtinc is created from the existing variable DEBTINC.



Add Transformation

Property	Value
Name	LogDebtinc
Type	Numeric
Length	8
Format	
Level	Interval
Label	
Role	Input
Report	No

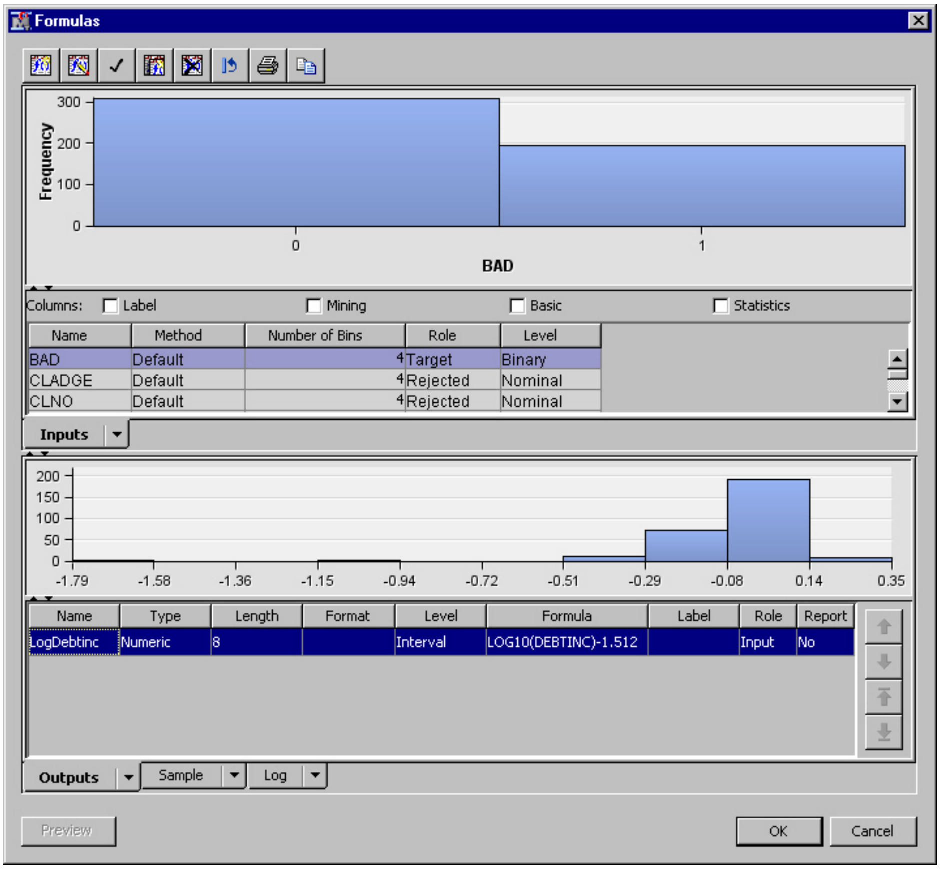
Formula:

LogDebtinc =

```
LOG10(DEBTINC)-1.512
```

- To modify the properties of a customized transformation, click the Edit properties icon () at the top of the Formulas window and the Edit Transformation window opens. You can change the variable name, type (Numeric or Character with drop-down menu), variable length, format, level, label, role, and report setting.

- To modify the definition of a customized transformation, click the Edit expression icon (✓) at the top of the Formulas window and the Expression Builder window opens. The expression can be modified as needed.
- To create a customized transformation from an existing customized transformation, select the existing variable in the table and click the Duplicate icon (▦). The variable name will be generated automatically.
- To delete a variable, select it in the table and click the Delete icon (☒).



Name	Method	Number of Bins	Role	Level
BAD	Default	4	Target	Binary
CLADGE	Default	4	Rejected	Nominal
CLNO	Default	4	Rejected	Nominal

Name	Type	Length	Format	Level	Formula	Label	Role	Report
LogDebtinc	Numeric	8		Interval	LOG10(DEBTINC)-1.512		Input	No

Sample Tab

The Sample tab displays a sample of the observations in the data set, including transformed variables.

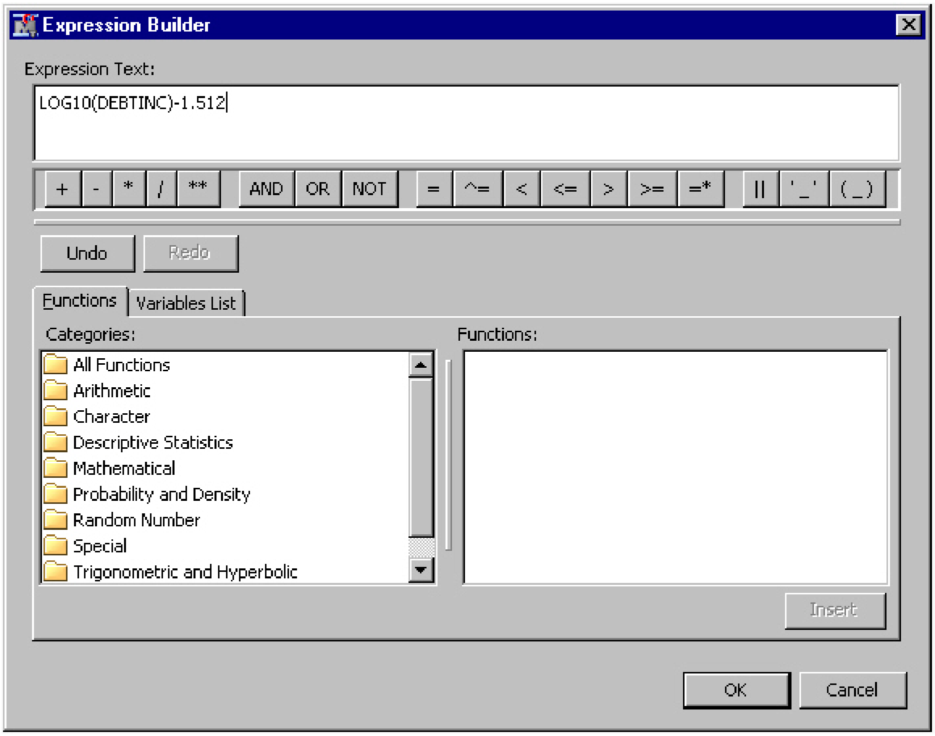
Log Tab

The Log tab displays the contents of the log that is generated when you create customized transformation.

Using the Expression Builder

Overview

You use the Expression Builder to define an expression.



Expression Text:

LOG10(DEBTINC)-1.512

The Expression Builder window contains the following elements:

- **Expression text** — specifies the combination of functions and mathematical operations that are used to derive a value. To enter an expression, you can type directly in the field or use the operator toolbar, the Functions tab, and the Variables List tabs to add operators, functions, and data to your expression. To clear the expression, use the BACKSPACE or DELETE keys.
- **Operator toolbar** — contains symbols that you can add to the expression including arithmetic operators, comparison operators, and logical operators.
- **Functions tab** — displays the list of available functions that you can use to create expressions. The functions are categorized by type. To add a function to the expression, double-click an item in the Functions tab, or select the item and click Insert.
- **Variables List tab** — displays a list of variables in the input data source that you can add to the expression. To add a variable to the expression, select the item and click Insert.
- **Insert** — inserts selected functions or data elements into the expression. This button is only enabled when you select a function or data source that can be inserted into the expression.

Functions Tab

Use the Functions tab to add functions to the expression. The Functions tab displays the list of available functions that you can use to create expressions. The functions are categorized by type. The list of available functions is determined by the underlying structure of your physical data.

- If you are building an expression that is based on relational data, then the functions that are supported by the SAS SQL procedure are displayed. For information about these functions, see "Functions and CALL Routines" in the SAS Language

Reference: Dictionary and "Summarizing Data" in the SAS SQL Procedure User's Guide.

- If you are building an expression that is based on OLAP data, then the MDX functions that are supported by SAS are displayed. For information about these functions, see "MDX Functions" in the SAS OLAP Server: MDX Guide.
- To specify a function that is specific to a different database system, manually enter the function name in the Expression Text field.

The Functions tab contains the following items:

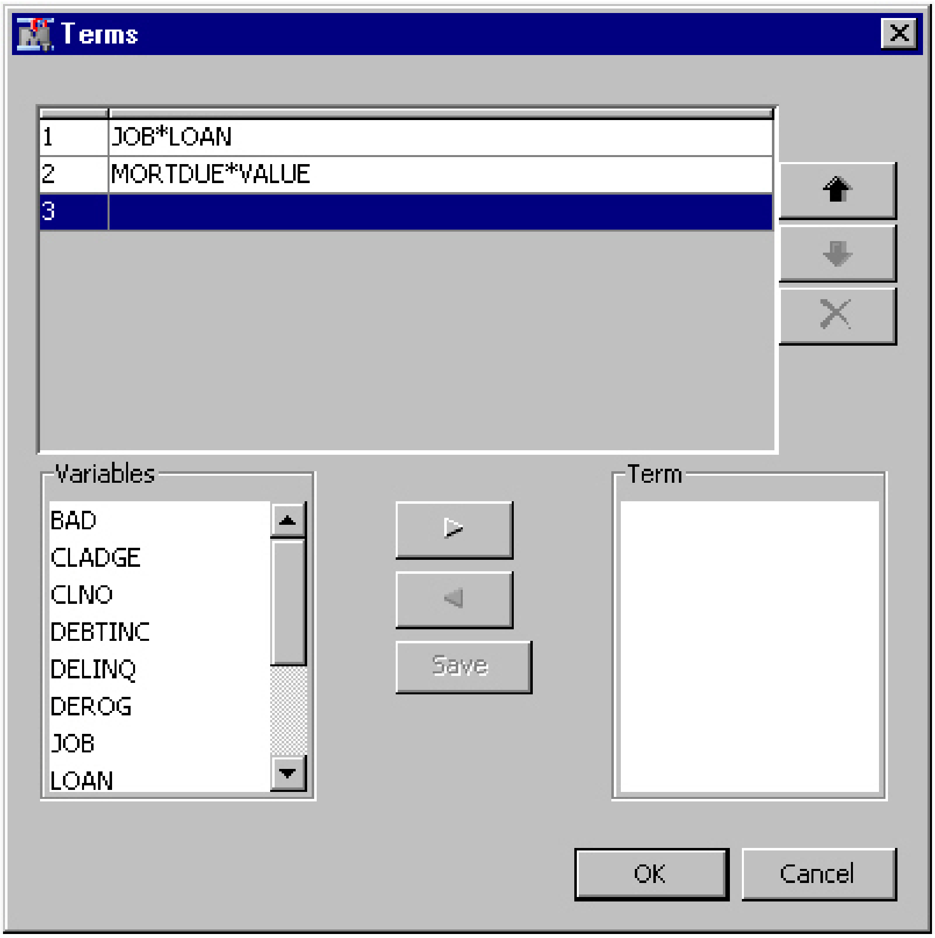
- **Categories** — displays the different types of functions that you can select to add to the expression. When you select a category, a list of functions for that category appears in the **Functions** list box.
- **Functions** — displays the specific functions that you can add to the expression. You can add a function to the expression by double-clicking the selected function or by selecting it and then clicking **Insert**.

Variables List Tab

Use the Variables List tab to add variables to the expression. To add a variable, select the variable and click **Insert**.

Creating Interaction Variables

One of the enhancements of the Transform Variables node since Enterprise Miner 4.3 is the ability to create interaction variables in the Terms window. To open the Terms window, select the ... button for the Interactions property in the properties panel.



1	JOB*LOAN
2	MORTDUE*VALUE
3	

BAD
CLADGE
CLNO
DEBTINC
DELINQ
DEROG
JOB
LOAN

An interaction term is a product of existing explanatory inputs. For example, the interaction of the variables JOB and LOAN is JOB*LOAN.

The **Variables** list contains the variables in the input data source. The interaction variable is the product of the variables in the **Term** list. Follow these steps to create an interaction variable.

1. Select the variables in the Variables list.
2. Use the right arrow to move the selected variables to the Term list.
3. Click Save and the interaction variable will be displayed at the top portion of the Terms window.

You use the up and down arrows to move the interaction variables. To delete an interaction variable, select an interaction variable and click ✕.

Note: When you have a Merge node before the Transform Variables node, you must run the Merge node first before you can invoke the Interactions editor.

Transform Variables Node Results

You can open the Results window of the Transform Variables node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results.

- **Properties**
  - **Settings** — displays a window with a read-only table of the Transform Variables node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Transform Variables node run. Information about whether the run completed successfully, the Run Start Time, Run Duration, and the Run ID are displayed in this window.
  - **Variables** — a table of the variables property of the node. (Not all variables appear in this table.) You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — Training is not available in the Transform Variables node.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Transform Variables node run.
  - **Output** — the SAS output of the Transform Variables node run. The SAS output displays a variable summary table, a transformations table by input variable, summary statistics for input interval and class variables, as well as summary statistics for output interval and class variables.
  - **Flow Code** — the SAS code used to produce the output that the Transform Variables node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications. If no transformed values were generated, the SAS Code menu item is dimmed and unavailable.
  - **PMML Code** — The Transform Variables code does not generate PMML code.
- **Transformations**
  - **Formula** — Opens a read-only table that displays user defined transformations and their basic statistics.
  - **Computed** — Opens a read-only table that displays computed transformations and their basic statistics.
  - **EM Meta** — Opens a read-only table that displays the transformations that were created using the Enterprise Miner Meta library.
  - **SAS Code** — Opens a read-only table that displays the transformations and their basic statistics that were created using SAS Code.
  - **Transformations Statistics** — displays the Transformations Statistics table.
- **Table** — open the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart Type window that you can use to create a custom plot of the data in the table that you have in focus.

References

Box, G.E.P. and Cox, D.R "An Analysis of Transformation." 1964. *Journal of the Royal Statistical Society* Ser. B, 26: 211–252.

Emerson, J.D. and Stoto, M.A "Exploratory Methods for Choosing Power Transformations." 1982. *Journal of American Statistical Association* LXXVII: 103–108.

Hoaglin, D.C., Mosteller, F., and Tukey, J.W, ed. 1983. *Understanding Robust and Exploratory Data Analysis*. New York, NY: John Wiley and Sons, Inc.

Masters, T 1993. *Practical Neural Network Recipes in C++*. San Diego, CA: Academic Press, Inc.

Rawlings, J.O 1988. *Applied Regression Analysis: A Research Tool*. Pacific Grove, CA: Wadsworth and Brooks/Cole.

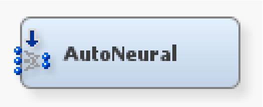
Part 13

Node Reference: Model Nodes

Chapter 48

AutoNeural Node

AutoNeural Node



Overview of the AutoNeural Node

The AutoNeural node belongs to the Model category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA). You can use the AutoNeural node as an automated tool to help you find optimal configurations for a neural network model.

The AutoNeural node conducts limited searches in order to find better network configurations. There are several options that the node uses to control the algorithm.

- Hidden nodes are added one at a time. A node may contain one or more neurons.
- For each training iteration, one estimate vector and one fit vector are retained according to the criteria that are displayed below.
- Subsequent training is initialized with the current estimates. The new node is initialized by PROC NEURAL with output weights = 0. Therefore, the beginning error is the same as the final error of the previous level.
- An adjustable setting for the maximum number of iterations is used. The training Maximum Iterations is adjusted higher if the selected iteration equals the Maximum Iterations. The Maximum Iterations is adjusted lower if the selected iteration is significantly lower than the Maximum Iterations setting.

- An adjustable setting for the amount of time after which training stops is used. The Total Time property enables you to set the maximum amount of time that can be used for training and preliminary training.
- The scale parameter for random numbers decreases at each training iteration when you use the cascade architecture.
- You can freeze or not freeze previously trained layers.
- Preliminary training is performed, depending on the value of the Tolerance property.
- The default combination functions and error functions are used.
- AutoNeural Model selection criteria:

Target	Validation Data	Selection
interval	No	_SBC_
interval	Yes	_VAVERR_
class	No	_MISC_
class	Yes	_VMISC_

See the "Predictive Modeling" on page 149 section for information that applies to all of the predictive modeling nodes.

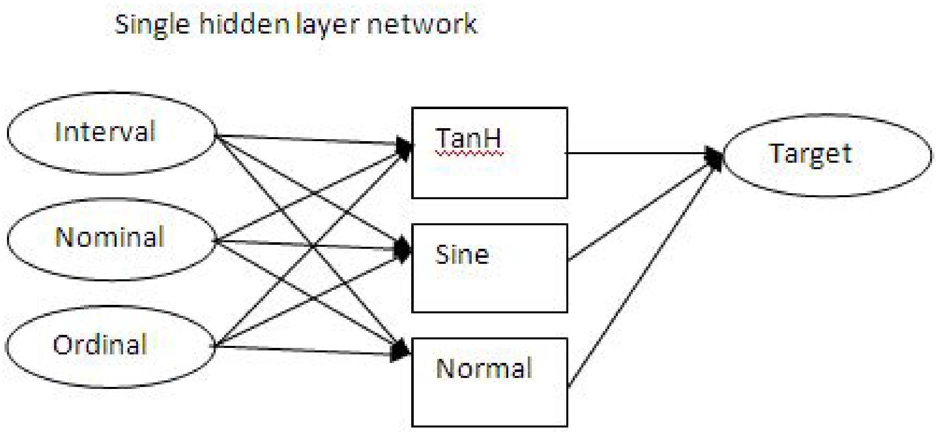
Network Architectures

The Autoneural node can create different types of feed forward network architectures. In the charts below, the solid or dashed lines represent weight vectors that are optimized by the model fitting function. Each activation function and target function contains a bias weight that is also optimized.

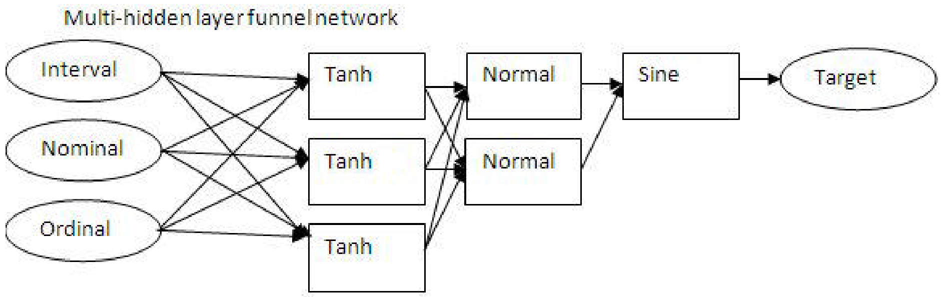
At each step, a new network model is optimized using an iterative nonlinear solution. Multiple candidate activation functions are optimized, then the best ones are retained in the model. The Autoneural node only performs the model search when the Train Action property is set to Search.

MLP (Multi-Layer Perceptron) networks are genearlly not interpretable due to the highly nonlinear nature of combinations of activation functions. While these algorithms have been tuned to avoid overfitting, you should use both validation and test data to make sure that these networks are generalizable for your model.

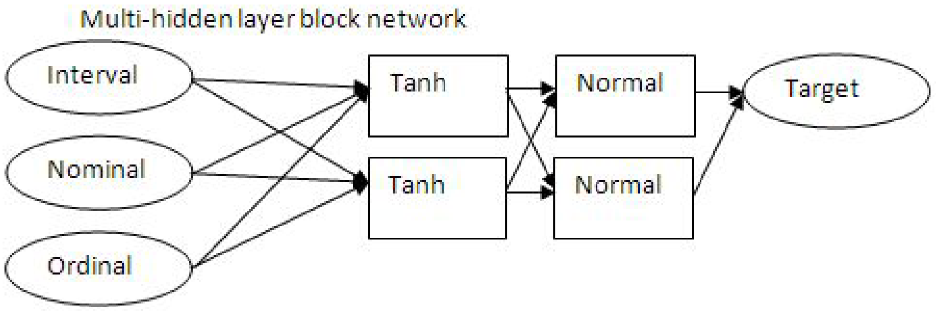
In a *single hidden layer network*, new hidden neuron units are added one at a time and are selected according to which activation function provides the most benefit. Many models may be successfully fit with a single hidden layer.



In a *funnel network*, new hidden neuron units are added to form a funnel pattern and are selected according to which activation function provides the most benefit.

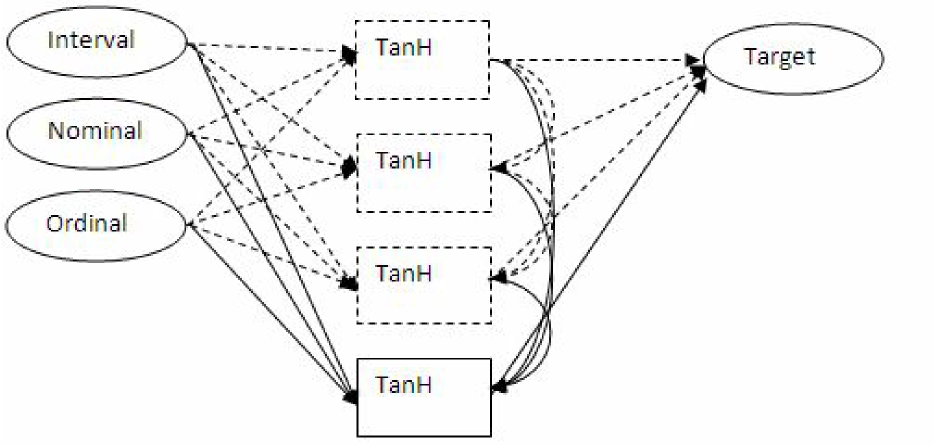


In a *block network*, new hidden neuron units are added as new layers in the network, according to which activation function provides the most benefit.



Finally, in a *cascade network*, new hidden neuron units are added and connected from all existing input and hidden neurons. All previously trained weights and biases are frozen. Each additional step optimizes the fit of the network to the residuals of the previous step.

Cascade network. Dashed connections and units are frozen at current values.



AutoNeural Node Data Set Requirements

The training data that you input into the AutoNeural node must contain at least one target variable and at least one input variable.

AutoNeural Node Properties

AutoNeural Node General Properties

The following general properties are associated with the AutoNeural node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first AutoNeural node added to a diagram will have a Node ID of AutoNeural. The second AutoNeural node added to a diagram will have a Node ID of AutoNeural2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — AutoNeural window. The Imported Data — AutoNeural window contains a lists of the ports that provide data sources to the AutoNeural node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — the Exported Data property provides access to the Exported Data — AutoNeural window. The Exported Data — AutoNeural window contains a list of the output data ports that the AutoNeural node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

AutoNeural Node Train Properties

The following train properties are associated with the AutoNeural node:

- **Variables** — Use the Variables window to view variable information, and specify the Use and Report status of a variable. Select the ... button to open a window containing the variables table. You can specify the Use and Report status of a variable, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. In default, the Variables window displays columns for a variable's name, use, report, role, and level.

  You can use the following buttons to view additional metadata, or limit metadata:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

AutoNeural Node Train Properties: Model Options

- **Architecture** — Use the Architecture property of the AutoNeural node to specify the neural network architecture that you want to use when you build the network.

  You can choose from the following networks:

  - **Single Layer** — (Default Setting) hidden layers are added in parallel. One or more activation functions can be used.
  - **Block Layers** — hidden layers are added as additional layers with a uniform number of neurons.
  - **Funnel Layers** — hidden nodes are added both to each existing hidden layer and to a new layer, so that the layers form a funnel pattern with a single node feeding into the output node.
  - **Cascade** — hidden nodes are added in a cascading fashion.

- **Termination** — Use the Termination property of the AutoNeural node to specify the method that you want to use to end training. Training always stops when the maximum run time is exceeded. When training is terminated, the model that has the best selection criteria is retained.

  You can choose from the following termination methods:

  - **Overfitting** — (Default Setting) training stops when overfitting is detected.
  - **Training Error** — training stops when the reduction in training data set error is less than 0.001.
  - **Time Limit** — training stops when the time specified in the Total Time property is exceeded.

- **Train Action** — Use the Train Action property of the AutoNeural node to specify the method that you want to use when training.

  You can choose from the following actions:

  - **Train** — all selected functions are trained until stopping.
  - **Increment** — nodes are added one at a time. No activation function is reused. If a layer lowers the average error, it is retained. If it does not, then it is dropped from the model.
  - **Search** — (Default Setting) nodes are added according to the architecture. The best network is retained as the final model.

- **Target Layer Error Function** — Each unit in a neural network produces a single computed value. Input and hidden units pass the computed values to other hidden or output units. The predicted value for output units is compared with the target value to compute the error function. Use the Target Layer Error Function property to specify the target layer error function for a user-defined network. Permissible values are

  - **Default** — For a categorical target variable, the default error function is multiple Bernoulli. For interval targets, the default error function is normal distribution.
  - **Normal** — may be used with any kind of target variable to predict the conditional mean. It is most often used with interval targets for which the noise distribution is approximately normal with constant variance. It can also be used with categorical targets for robust estimation of posterior probabilities.
  - **Cauchy** — The Cauchy distribution may be used with any kind of target variable. It is most often used when you want to predict the approximate conditional mode instead of the conditional mean.
  - **Logistic** — Logistic distribution may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain outliers.
  - **Huber** — The Huber M-estimator is suitable for unbounded interval targets that have outliers or that have a moderate degree of inequality of the conditional variance and a symmetric distribution. Huber can also be used for categorical targets when you want to predict the mode rather than the posterior probability.
  - **Biweight** — The Biweight M-estimator may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain severe outliers. Because of severe problems with local minima, you should obtain initial values by training with the Huber M-estimator before using the biweight M-estimator.
  - **Wave** — The Wave M-estimator may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain

  severe outliers. Because of severe problems with local minima, you should obtain initial values by training with the Huber M-estimator before using the Wave M-estimator.

  - **Gamma** — The Gamma distribution may be used only with strictly positive interval target variables. It is most often used when the standard deviation of the noise is proportional to the mean of the target variable.

  - **Poisson** — The Poisson distribution is suitable for skewed, nonnegative interval targets, especially counts of rare events, where the conditional variance is proportional to the conditional mean.

  - **Bernoulli** — Suitable for a target that takes only the values zero and one. Same as a binomial distribution with one trial.

  - **Entropy** — The Entropy error function is cross-entropy or relative entropy for independent interval targets with values between zero and one inclusive.

  - **MBernoulli** — The Multiple Bernoulli error function is suitable for categorical (nominal or ordinal) targets.

  - **Multinomial** — The Multinomial error function is may be used only with two or more interval target variables with non-negative values, where each case represents a number of trials that are equal to the sum of the target values. The activation function must force the outputs to sum to one, like Softmax.

  - **Mentropy** — The Multiple Entropy error function is cross-entropy or relative entropy for targets that sum to 1. It is the same criterion as Kullback-Leibler divergence. Multiple entropy should normally be used only with two or more interval target variables with values that lie between zero and one inclusive, and that sum to one for each case. However, multiple entropy may also be used with nominal or ordinal targets, primarily for software testing. The activation function must force the outputs to sum to one, like Softmax.

- **Maximum Iterations** — Use the Maximum Iterations property of the AutoNeural node to specify the maximum number of iterations permitted during training. Permissible values are any integer between 1 and 50. The default setting is 8.

- **Number of Hidden Units** — Use the Number of Hidden Units property of the AutoNeural node to specify the number of hidden units that you want to use. Permissible values are integers between 1 and 8. The default setting is 2.

- **Tolerance** — Use the Tolerance property of the AutoNeural node to configure the extent of the preliminary search. Valid values are

  - **Low** — no preliminary search is performed.

  - **Medium** — (Default Setting) preliminary statements are executed.

  - **High** — preliminary statements are executed, as well as applying the setting ABSCONV=0.001 to the training data.

- **Total Time** — specifies the amount of training time that PROC NEURAL is allowed to use and the amount of time that is allowed for preliminary training. This means that the AutoNeural node might require approximately twice the amount of time specified here in order to complete a successful run of the node. The allowable choices are

  - **Five Minutes**

  - **Ten Minutes**

  - **Thirty Minutes**

  - **One Hour**

- **Two Hours**
- **Four Hours**
- **Seven Hours**
- **One Day**
- **Two Days**
- **Four Days**
- **Seven Days**

The default Total Time setting is 1 hour.

AutoNeural Node Train Properties: Increment and Search

- **Adjust Iterations** — When set to No, the Adjust Iterations property of the AutoNeural node suppresses adjustments that are made to the Maximum Iterations property value setting when the Train Action is set to Search or Increment. If the Train Action is set to Search or Increment and Adjust Iterations is set to Yes, then the Maximum Iterations value is adjusted higher if the selected iteration equals the previously used Maximum Iterations. Similarly, the Maximum Iterations value can be adjusted lower if the selected iteration is significantly lower than the previously used Maximum Iterations. The default setting for the Adjust Iterations property is Yes.
- **Freeze Connections** — Use the Freeze Connections property of the AutoNeural node to specify whether connections should be frozen. The default setting for the Freeze Connections property is No.
- **Total Number of Hidden Units** — Use the Total Number of Hidden Units property of the AutoNeural node to specify the total hidden units ceiling when the Train Action is set to Search. When Search is performed, the total number of hidden units trained is calculated at each run. If the calculated number of total hidden units is greater than or equal to the value stored in Total Hidden, then training stops and a final model is selected. Permissible values are 5, 10, 20, 30, 40, 50, 75, or 100. The default setting is 30.
- **Final Training** — Use the Final Training property of the AutoNeural node to indicate whether the final model should be trained again to allow the model to converge. If the Final Training property is set to Yes, the number of iterations that are used will correspond to the value set in the Final Iterations property.
- **Final Iterations** — Use the Final Iterations property of the AutoNeural node to indicate the number of iterations to use when the Final Training property is set to Yes. The Final Iterations property is unavailable if the Final Training property is set to No. The Final Iterations property accepts integers greater than zero.

AutoNeural Node Train Properties: Activation Functions

Use Activation Functions property group to set the use status for each of the available activation functions. on page 728

- **Direct** — Use the Direct property of the AutoNeural node to indicate that you want to use the direct activation function during training. The default setting of the Direct property is **Yes**.
- **Exponential** — Use the Exponential property of the AutoNeural node to indicate that you want to use the exponential activation function during training. The default setting of the Exponential property is **No**.

- **Identity** — Use the Identity property of the AutoNeural node to indicate that you want to use the identity activation function during training. The default setting of the Identity property is **No**.
- **Logistic** — Use the Logistic property of the AutoNeural node to indicate that you want to use the logistic activation function during training. The default setting of the Logistic property is **No**.
- **Normal** — Use the Normal property of the AutoNeural node to indicate that you want to use the normal activation function during training. The default setting of the Normal property is **Yes**.
- **Reciprocal** — Use the Reciprocal property of the AutoNeural node to indicate that you want to use the reciprocal activation function during training. The default setting of the Reciprocal property is **No**.
- **Sine** — Use the Sine property of the AutoNeural node to indicate that you want to use the sine activation function during training. The default setting of the Sine property is **Yes**.
- **Softmax** — Use the Softmax property of the AutoNeural node to indicate that you want to use the softmax activation function during training. The default setting of the Softmax property is **No**.
- **Square** — Use the Square property of the AutoNeural node to indicate that you want to use the square activation function during training. The default setting of the Square property is **No**.
- **Tanh** — Use the Tanh property of the AutoNeural node to indicate that you want to use the hyperbolic tangent activation function during training. The default setting of the Tanh property is **Yes**.

AutoNeural Node Score Properties

The following score properties are associated with the AutoNeural node:

- **Hidden Units** — Use the Hidden Units property of the AutoNeural node to specify whether or not you want to create hidden unit variables. The default setting of the Hidden Units property is **No**.
- **Residuals** — Use the Residuals property of the AutoNeural node to specify whether or not you want to create residual variables. The default setting of the Residuals Property is **Yes**.
- **Standardization** — Use the Standardization property of the AutoNeural node to specify whether or not you want to create standardization variables. The default setting of the Standardization property is **No**.

AutoNeural Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

AutoNeural Node Activation Functions

At each hidden or output layer, the net input is transformed by an activation function. The following activation functions are available:

Function Name	Range	Expression in terms of g
Exponential	(0, ∞)	exp(g)
Identity	(-∞, ∞)	g
Logistic	(0, 1)	$\frac{1}{1+\exp(-g)}$
Reciprocal	(0, ∞)	1/g
Sine	[-1, 1]	sin(g)
Softmax	(0, 1)	$\frac{\exp(g)}{\sum(exponentials)}$
Square	[0, ∞)	g^2
Hyperbolic Tangent	(-1, 1)	$\tanh(g) = 1 - \frac{2}{1+\exp(2g)}$

AutoNeural Node Results Window

You can open the Results window of the AutoNeural node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- Properties
  - **Settings** — displays a window with a read-only table of the AutoNeural Network node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the AutoNeural Network node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.

  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Autoneural node run.
  - **Output** — the SAS output of the Autoneural node run.
  - **Flow Code** — the SAS data step code that the AutoNeural node generates to score incoming data sets and to pass on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Assessment**
  - **Fit Statistics** — The Fit Statistics table for the AutoNeural node includes the following columns:
    - **Target** — name of the target variable.
    - **Fit Statistics** — each row in the Fit Statistics column contains a different statistic for model goodness-of-fit. The variable name that is used in SAS code for each fit statistic appears in the Fit Statistics column.
    - **Statistics Label** — an easy-to-understand label name for each fit statistic variable.
    - **Train** — The fit statistic value for the target variable in the training data set.
    - **Validation** — The fit statistic value for the target variable using the validation data set.
    - **Test** — The fit statistic value for the target variable using the test data set.

    Some fit statistics are not calculated for both interval and non-interval targets. The table below provides a key to which fit statistics are calculated according to the type of target used. Refer to the "Fit Statistics Variable Table" on page 732 to determine what statistics are calculated for each variable.
  - **Classification Chart** — a bar chart that shows the correct classification of the values of the class target variable.
  - **Residual Statistics** — The Residual Statistics plot displays a box-and-whisker plot for the residual measurements when the target is interval.
  - **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

You can also plot the base and best values along with model values for class target variables.

The Score Rankings Overlay plot displays both train and validate statistics on the same axis.

By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any).

- posterior probability of target event
- number of events
- cumulative and noncumulative lift values
- cumulative and noncumulative % response
- cumulative and noncumulative % captured response
- gain
- actual profit or loss
- expected profit or loss

- **Score Rankings Matrix** — The score ranking matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. You must partition your data before you can create a Score Ranking Matrix chart. Plots can also be created for report variables.

  The parameters that the Score Rankings Matrix Chart can plot are:

  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response

- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

  For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

  The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target.

  You can select the score distribution statistic for the Score Distribution chart. For class targets, the default chart choice is Percentage of Events. For interval targets, the default chart choice is Mean for Predicted. Multiple chart choices are available for the Score Distribution Chart. The chart choices are

Categorical Targets:

- Percentage of Events
- Number of Events
- Cumulative Percentage of Events
- Expected Profits (if you have defined the decision information)

Interval Targets:

- Mean for Predicted
- Max. for Predicted
- Min. for Predicted

- **Model**
  - **Iteration Plot** — By default, the Iteration plot is a line plot of the average squared error. If the Train Action property is set to Train, then the X-axis is labeled Training Iteration. If you set the Train Action property to Increment or Search, then the X-axis is labeled Training Step. You can select other response variables from the Select Chart list. The other response variables are
    - Misclassification Rate
    - Number of Wrong Classifications
    - Error Function
    - Sum of Squared Errors
    - Average Error Function
    - Average Squared Error
    - Maximum Absolute Error
    - Mean Squared Error
    - Root Mean Squared Error
    - Final Prediction Error
    - Akaike's Information Criterion
    - Schwarz's Bayesian Criterion
  - **Weights — Final** — The Weights — Final plot is a two-dimensional histogram of all weights that were generated for the selected iteration of the selected run.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard that you can use to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Fit Statistics Variable Table

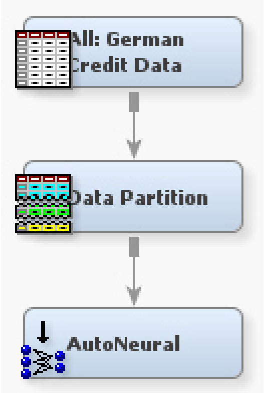
		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
AIC	Akaike's Information Criterion	Yes	Yes
APROF	Average Profit of the Target	Yes	No
ASE	Average Squared Error	Yes	Yes
AVERR	Average Error Function	Yes	Yes
DFE	Degrees of Freedom for Error	Yes	Yes
DFM	Model Degrees of Freedom	Yes	Yes
DFT	Total Degrees of Freedom	Yes	Yes
DIV	Divisor for _ASE_	Yes	Yes
ERR	Error Function	Yes	Yes
FPE	Final Prediction Error	Yes	Yes
MAX	Maximum Absolute Error	Yes	Yes
MISC	Misclassification Rate	Yes	No
MSE	Mean Squared Error	Yes	Yes
NOBS	Sum of Frequencies	Yes	Yes
_NW_	Number of Estimated Weights	Yes	Yes
PROF	Total Profit	Yes	Yes
RASE	Root Average Squared Error	Yes	Yes

		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
RFPE	Root Final Prediction Error	Yes	Yes
RMSE	Root Mean Squared Error	Yes	Yes
SBC	Schwarz's Bayesian Criterion	Yes	Yes
SSE	Sum of Squared Errors	Yes	Yes
SUMW	Sum of Case Weights Times Frequency	Yes	Yes
WRONG	Number of Wrong Classifications	Yes	No

Example: Using the Search Train Action

Overview

The following example demonstrates how to use the search train action, in which nodes are added to the network depending on the architecture that you specify.



Follow these steps to create the following process flow diagram:

1. “Define the Data Source” on page 734
2. “Add Nodes to the Diagram Workspace” on page 734
3. “Set the AutoNeural Node Properties” on page 734
4. “Run the AutoNeural Node” on page 734
5. “View the AutoNeural Node Output” on page 734
6. “View the AutoNeural Node Training Code” on page 737

Define the Data Source

1. From the main menu, select: **File** ⇨ **New** ⇨ **Data Source**.
2. Select SAS Table as the metadata source and click **Next**.
3. When prompted to Select a SAS Table, enter **`SAMPSIO.DMAGECR`** in the input field and click **Next**.
4. In the Table Properties window, click **Next**.
5. In the Metadata Advisor Options window, select the Advanced advisor. Click **Next**.
6. In the Column Metadata window, set the role of the variable good_bad to **Target**. Click **Next**.
7. If the Decision Processing window opens, select **No** and click **Next**.
8. In the Data Source Attributes window, leave the data source role as **Raw** and click **Next**.
9. In the Summary window, click **Finish**.

Add Nodes to the Diagram Workspace

1. Drag the newly created data source onto an Enterprise Miner diagram workspace.
2. Drag a Data Partition node onto the diagram from the node toolbar Sample folder, and then connect the All: German Credit node to the Data Partition node.
3. Drag an AutoNeural node onto the diagram from the node toolbar Model folder, and connect the Data Partition node to the AutoNeural node.

Set the AutoNeural Node Properties

1. Click the AutoNeural node in the diagram workspace to select it.
2. In the Properties panel for the AutoNeural node, set the Train Action property of the Model Options group to **Search**.
3. In the Activation Functions property group, set the Direct, Normal, Square, and Tanh properties to **Yes**. Set the remaining activation functions to **No**.
4. Note that the Termination property of the Model Options group is set to **Overfitting**. Because you partitioned the data and you are using a class target variable, the misclassification rate of the Validation data set is used.
5. Note that the Tolerance property of the Model Options group is set to **Medium**. This setting indicates that preliminary statements are executed when the node runs.

Run the AutoNeural Node

1. Right-click the AutoNeural node, and select **Run**.
2. When the AutoNeural node run is complete, select **Results**.

View the AutoNeural Node Output

1. In the Results window, expand the Output window.
2. The first section of the Output window displays a variable summary, a table of the model events, and a table of the predicted and decision variables. Scroll down to the first search that the node performed.

```
------ Search # 1 SINGLE LAYER trial # 1 : DIRECT :  Training -----


_ITER_     _AIC_      _AVERR_     _MISC_     _VAVERR_     _VMISC_
```

```
   0      598.691    0.61086    0.3000     0.60888    0.29766
   1      431.439    0.40180    0.1875     0.62017    0.29766
   2      421.327    0.38916    0.1900     0.71923    0.30100
   3      420.847    0.38856    0.1875     0.71225    0.29431
   4      420.691    0.38836    0.1900     0.74103    0.29766
   5      420.647    0.38831    0.1900     0.74369    0.29431
   6      420.623    0.38828    0.1900     0.76462    0.29766
   7      420.614    0.38827    0.1900     0.77283    0.29431
   8      420.610    0.38826    0.1900     0.78979    0.29431
   8      420.610    0.38826    0.1900     0.78979    0.29431

----- Selected Iteration based on _VMISC_ -----

_ITER_     _AIC_      _AVERR_     _MISC_     _VAVERR_    _VMISC_

   3      420.847    0.38856    0.1875     0.71225    0.29431
```

Eight iterations are performed in this search (the default setting for the Maximum Iterations property is eight). Because you partitioned the data and you are using a class target variable, the misclassification rate of the Validation data set is used to select and iteration. Based on these results, the network from iteration 3 is selected.

Note: Because this network only includes a direct connection, no preliminary training is performed.

3. Scroll down to Trial #2. This trial is performed using the Tanh activation function.

```
----- Search # 1 SINGLE LAYER trial # 2 : TANH :  Prelim -------

The NEURAL Procedure

Preliminary       Starting        Objective        Number
 Training          Random          Function          of        Terminating
    Run             Seed             Value       Iterations     Criteria

        1          196765887     0.417286920225           8
        2         1258498424     0.493684502807           8
        3         1121166870     0.402140358109           8
        4         1463483536     0.428628812381           8
        5         1266664492     0.424141526204           8
        6         1370487091     0.397375204114           8
        7         1472610205     0.436653553198           8
        8         1986582559     0.419403549987           8

------ Search # 1 SINGLE LAYER trial # 2 : TANH :  Training -----

_ITER_     _AIC_      _AVERR_     _MISC_     _VAVERR_    _VMISC_

   0      543.900    0.39738    0.1900     0.62328    0.23746
   1      531.166    0.38146    0.1800     0.57797    0.25084
   2      517.923    0.36490    0.1800     0.61547    0.26756
   3      513.275    0.35909    0.1650     0.63134    0.28094
   4      502.382    0.34548    0.1825     0.64671    0.27425
   5      495.090    0.33636    0.1800     0.67126    0.27425
   6      486.778    0.32597    0.1750     0.70288    0.26421
   7      481.578    0.31947    0.1675     0.71332    0.26087
   8      474.139    0.31017    0.1675     0.71659    0.27425
   8      474.139    0.31017    0.1675     0.71659    0.27425

----- Selected Iteration based on _VMISC_ -----

_ITER_      _AIC_      _AVERR_     _MISC_     _VAVERR_     _VMISC_

   0      543.900    0.39738     0.19       0.62328    0.23746
```

Preliminary training is performed in this run. The final weights of the best network that was trained in the preliminary training are used to initialize the subsequent training. Iteration 0 is chosen as the best network in this search.

4. The AutoNeural node then runs searches using the normal and square activation functions. Scroll down in the Output window. Note that the AutoNeural node runs a second search, using all of the activation functions that you specified. Each of these searches is run with preliminary training.

5. Scroll down to view the final training history.

```
----- Final Training History -----

    _step_     _func_ _status_  _iter_ _AVERR_ _MISC_  _AIC_  _VAVERR_ _VMISC_

SINGLE LAYER 1 DIRECT initial      0   0.61086 0.3000 598.691  0.60888 0.29766
SINGLE LAYER 1 DIRECT candidate    3   0.38856 0.1875 420.847  0.71225 0.29431
SINGLE LAYER 1 TANH   candidate    0   0.39738 0.1900 543.900  0.62328 0.23746
SINGLE LAYER 1 NORMAL reject       5   0.35864 0.2000 512.914  0.68756 0.27759
SINGLE LAYER 1 SQUARE reject       0   0.36231 0.1775 515.850  0.62200 0.27759
SINGLE LAYER 1 TANH   keep         0   0.39738 0.1900 543.900  0.62328 0.23746
SINGLE LAYER 2 DIRECT reject       2   0.27803 0.1300 556.428  0.75596 0.30100
SINGLE LAYER 2 TANH   reject       0   0.29815 0.1550 688.520  0.77238 0.27759
SINGLE LAYER 2 NORMAL reject       0   0.30277 0.1650 692.219  0.73099 0.26087
SINGLE LAYER 2 SQUARE reject       8   0.33162 0.1775 715.298  0.69221 0.27090
```

In this table, you can see the results of all of the searches that the node ran. The first rows indicate the results of the first search. Two of the networks, based on the Direct and Tanh activation functions, were identified as candidate networks. The remaining networks, based on the Normal and Square activation functions, were rejected. All of the networks in the second search were rejected.

Because it has the lowest value of _VMISC_, iteration 0 of the Tanh network from the first search was chosen.

6. Scroll down to view the final model.

```
----- Final Model -----
----- Stopping: Termination criteria was satisfied: overfitting based on _VMISC_

_func_     _AVERR_     _VAVERR_     neurons

 TANH     0.39738     0.62328        2
                                  =======
                                     2
```

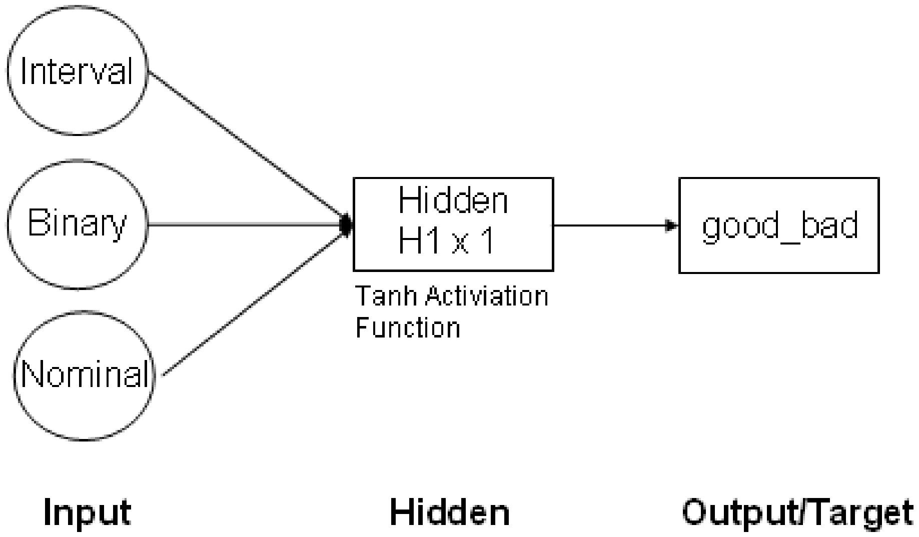
The network that was chosen uses a Tanh activation function, and has two hidden units. You specify the number of hidden units in the Total Number of Hidden Units property.

View the AutoNeural Node Training Code

You can view the final network by examining the SAS training code. From the Results window main menu, select **View** ⇨ **Properties** ⇨ **Train Code**. Scroll down in the Train Code window to view the final network.

```
*------------------------------------------------------------*;
* AutoNeural Final Network;
*------------------------------------------------------------*;
*;
proc neural data=EMWS.Part_TRAIN dmdbcat=WORK.AutoNeural_DMDB
validdata=EMWS.Part_VALIDATE
;
input %INTINPUTS / level=interval id=interval;
input %BININPUTS / level=nominal id=binary;
input %NOMINPUTS / level=nominal id=nominal;
target good_bad / level=NOMINAL id=good_bad;
*------------------------------------------------------------*;
* Layer # 1;
*------------------------------------------------------------*;
Hidden 2 / id = H1x1_ act=TANH;
connect interval H1x1_;
connect binary H1x1_;
connect nominal H1x1_;
connect H1x1_ good_bad;
```

You can visualize this network as follows. Note that this diagram does not separate the output and target layers.



Chapter 49
Decision Tree Node

Decision Tree Node

Overview of the Decision Tree Node

The **Decision Tree** node is located in the Model folder of the SAS Enterprise Miner toolbar.

An empirical tree represents a segmentation of the data that is created by applying a series of simple rules. Each rule assigns an observation to a segment based on the value of one input. One rule is applied after another, resulting in a hierarchy of segments within segments. The hierarchy is called a tree, and each segment is called a node. The original segment contains the entire data set and is called the root node of the tree. A node with all its successors forms a branch of the node that created it. The final nodes are called leaves. For each leaf, a decision is made and applied to all observations in the leaf. The type of decision depends on the context. In predictive modeling, the decision is the predicted value.

You use the **Decision Tree** node to create decision trees that do one of the following tasks:

- classify observations based on the values of nominal, binary, or ordinal targets

- predict outcomes for interval targets
- predict the appropriate decision when you specify decision alternatives

An advantage of the **Decision Tree** node over other modeling nodes, such as the **Neural Network** node, is that it produces output that describes the scoring model with interpretable English rules. The **Decision Tree** node also produces detailed score code output that completely describes the scoring algorithm in detail. For example, the English rules for a model might describe the rules, "If monthly mortgage-to-income ratio is less than 28% and months posted late is less than 1 and salary is greater than $30,000, then issue a gold card."

Another advantage of the **Decision Tree** node is the treatment of missing data. The search for a splitting rule uses the missing values of an input. Surrogate rules are available as backup when missing data prohibits the application of a splitting rule.

Note: English rules are useful for understanding the structure of a decision tree. But using the English rules set as the sole basis for a scoring algorithm is not recommended. The English rules output does not completely describe how the **Decision Tree** node handles missing and unknown data values in your scoring model. If you want to export the logic in your scoring model for use in an external application, you should use the output score code from the Decision Tree Results.

Decision trees produce a set of rules that can be used to generate predictions for a new data set. This information can then be used to drive business decisions. For example, in database marketing, decision trees can be used to develop customer profiles that help marketers target promotional mailings in order to generate a higher response rate.

The SAS implementation of decision trees finds multi-way splits based on nominal, ordinal, and interval inputs. You choose the splitting criteria and other options that determine the method of tree construction. The options include the popular features of CHAID (Chi-square automatic interaction detection), and those described in *Classification and Regression Trees*. (See L. Breiman, J.H. Friedman, R. A. Olsen, and C. J. Stone, 1984.).

Prior probabilities and frequencies enable you to use the training data in different proportions than in the populations on which predictions are made. For example, if fraud occurs in 1% of transactions, then one-tenth of the non-fraud data is often adequate to develop the model using prior probabilities that adjust the 10-to-1 ratio in the training data to the 100-to-1 ratio in the general population. You specify the prior vector as part of the target profile for the categorical target.

The criterion for evaluating a splitting rule can be based either on a statistical significance test, namely an F test or a Chi-square test, or on the reduction in variance, entropy, or Gini impurity measure. The F test and Chi-square test accept a p-value input as a stopping rule. All criteria allow the creation of a sequence of subtrees. You can use validation data to select the best subtree.

See the “Predictive Modeling” on page 149 section for information that applies to all of the predictive modeling nodes.

Decision Tree Node Variable Requirements

Overview

The **Decision Tree** node requires at least one target (response) variable and at least one input (independent explanatory) variable. You can also specify cost and frequency variables. Multiple target variables, multiple input variables, multiple cost variables, and a single frequency variable are allowed for the **Decision Tree** node. The target, input, cost, and frequency variables are mutually exclusive variables.

The node supports all target measurement levels (nominal, binary, ordinal, or interval). Cost and frequency variables must be interval variables.

A frequency variable represents the frequency of occurrence for other values in each observation. If the value of a frequency variable is missing or less than 0, then the observation is not used in the analysis. Frequency variable values are not truncated. You assign the frequency model role to the appropriate variable when you create a data source.

Multiple Targets

Decision trees are commonly used for interactive segmentation, usually market segmentation, sometimes followed by predictive modeling in the segments. With multiple targets, the user can select a different target in a leaf that has become sufficiently homogeneous with respect to the original target. The user ends up with one set of rules to display or score new data with later, instead of having to piece together rules from separate trees.

In contrast to segmentation, statisticians use trees as a predictive model in itself. SAS decision trees with multiple targets are not suitable for this because the split search only looks at one of the targets.

In order to use multiple targets, you set the Use Multiple Targets property to Yes. When multiple target variables are used, you use the Variables property of the **Decision Tree** node to specify which target variable is to be used as the decision variable for the root node of the tree. You do this by setting the Use property to Yes. All other target variables must set their Use property to No. You can switch targets by clicking on a leaf and then clicking the Switch Targets icon that appears on the toolbar of the Interactive Decision Tree window. When multiple targets are used, Assessment Plots and Assessment Tables are not available.

Decision Tree Node Properties

Decision Tree Node General Properties

The following general properties are associated with the **Decision Tree** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Decision Tree** node added to a diagram has a Node ID of Tree. The second **Decision Tree** node added to a diagram has a Node ID of Tree2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — Decision Tree window. The Imported Data — Decision Tree window contains a list of the ports that provide data sources to the **Decision Tree** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — the Exported Data property provides access to the Exported Data — Decision Tree window. The Exported Data — Decision Tree window contains a

list of the output data ports that the **Decision Tree** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

If data exists for an imported data source, you can select the row in the imported data table and click as follows:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Decision Tree Node Train Properties

The following train properties are associated with the **Decision Tree** node:

- **Variables** — Use the Variables property to specify the properties of each variable that you want to use in the data source. Select the ... button to the right of the Variables property to open a variables table. You can specify whether to use a variable or generate a report. For input variables, the Use status determines whether a variable is included in the model. For target variables, the Use status determines which target is the primary target. You can have only one target variable with a Use status of Yes. When you train a model using the Interactive Decision Tree on page 813 application, you can switch targets and use a target variable that has a Use status of No. The Explore functionality to view the distribution of a variable is available here.
- **Interactive** — Use the Interactive property of the **Decision Tree** node to launch an interactive training session in the Interactive Decision Tree. Click the ... button at the right of the Interactive Training property to launch the SAS Enterprise Miner Interactive Decision Tree. For information about interactive training, see Interactive Decision Tree on page 813 .
- **Use Frozen Tree** — specifies whether a frozen tree definition should be used or if a new tree should be created during training. The **Decision Tree** node must have already been run before you can run the node with this property set to Yes.
- **Use Multiple Targets** — specifies whether multiple targets should be available during training of the **Decision Tree** node.

Decision Tree Node Train Properties: Splitting Rule

- **Interval Target Criterion** — specify the method that you want to use to evaluate candidate splitting rules for interval variables and to search for the best one.

  Choose from the following splitting criteria:

  - **ProbF** — p-value of the F test that is associated with the node variance.
  - **Variance** — reduction in the square error from the node means.

- **Nominal Target Criterion** — specify the method that you want to use to evaluate candidate splitting rules for nominal variables and to search for the best one.

Choose from the following splitting criteria:

- **ProbChisq** — p-value of Pearson Chi-square statistic for target versus the branch node.
- **Entropy** — reduction in the entropy measure.
- **Gini** — reduction in the Gini index.

- **Ordinal Target Criterion** — specify the method that you want to use to evaluate candidate splitting rules for ordinal variables and to search for the best one.

  Choose from the following splitting criteria:

  - **Entropy** — reduction in the entropy measure, adjusted with ordinal distances.
  - **Gini** — reduction in the Gini index, adjusted with ordinal distances.

  For more detailed information, see the section on "Method of Split Search" on page 767 .

- **Significance Level** — Specifies the maximum acceptable p-value for the worth of a candidate splitting rule when a **ProbChisq** or **ProbF** criterion method is selected. The measure of worth depends on the value of the Criterion property. For criterion methods that are based on p-values, the threshold is a maximum acceptable p-value. For other criteria, the threshold is the minimum acceptable increase in the measure of worth. Permissible values are real numbers greater than 0 and less than or equal to 1.

- **Missing Values** — Use the Missing Values property of the **Decision Tree** node to specify how splitting rules handle observations that contain missing values for a variable. The default value is Use in search.

  Select from the following available missing value policies:

  - **Use in search** — uses missing values in the calculation of the worth of a splitting rule. This consequently produces a splitting rule that assigns the missing values to the branch that maximizes the worth of the split. This is a desirable option when the existence of a missing value is predictive of a target value.
  - **Largest branch** — assigns the observations that contain missing values to the largest branch.
  - **Most correlated branch** — assigns the observation to the branch with the smallest residual sum of squares among observations that contain missing values.

- **Use Input Once** — The Use Input Once property of the **Decision Tree** node specifies whether a splitting variable can be used only once, or whether a splitting variable can be repeatedly used in splitting rules that apply to descendant nodes. The default value for the Use Input Once property is **No**.

- **Maximum Branch** — Use the Maximum Branch property of the **Decision Tree** node to specify the maximum number of branches that you want a splitting rule to produce. Permissible values for the Maximum Branch property are integers between 2 and 100. The minimum value of 2 results in binary splits. The default value for the Maximum Branch property is 2.

- **Maximum Depth** — Use the Maximum Depth property of the **Decision Tree** node to specify the maximum number of generations of nodes that you want to allow in your decision tree. The original node is the root node. Children of the root node are the first generation. Permissible values are integers between 1 and 50. The default number of generations for the Maximum Depth property is 6.

- **Minimum Categorical Size** — Use the Minimum Categorical Size property of the **Decision Tree** node to specify the minimum number of training observations that a categorical value must have before the category can be used in a split search.

Permissible values are integers greater than or equal to 1. The default value for the Minimum Categorical Size property is 5.

Decision Tree Node Train Properties: Node

- **Leaf Size** — Use the Leaf Size property of the **Decision Tree** node to specify the minimum number of training observations that are allowed in a leaf node. Permissible values are integers greater than or equal to 1. The default setting is 5.
- **Number of Rules** — Use the Number of Rules property of the **Decision Tree** node to specify the number of splitting rules that you want to save with each node. The tree uses only one rule. The remaining rules are saved for comparison. Permissible values are integers greater than or equal to 1. The default value for the Number of Rules property is 5.
- **Number of Surrogate Rules** — Use the Number of Surrogate Rules property of the **Decision Tree** node to specify the maximum number of surrogate rules that the **Decision Tree** node seeks in each non-leaf node. The first surrogate rule is used when the main splitting rule relies on an input whose value is missing. Permissible values are nonnegative integers. The default value for the Number of Surrogate Rules property is 0.
- **Split Size** — Use the Split Size property of the **Decision Tree** node to specify the smallest number of training observations that a node must have before it is eligible to be split. Permissible values are integers greater than or equal to 2. The Split Size property uses a default value of (2*Leaf Size), unless you specify an integer value that is greater than the calculated default value.

Decision Tree Node Train Properties: Split Search

- **Use Decisions** — Select **Yes** to use the decision information during the split search.
- **Use Priors** — Select **Yes** to use the prior probabilities during the split search.
- **Exhaustive** — Use the Exhaustive property of the **Decision Tree** node to specify the highest number of candidate splits that you want to find in an exhaustive search. The Exhaustive property applies to multi-way splits and to binary splits on nominal targets with more than two values. Permissible values are integers between 0 and 2,000,000,000. The default setting for the Exhaustive property is 5000.
- **Node Sample** — Use the Node Sample property of the **Decision Tree** node to specify the maximum within-node sample size n that you want to use to find splits. If the number of training observations in a node is larger than n, then the split search for that node is based on a random sample of size n. Permissible values are integers greater than or equal to 2. The default value for the Node Sample property is 20000.

Decision Tree Node Train Properties: Subtree

- **Method** — Use the Method property to specify the method that you want to use to select a subtree from the fully grown tree for each possible number of leaves.

  The following subtree methods are available:

  - **Assessment** (default) — The smallest subtree with the best assessment value. The assessment value depends on the setting that you choose for the Assessment Measure property. Validation data set is used if available.
  - **Largest** — The largest (full) tree is selected.
  - **N** — The largest subtree with at most N leaves is selected. Use the Number of Leaves property to specify the value of N, the number of leaves.

- **Number of Leaves** — When the Method property of the **Decision Tree** node is set to N, use the Number of Leaves property to specify the largest number of leaves that you want in a subtree of n leaves. Permitted values are integers greater than or equal to 1. The default value for the Number of Leaves property is 1.

- **Assessment Measure** — Use the Assessment Measure property of the **Decision Tree** node to specify the method that you want to use to select the best tree, based on the validation data when the Method property is set to Assessment. If no validation data is available, training data is used.

  The available assessment measurements are as follows:

  - **Decision** (default setting) — The Decision method selects the tree that has the largest average profit and smallest average loss if a profit or loss matrix is defined. If no profit or loss matrix is defined, the value of the model assessment measure is reset in the training process, depending on the measurement level of the target. If the target is interval, the measure is set to Average Square Error. If the target is categorical, the measure is set to Misclassification.

  - **Average Square Error** — The Average Square Error method selects the tree that has the smallest average square error.

  - **Misclassification** — The Misclassification method selects the tree that has the smallest misclassification rate.

  - **Lift** — The Lift method evaluates the tree based on the prediction of the top n% of the ranked observations. Observations are ranked based on their posterior probabilities or predicted target values. For an interval target, it is the average predicted target value of the top n% observations. For a categorical target, it is the proportion of events in the top n% of the data. When you set the Measure property to Lift, you must use the Assessment Fraction property to specify the proportion for the top n% of cases.

- **Assessment Fraction** — When the Assessment Measure property of the **Decision Tree** node is set to Lift, use the Assessment Fraction property to specify the proportion n (of the top n% of observations to use) during model assessment. Permissible values for the Percentage property are real numbers between 0 and 1. The default value for the Percentage property is 0.25. When a decision matrix has been defined and the Measure property is set to Lift, the percentage indicates the average profit or loss among the top n% of observations.

Decision Tree Node Train Properties: Cross Validation

- **Perform Cross Validation** — Use the Perform Cross Validation property to specify whether to perform cross validation for each subtree in the sequence.

- **Number of Subsets** — Use the Number of Subsets property to specify the number of cross validation subsets or folds. The maximum allowed value is 20.

- **Number of Repeats** — Use the Number of Repeats property to specify the number of times to repeat cross validation. The estimates from repeated cross validation are the averages of the estimates from the individual cross validation runs. The maximum allowed value is 100.

  Note that the Classification Matrix generated in the Interactive Decision Tree Application on page 813 is affected by this property. If you perform cross validation and set the **Number of Repeats** property to a value greater than 1, then the Classification Matrix displays the **Average Validation Frequency**. If the **Number of Repeats** property is equal to 1, then the Classification matrix displays the **Validation Frequency**.

- **Seed** — Use the Seed property to specify the random number seed for generating the validation subsets.

Note: No cross validation is performed when the product of the following three numbers is greater than 2,000,000,000:

  - the number of nodes in the original decision tree
  - the value of the **Number of Subsets** property
  - the value of the **Number of Repeats** property

Decision Tree Node Train Properties: Observation-Based Importance

- **Observation Based Importance** — Use the Observation Based Importance property to specify whether observation-based importance statistics should be generated. Variable importance is calculated using the SAS decision forest tree methodology. See the section in this document on Variable Importance on page 749 , as well as the Gradient Boosting node on page 801 documentation for more details about observation-based Importance.
- **Number Single Var Importance** — Use the Number Single Var Import property to specify the number of variables for which one way importance statistics should be generated. The Number Single Var Import property is valid only when the Observation Based Importance property is enabled.

Decision Tree Node Train Properties: P-Value Adjustment

- **Bonferroni Adjustment** — When set to **No**, the Bonferroni Adjustment property of the **Decision Tree** node suppresses Bonferroni adjustments to the p-values. The default setting is **Yes**.
- **Time of Bonferroni Adjustment** — Use the Time of Bonferroni Adjustment property of the **Decision Tree** node to indicate whether the Bonferroni adjustment should take place **Before** or **After** the split is chosen. The default setting is **Before**. The Time of Bonferroni Adjustment property is ignored if the Bonferroni Adjustment property is set to **No**.
- **Inputs** — When set to **Yes**, the Inputs property of the **Decision Tree** node adjusts the p-values for the number of inputs. The default setting for the Inputs property is **No**. When Inputs is set to **Yes**, you must use the Number of Inputs property to specify the number of inputs that you want to consider uncorrelated.
- **Number of Inputs** — When the Inputs property of the **Decision Tree** node is set to **Yes**, use the Number of Inputs property to specify the number of inputs that you want to consider uncorrelated. The Number of Inputs property is ignored if the Inputs property is set to **No**. Permissible values are integers greater than or equal to 1. The default value for Number of Inputs is 1. If the specified value is greater than the number of input variables, then the **Decision Tree** node uses the number of input variable.
- **Depth Adjustment** — When set to **Yes**, the Depth Adjustment property adjusts the p-values for the number of ancestor splits. The default setting for the Depth Adjustment property is **Yes**.

Decision Tree Node Train Properties: Output Variables

- **Leaf Variable** — Set the Leaf Variable property of the **Decision Tree** node to **No** to indicate that you want to suppress the default creation of _NODE_ variables in the output data. _NODE_ variables store numeric identification numbers for each leaf that has observations assigned to it. The default setting for the **Leaf Identifier** node is **Yes**.

Decision Tree Node Train Properties: Interactive Sample

- **Create Sample** — specifies whether the **Decision Tree** node uses the entire data set or a sample of the data set in the Interactive Decision Tree application. Specify **Default** to use the default sampling settings, **None** to use the entire data set, and **User** to specify your own sampling settings.
- **Sample Method** — specifies the method that is used to sample the data set. You can choose from **Random**, **First N**, or **Stratify**. If you select **Stratify**, then a stratified sample is created based on the target variable. This property is available only when the **Create Sample** property is set to **User**.
- **Sample Size** — specifies the size of the sample that is create. Valid values are positive integers. This property is available only when the **Create Sample** property is set to **User**.
- **Sample Seed** — specifies the seed for the random number generator when creating a sample. This property is available only when the **Create Sample** property is set to **User**.
- **Performance** — Use the Performance property to specify where to store the working copy of the training data.
  - **Disk** — The Disk option stores the working copy in a disk utility file. Storing the copy on disk can free a considerable amount of memory for calculations, possibly shortening the execution time. The default setting of the Performance property is Disk.
  - **RAM** — The RAM option stores the working copy in memory, if enough memory is available for both the working copy and a minimum number of calculations.

Decision Tree Node Score Properties

The following score properties are associated with the **Decision Tree** node:

- **Variable Selection** — Use the Variable Selection property to specify whether variable selection should be performed based on importance values. If the Variable Selection property is set to **Yes**, all variables that have an importance value greater than or equal to 0.05 have the variable role set to Input. All other variables are set to Rejected. The default setting for the Variable Selection property is **Yes**.
- **Leaf Role** — When the Leaf Identifier property of the **Decision Tree** node is set to **Yes**, use the Leaf Role property to specify the variable role that you want to apply to the created _NODE_ variables. The selection choices are **Segment**, **Input**, and **Rejected**. The default setting is **Segment**.

Decision Tree Node Report Properties

The following report properties are associated with the **Decision Tree** node:

- **Precision** — specifies the number of decimals displayed in the Variable Importance, Subtree Assessment, and Observation Based Importance tables and plots.
- **Tree Precision** — specifies the number of decimals displayed in the splitting values and average values displayed in nodes.
- **Class Target Node Color** — specifies how the nodes for a class target variable are colored. All coloring is performed based on the training data only.

  Select from the following options to determine node coloring:

  - **Percent Correctly Classified** — Node colors are based on the percentage of observations that are correctly classified as the decision value for that node.
  - **Percent of Event** — Node colors are based on the predicted variable that corresponds to the event level of the target variable.
  - **Profit/Loss** — Node colors are based on the total profit or loss gained in each node. If no profit/loss information is provided, then the **Percent Correctly Classified** method is used.
  - **Single Color** — Enables you to specify a single color for all nodes.

  The values specified here are used as the default color values Interactive Decision Tree application. The Interactive Decision Tree application enables you to modify the color scheme during interactive mode. However, those changes are valid only during the current interactive session.

- **Interval Target Node Color** — specifies how nodes for a class target variable are colored. All coloring is performed based on the training data only.

  Select from the following options to determine node coloring:

  - **Average** — Node colors are based on the average target value inside each node.
  - **Root Average Square Error** — Node colors are based on the square root of the average square error for each node.
  - **Single Color** — Enables you to specify a single color for all nodes.

  The values specified here are used as the default color values Interactive Decision Tree application. The Interactive Decision Tree application enables you to modify the color scheme during interactive mode. However, those changes are valid only during the current interactive session.

- **Node Text** — Click the ... button next to the Node Text property to open the Node Text window. The Node Text property enables you to determine what text is displayed in each node of your decision tree.

  The following properties are available:

  - **Node ID** — specifies whether the node ID is displayed.
  - **Show Validation** — specifies whether the selected statistics calculated for the validation data are displayed. This property is ignored if the validation data does not exist.
  - **Count** — specifies whether to display the node count.
  - **Class Targets** — information-specific class target variables.
    - **Predicted Value** — specifies whether the predicted value is displayed.
    - **Target Values** — specifies what target values are displayed. Select **All Values** to display the classification percentage for all target values, **Event Only** to display the classification percentage for the target event of that node, and **None** to display no classification percentage.
    - **Percent Correct** — specifies whether the percentage of correct observations is displayed.
    - **Profit or Loss Values** — specifies what profit or loss information is displayed. Select **All Values** to display the profit or loss information for all target values, **Event Only** to display the profit or loss information for the target event of that node, and **None** to display no profit or loss information.

- **Interval Targets** — information specific to interval target variables.
  - **Average** — specifies if the node's average target value is displayed.
  - **Root Average Squared Error** — specifies if the node's root average squared error is displayed.

Decision Tree Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Variable Importance

Some tree node output variables are selected based on their relative importance, which is reported in the Variable Importance Table. This is an overview of the components of variable importance.

The relative importance of an input variable υ in subtree T is computed as follows:

$$I(v;T) \propto \sqrt{\sum_{\tau \in T} a(s_v,\tau)\Delta SSE(\tau)}$$

Here, the sum is over nodes τ in T, and s_υ denotes the primary or surrogate splitting rule using υ. $a(s_\upsilon,\tau)$ is the measure of agreement for the rule using υ in node τ:

$$a(s_v,\tau) = \begin{cases} 1 & \text{if } s_v \text{ is the primary splitting rule} \\ agreement & \text{if } s_v \text{ is a surrogate rule} \\ 0 & \text{otherwise} \end{cases}$$

ΔSSE(τ) is the reduction in sum of square errors from the predicted values:

$$\Delta SSE(\tau) = SSE(\tau) - \sum_{b \in B(\tau)} SSE(\tau_b)$$

$$SSE(\tau) = \begin{cases} \sum_{i=1}^{N(\tau)} (Y_i - \hat{Y}(\tau))^2 & \text{for interval target } Y \\ \sum_{i=1}^{N(\tau)} \sum_{j=1}^{J} (\delta_{ij} - \hat{p}_j(\tau))^2 & \text{for target with } J \text{ categories} \end{cases}$$

These conditions apply:

$$
\begin{aligned}
B(\tau) &= \text{set of branches from } \tau \\
\tau_b &= \text{child node of } \tau \text{ in branch } b \\
N(\tau) &= \text{number of observations in } \tau \\
\hat{Y}(\tau) &= \text{average } Y \text{ in training data in } \tau \\
\delta_{ij} &= 1 \text{ if } Y_i = j, 0 \text{ otherwise} \\
\hat{p}_j(\tau) &= \text{average } \delta_{ij} \text{ in training data in } \tau
\end{aligned}
$$

For a categorical target, the formula for SSE(τ) reduces to

$$
SSE(\tau) = \begin{cases} N(1 - \sum_{j=1}^{J} \hat{p}_j^2) & \text{for training data} \\ N(1 - \sum_{j=1}^{J} (2p_j - \hat{p}_j)\hat{p}_j) & \text{for validation data} \end{cases}
$$

Here, p_j is the proportion of the validation data with target value j, and N, pj, and pj-hat are evaluated in node τ.

In the **Decision Tree** node, the sum of square errors for a categorical target variable is always computed from the training data set. Therefore, the sum of square errors is equal to the Gini index.

Decision Tree Node Results

Results Window Menu

You can open the Results window of the **Decision Tree** node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **Decision Tree** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **Decision Tree** node run. The Run Start Time, Run Duration, Run ID, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays the information entered by the user for the node.
- **SAS Results**
  - **Log** — the SAS log of the Decision Tree run.
  - **Output** — the SAS output of the Decision Tree run.
  - **Flow Code** — the SAS code used to produce the output that the **Decision Tree** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.

  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The **PMML Code** menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Assessment**
  - "Fit Statistics Table" on page 752 — a table of the fit statistics from the model.
  - "Classification Chart" on page 752 — a bar chart that shows the correct classification of the values of the target variable.
  - "Score Rankings Overlay Chart" on page 753 — opens the Score Rankings Overlay chart.
  - "Score Rankings Matrix" on page 754 — opens the Score Rankings Matrix chart.
  - "Score Distribution Plot" on page 755 — opens the Score Distribution chart.
  - **Adjusted Charts** — if priors are used, then adjusted charts are available.
- **Model**
  - "Leaf Statistics Plot" on page 756 — opens the leaf statistics plot for the **Decision Tree** node.
  - "English Rules Window" on page 757 — displays the node definition (the English rules). English Rules are unavailable in the presence of multiple targets.
  - "Tree Plot" on page 758 — opens the Tree plot.
  - "Treemap Plot" on page 759 — opens the Treemap plot.
  - "Iteration Plot" on page 759 — the graph of statistic versus the number of leaves.

    You can choose from the following statistics:

    - Average Squared Error
    - Misclassification Rate
    - Sum of Squared Errors
    - Maximum Absolute Error
    - Subtree Assessment

    Iteration Plots are unavailable in the presence of multiple targets.

    Note: If decisions are defined, then additional statistics are available for viewing (for example, Profit).
  - **Variable Importance** — A table that contains the split-based measure of relative importance of each input variable in the selected subtree.
  - **Observation Based Importance Statistics** — a table that contains Input, Sum of Frequencies, Sum of Case Weights Times Freq, Number of Leaves, Misclassification Rate, Maximum Absolute Error, Sum of Squared Errors, Average Squared Error, Root Average Squared Error, Divisor for ASE, Total Degrees of Freedom.

    Note: This menu item is available when the Observation Importance property is set to Yes.
  - **Target Statistics by Node** — a table containing the Node number, Statistic Name, Statistic Value, Target, and Target Category or Decision. This report is available when there are multiple targets.

Note: This menu item is available when multiple variables have been assigned a role of Target and the Use Multiple Targets property is set to Yes.

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Select a Chart Type plotting wizard to modify an existing Results plot or create a Results plot of your own. The Select a Chart Type plotting wizard menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Results Tables and Plots

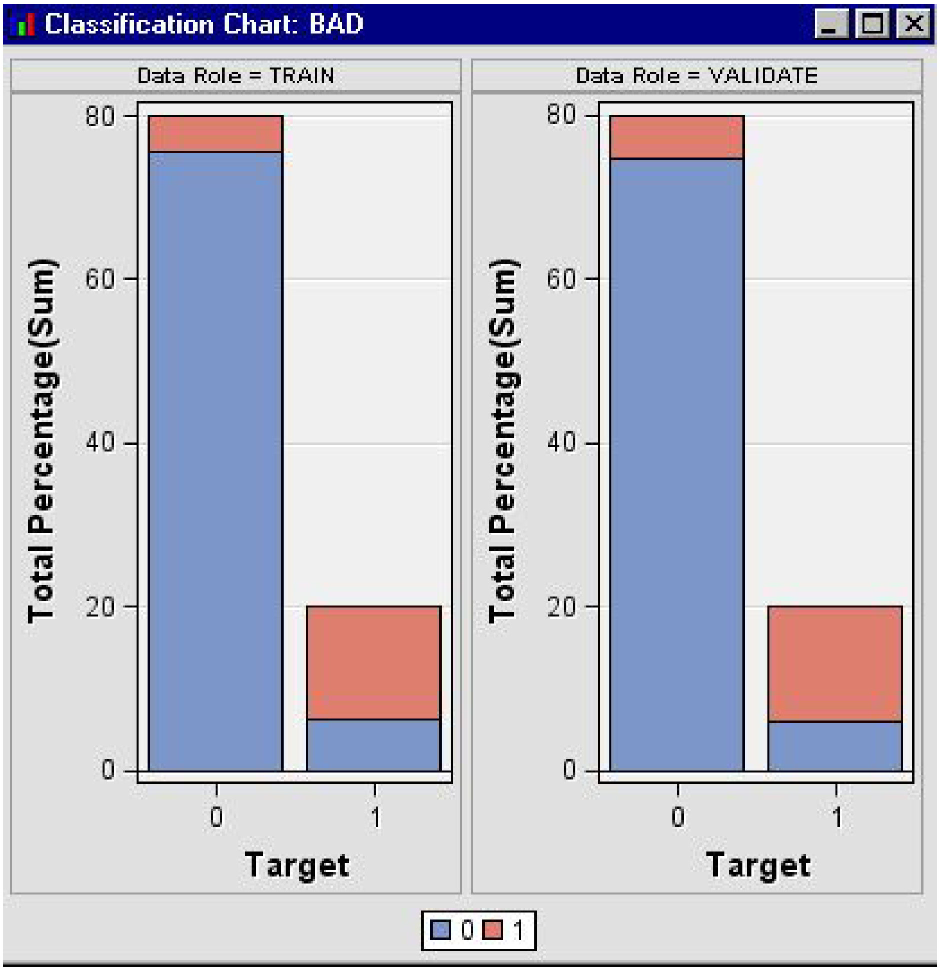
Fit Statistics Table

The Fit Statistics table displays the following statistics for the training, validation, and test data sets (if available):

- _NOBS_ — Sum of Frequencies
- _SUMW_ — Sum of Case Weights Times Freq
- _MISC_ — Misclassification Rate
- _MAX_ — Maximum Absolute Error
- _SSE_ — Sum of Square Errors
- _ASE_ — Average Sum of Squares
- _RASE_ — Root Average Sum of Squares
- _DIV_ — Divisor for ASE
- _DFT_ — Total Degrees of Freedom
- _APROF_ — Average Profit for Target
- _PROF_ — Total Profit for Target
- _PASE_ — Average Squared Error with Priors
- _PMISC_ — Misclassification Rate with Priors

Classification Chart

The Classification chart displays a stacked bar chart of the classification results for a categorical target variable. The following display shows an example of the Classification Table chart:



The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represents the percentage of total observations.

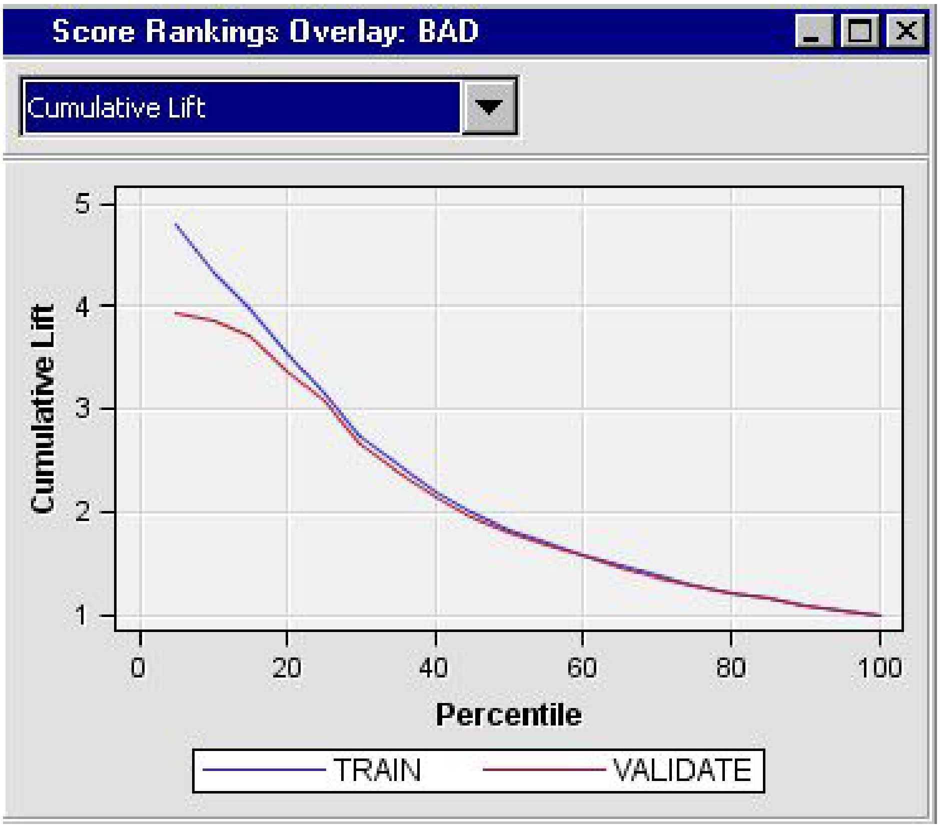
Score Rankings Overlay Chart

The Score Ranking Overlay chart enables you to overlay training and validation plots for the following statistics on the vertical axis. These statistics are computed based on the model that you create, the random baseline model, and the exact model.

- Cumulative Lift
- Lift
- Gain
- % Response
- Cumulative % Response
- % Captured Response
- Cumulative % Captured Response
- Total Profit
- Total Expected Profit
- Cumulative Expected Profit
- Cumulative Total Expected Profit

- Mean Expected Profit

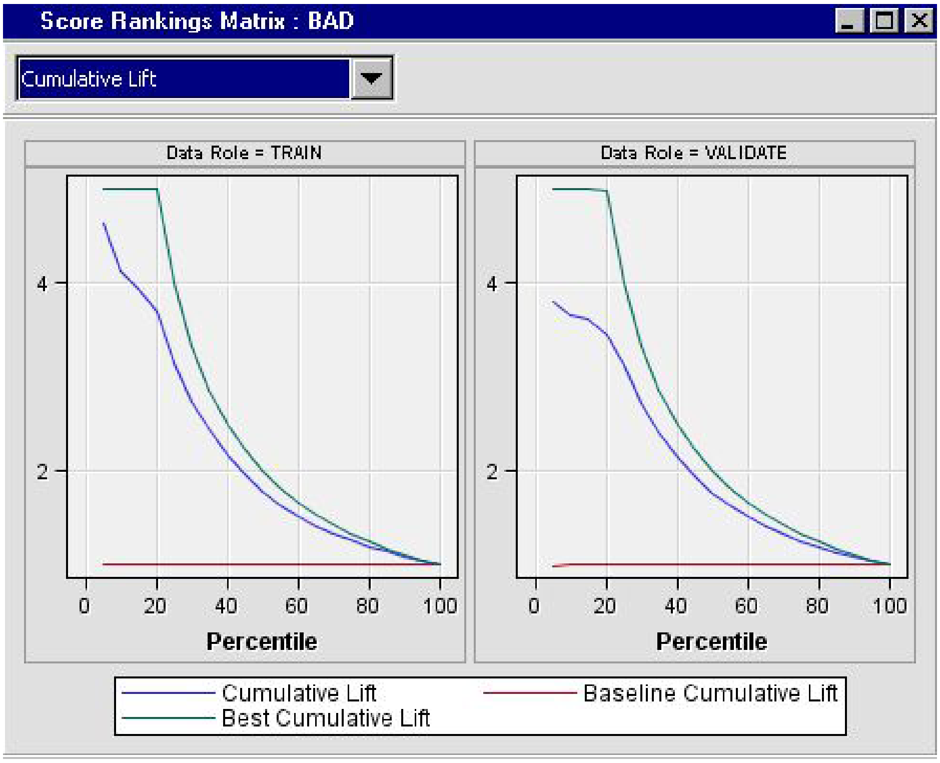
The horizontal axis of a Score Rankings chart displays the groups (percentile) of observations.



When there is no validation data set, the score rankings overlay simply overlays cumulative, baseline, and best plots of the training data for each statistic.

Score Rankings Matrix

The score rankings matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. The example below plots model cumulative lift, base model cumulative lift, and the best cumulative lift across percentiles. Plots can also be created for report variables. In order for the score matrix plots to be generated, a **Data Partition** node must be included in your process flow diagram to create a validation sample.

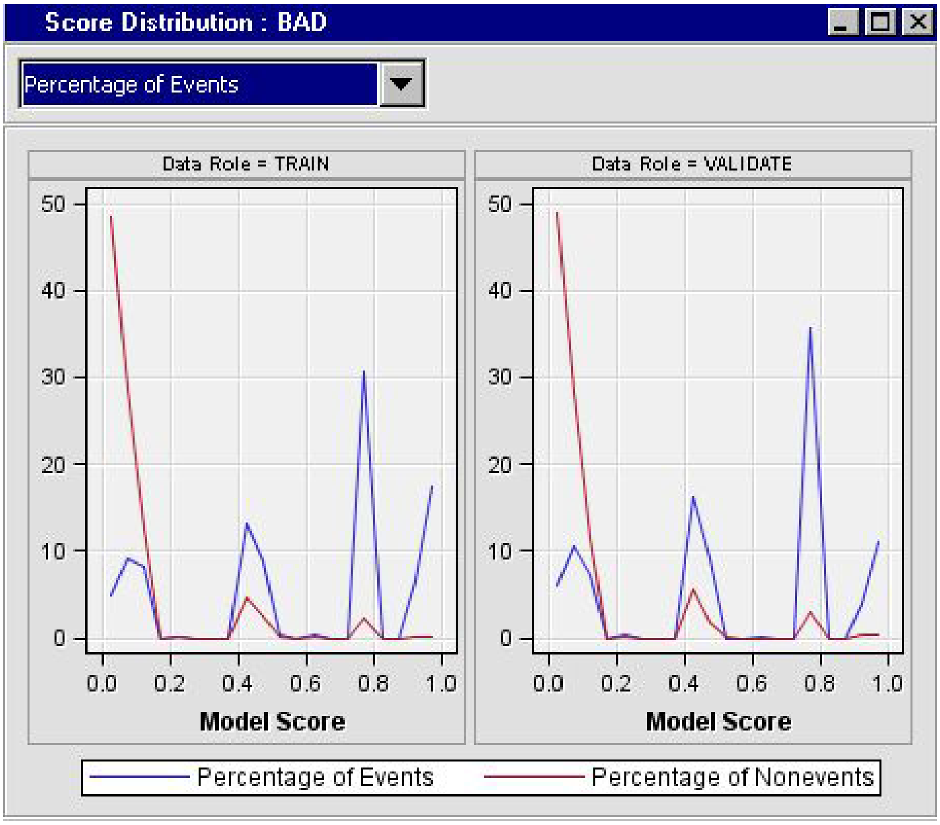


Score Distribution Plot

The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

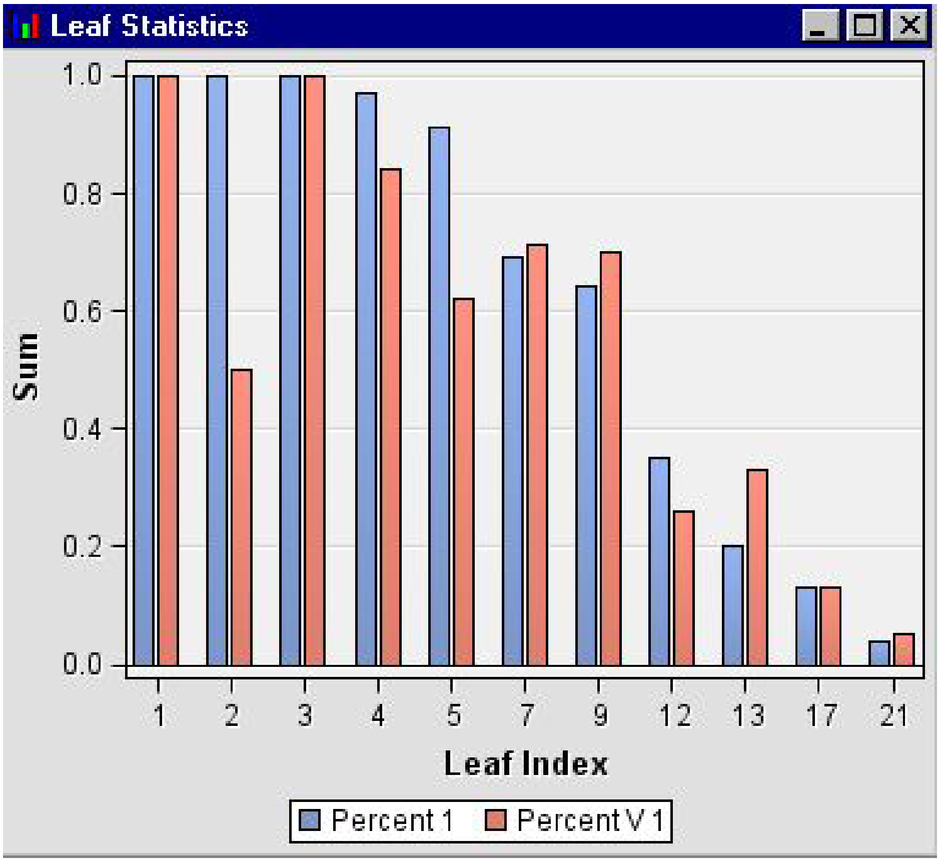
For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. For interval targets, observations are grouped into bins, based on the actual predicted values of the target.

The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores.



Leaf Statistics Plot

The Leaf Statistics plot displays a bar chart of summary statistics for the leaves of the currently selected subtree.

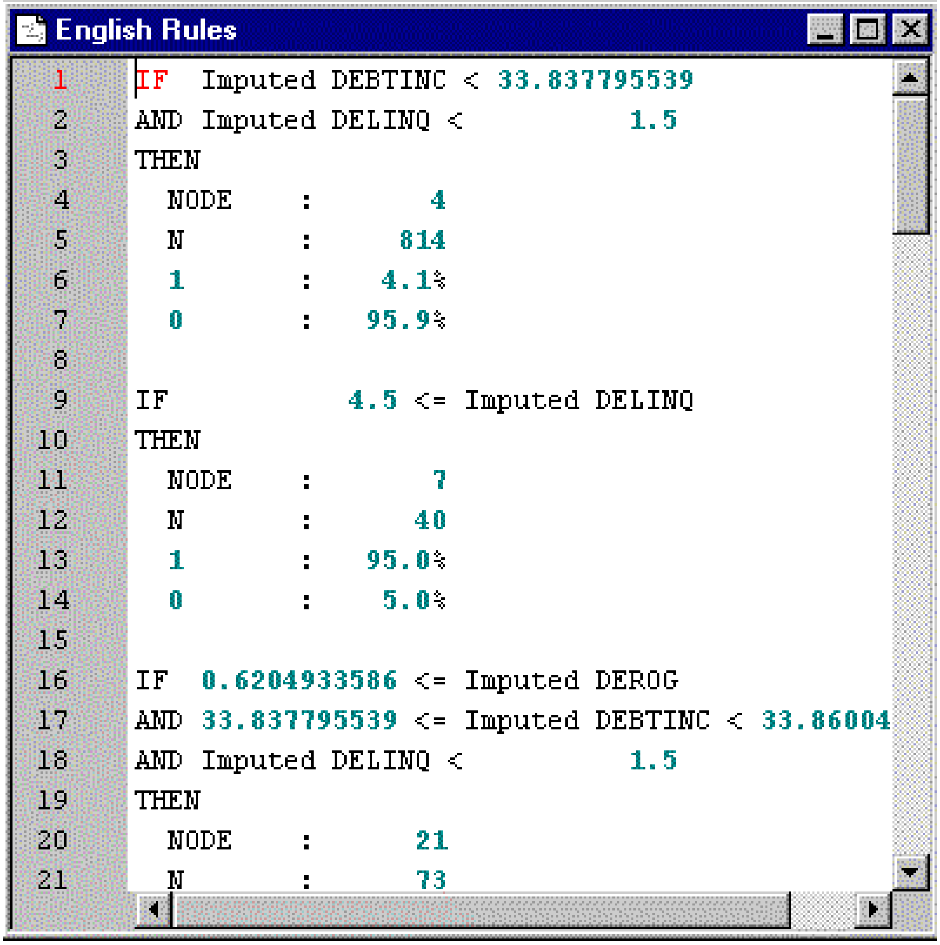


If you select the Leaf Statistics plot in the Decision Tree Results window, you can view the Leaf Statistics table by selecting **View** ⇨ **Table** from the Results main menus.

The information that is provided in this plot depends on whether the target variable is categorical or interval.

English Rules Window

The English Rules window displays the interpretable node definition for the leaf nodes in a tree model.



```
IF  Imputed DEBTINC < 33.837795539
AND Imputed DELINQ <          1.5
THEN
  NODE    :        4
  N       :      814
  1       :     4.1%
  0       :    95.9%

IF           4.5 <= Imputed DELINQ
THEN
  NODE    :        7
  N       :       40
  1       :    95.0%
  0       :     5.0%

IF  0.6204933586 <= Imputed DEROG
AND 33.837795539 <= Imputed DEBTINC < 33.86004
AND Imputed DELINQ <          1.5
THEN
  NODE    :       21
  N       :       73
```

To display the English Rules for the selected node, select **Edit** ⇨ **Tools** ⇨ **Display English Rules** from the Results window menu. The English Rules window is unavailable when there are multiple targets.

Tree Plot

A decision tree contains the following items:

- **Root node** — the top node of a vertical tree that contains all observations.
- **Internal nodes** — non-terminal nodes that contain the splitting rule. This includes the root node.
- **Leaf nodes** — terminal nodes that contain the final classification for a set of observations.

A default decision tree has the following properties:

- It is displayed in vertical orientation.
- The nodes are colored by the proportion of a categorical target value or the average of an interval target.
- The line width is proportional to the ratio of the number of observations in the branch to the number of observations in the root node.
- The line color is constant.

For more detailed information about changing these properties, see “Tree Properties Window” on page 761 .

You can select one or more nodes in a tree plot by doing one of the following:

- clicking or CTRL-clicking the nodes
- pressing the left mouse button and dragging the mouse to define a rectangle that contains the node or nodes that you want to select.

When you move your mouse pointer over a node of a tree, a text box displays information about the node. To display node text in the nodes of a tree, right-click in the tree and select **View** ⇨ **Node Text**.

The node text that is displayed in a tree depends on the target variable. For a categorical target variable, the text box contains separate rows for each target value and each decision. It provides information about the percentage of observations in a node for each target value. If the target is interval, the text box displays the number of observations in a node and the average value of the model assessment measure.

To zoom in or zoom out in the tree, right-click in the tree, select **View**, and then select the percentage that you want.

Treemap Plot

The Treemap plot displays a compact graphical display of the tree, which is similar to the tree in the Decision Tree window.

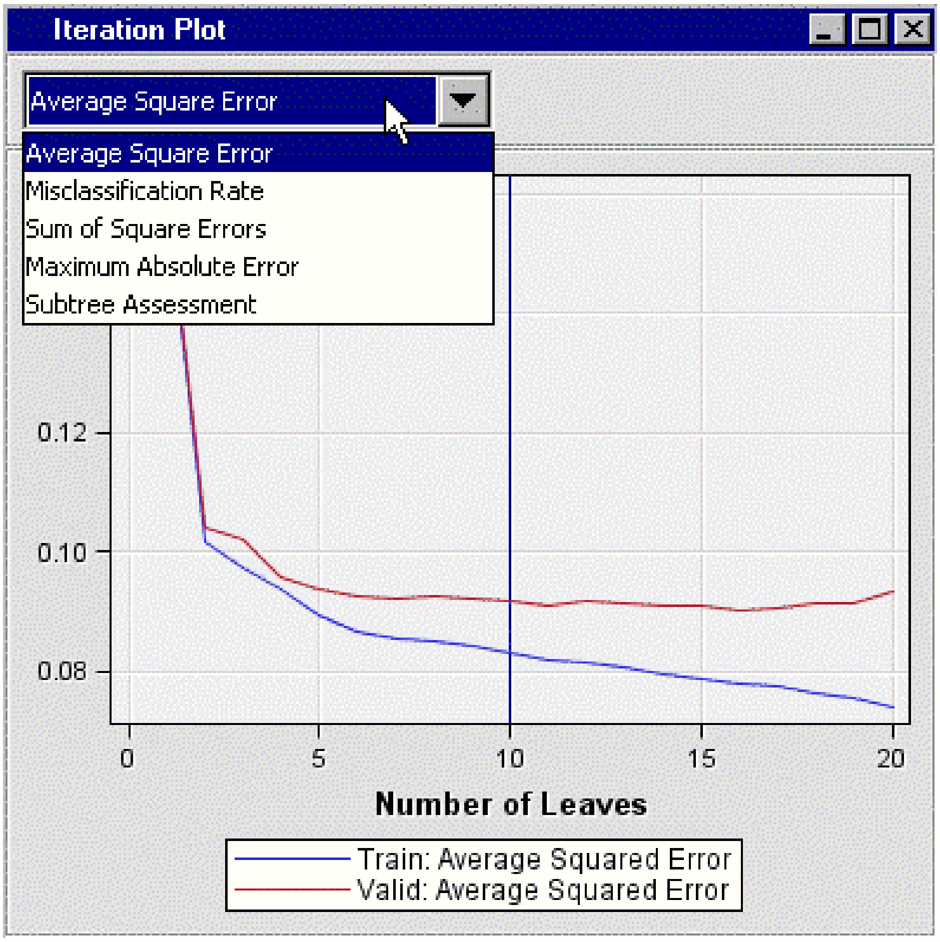
A default tree in the Treemap plot has the following properties:

- The nodes are colored by the proportion of a categorical target value or the average of an interval target.
- The node width is proportional to the number of observations in the node.

Selecting a node in the Treemap window automatically selects the same node in the Tree window.

Iteration Plot

The Iteration Plot window displays plots for different subtrees. The horizontal axis displays the number of leaves in a subtree. A reference line is drawn to indicate which subtree was selected as the final model.



Use the **Select Chart** list to choose from the following values that you can plot:

- Average Squared Error
- Misclassification Rate
- Sum of Squared Errors
- Maximum Absolute Error
- Subtree Assessment
- Average Profit (if decisions are defined)
- Total Profit (if decisions are defined)

The iteration plot is unavailable when there are multiple targets.

Variable Importance Table

Open the Results window of the **Decision Tree** node by right-clicking the node and selecting Results from the pop-up menu. Within the results output, select **View** and then **Model** from the pop-up menu. The Variable Importance table is accessed by next selecting **Variable Importance** from the pop-up menu. This table contains a measure of the relative importance of each input variable in the selected subtree.

The table contains the following columns:

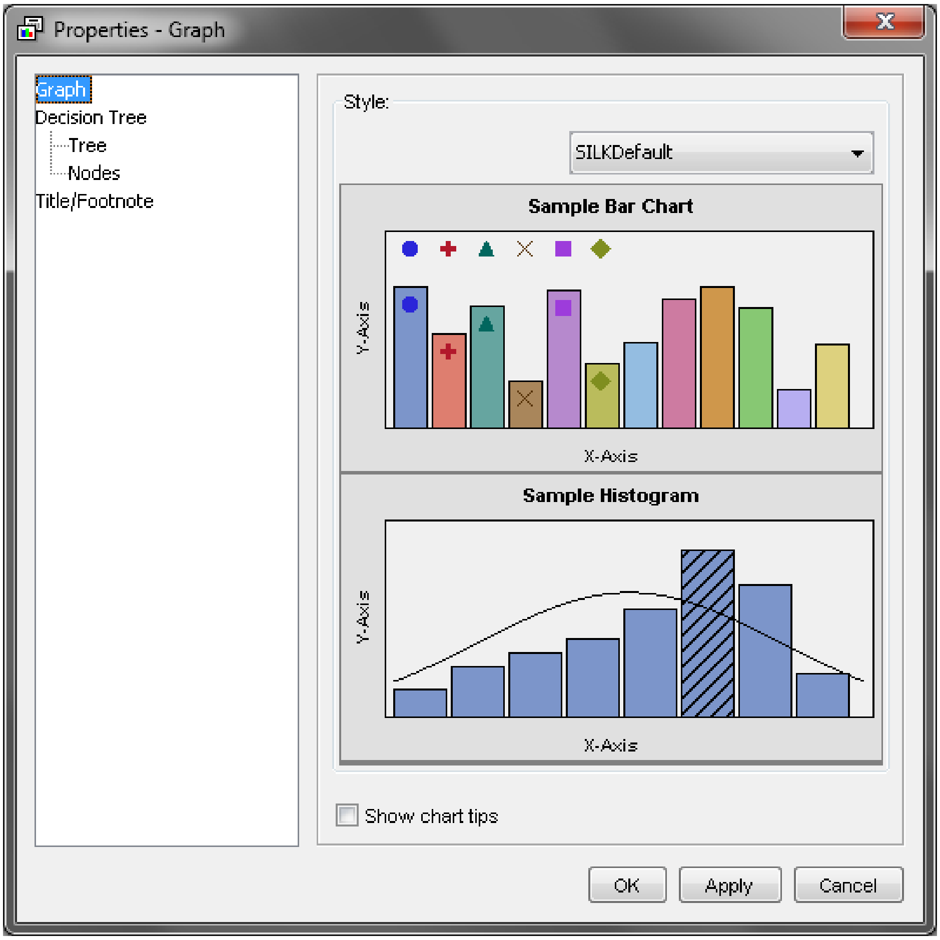
- Variable Name: The name of the selected variable.
- Label: The label given by the user for the selected variable.

- Number of Splitting Rules: The number of splitting rules that use the selected variable.
- Number of Surrogate Rules: the number of surrogate rules that use the selected variable if surrogate rules were generated.
- Importance: The observed value of the Variable Importance statistic for the training data set.
- Validation Importance: The observed value of the Variable Importance statistic for the validation data set.
- Ratio of Validation to Training Importance: The observed ratio of the Validation Variable Importance statistic to the Training Variable Importance statistic. A small ratio indicates a variable used in overly optimistic splitting rules. This value is set to missing if Training Importance is less than 0.0001.

Tree Properties Window

Graph Properties

You use the **Graph** properties of the Properties window to change the style of the tree plot and to control whether chart tips are displayed. To access the Properties — Graph window, right-click anywhere in the Tree window and select **Graph Properties**.

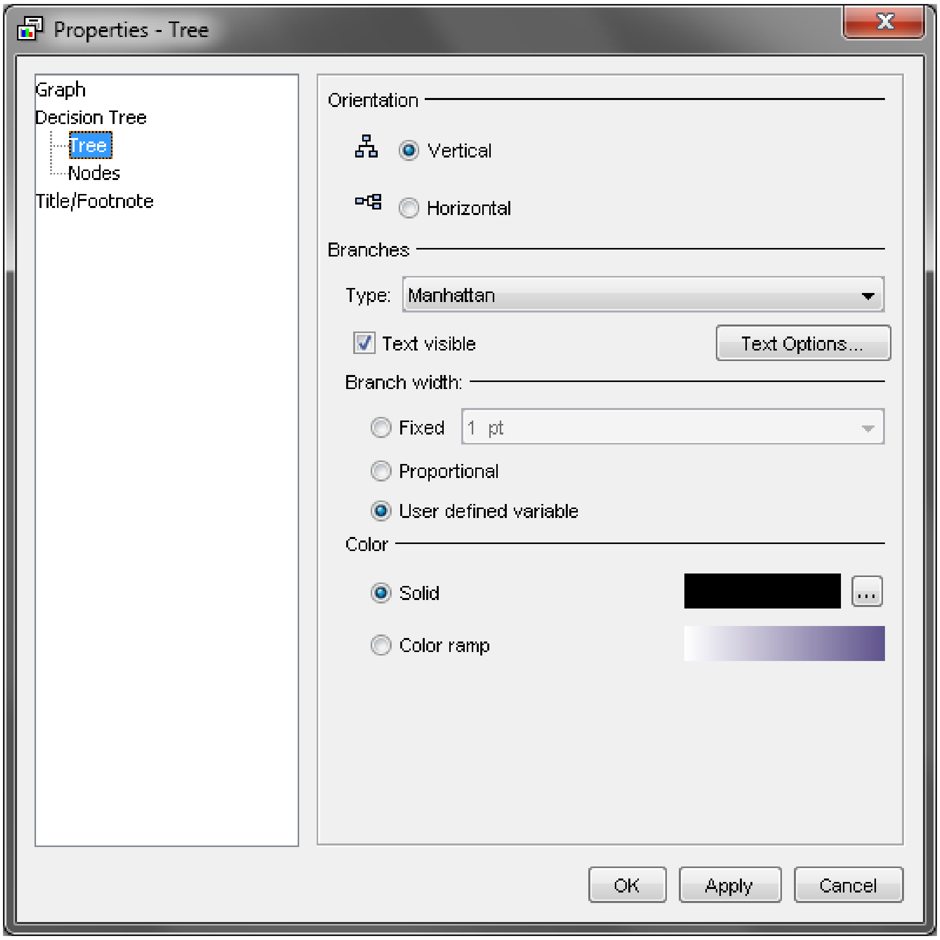

Properties - Graph

Style: SILKDefault

□ Show chart tips

Tree Properties

You use the **Tree** properties of the Properties window to specify the orientation of the tree and the following properties of the branches of a tree:



Orientation

● Vertical

○ Horizontal

Branches

Type: Manhattan

☑ Text visible

Branch width:

○ Fixed 1 pt

○ Proportional

● User defined variable

Color

● Solid

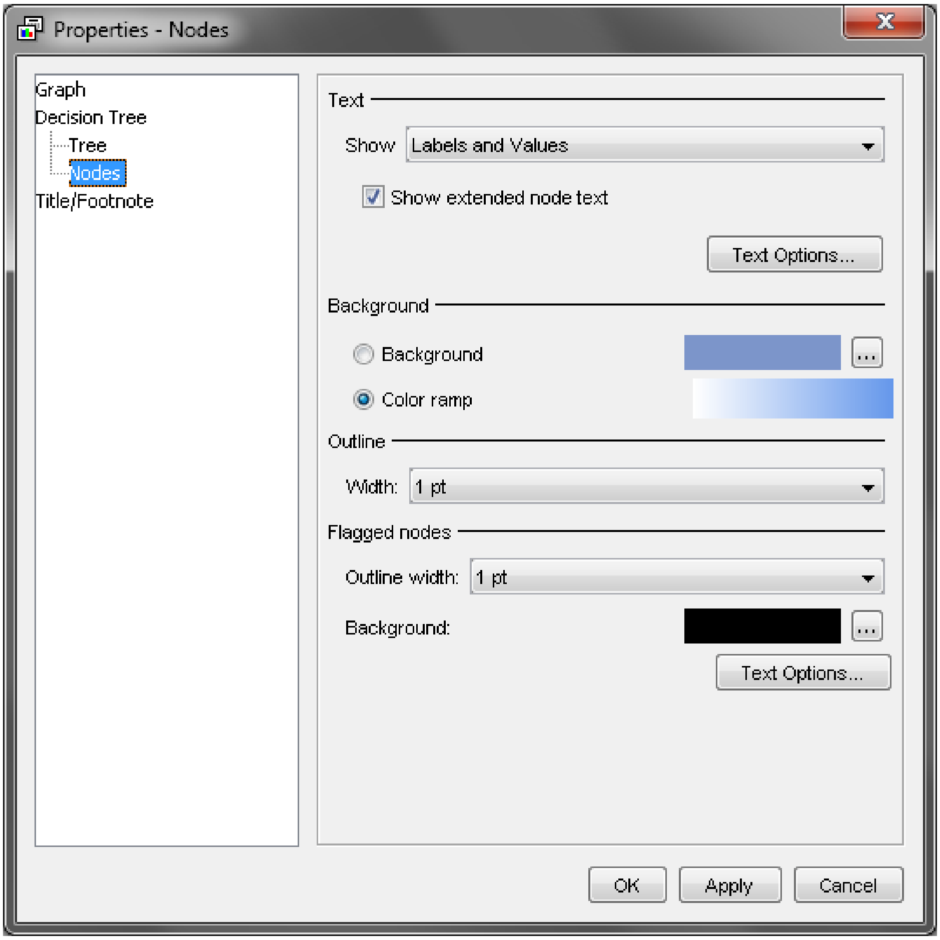
○ Color ramp

- **Orientation** — specifies whether to display the tree plot in the horizontal or the default vertical orientation.
- **Branches** — specifies the style of branch that appears between nodes in your tree plots. Available values are Manhattan, Straight, and Triangle. Manhattan uses orthogonal lines to display branches. Strait uses diagonal lines to display branches. Triangle shows the branches between nodes as solid triangles. The triangle apex begins at the parent node, and the triangle base abuts the successor node. If you select **Triangle** as your Branch Style, the Branch Width setting does not change the appearance of the tree.
- **Text visible** — Select the **Text Visible** check box if you want to display the splitting rule. When you enable the Text visible setting, you can click **Text Options** to configure the font, size, color, and style of the text.
- **Branch Width** — The Branch Width setting does not change the appearance of the tree when you use the triangle branch style. If you specify **Fixed**, the widths of all branches are fixed. You specify the line width by using the list. If you specify **Proportional**, the width of the branch is proportional to the ratio of the number of observations in the branch to the number of observations in the root node. If you specify **User Defined Variable**, the width of the branch is determined by a user-defined variable.

- **Color** — Select **Solid** to use a solid color for branches in the tree. Click ... to choose a different color. The Link Color window appears on the **Swatch** tab. The **Swatch** tab is a color swatch pallet. Click a swatch cell to choose a new color. If you prefer, you can fine-tune your color selection with the tools on the **RGB** and **HSB** tabs of the Link Color window. The **RGB** (Red-Green-Blue) tab contains a gradient RGB color tool. The **HSB** (Hue-Saturation-Brightness) tab contains color attribute settings

Nodes Properties

You use the **Nodes** properties of the Properties window to specify the following properties of the tree nodes:



Properties - Nodes

Graph
Decision Tree
Tree
Nodes
Title/Footnote

Text

Show: Labels and Values

☑ Show extended node text

Text Options...

Background

○ Background

● Color ramp

Outline

Width: 1 pt

Flagged nodes

Outline width: 1 pt

Background: ■ ...

Text Options...

OK Apply Cancel

- **Text** — Use the Text setting to configure how nodes are labeled in your tree plot. Select **Labels and Values** to display variable labels and values for each node. Select **Values Only** to display only variable values for each node. Select **No Text** to hide all text for the nodes.

- **Show extended node text** — Select the check box if you want to display the maximum information about a node. For example, with a binary target, when the check box is selected, each node displays the name of the target, The percentage of ones, the percentage of zeros, and the number of observations. When the check box is deselected, the percentage of ones is not displayed, because this can be inferred from the percentage of zeros.

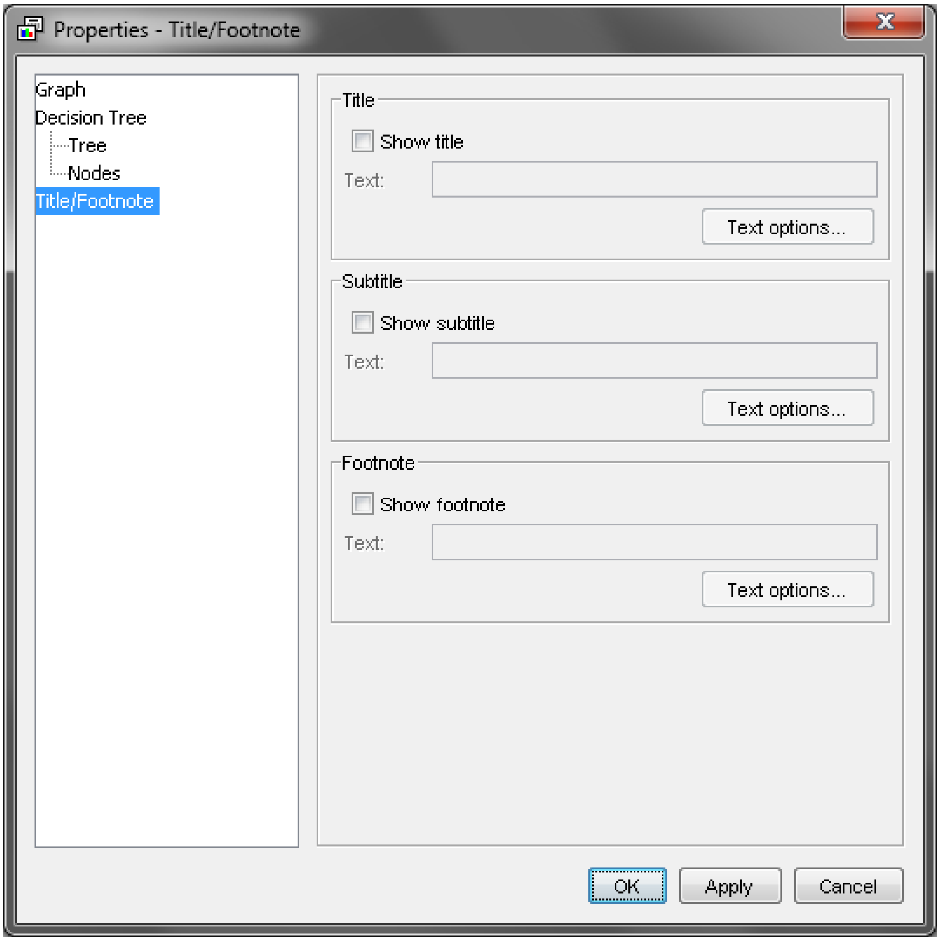
- **Text Options** — Use the Text Options setting to configure how node text appears in your tree plot. Click **Text Options** to open the Node Text Options window. The

Node Text Options window contains tabbed settings for the text font, size, color, style, and transparency. Close the Node Text Options window to return to the **Nodes** tab of the Tree Properties window.

- **Background** — Choose between two settings to configure the coloring or shading of node boxes in your tree plot. Use the **Background** setting to a solid color for the node boxes in the tree plot. Click ... to choose a different color. The Background Attributes window opens to a pallet display of color swatches. Choose a color scheme for your node box backgrounds.
- **Node Outline Width** — Use the Node Outline Width setting to specify the width of the node border. Use the drop-down list to choose a width for the node borders in your tree plot.
- **Flagged Node Outline Width** — Use the Flagged Node Outline Width setting to specify the width of flagged node borders. Use the list to choose a width for the flagged node borders in your tree plot.
- **Flagged Node Background** — Use the Flagged Node Background setting to a solid color for the flagged node boxes in the tree plot. Click on the ellipsis icon to the right of the color bar if you want to choose a different color. The Flagged Node Color window opens to a pallet display of color swatches. Choose a color scheme for your node box backgrounds and close the Flagged Node Color window to return to the **Nodes** tab of the Properties window.
- **Text Options** — enables you to configure how text appears in flagged nodes in your tree plot. Click **Text Options** to open the Flagged Node Text Options window. The Flagged Node Text Options window contains tabbed settings for text font, size, color, style, and transparency. Close the Flagged Node Text Options window to return to the **Nodes** tab of the Tree Properties window.

Title/Footnote Properties

You use the **Title/Footnote** tab of the Properties window to specify graph titles, subtitles, and footnotes:



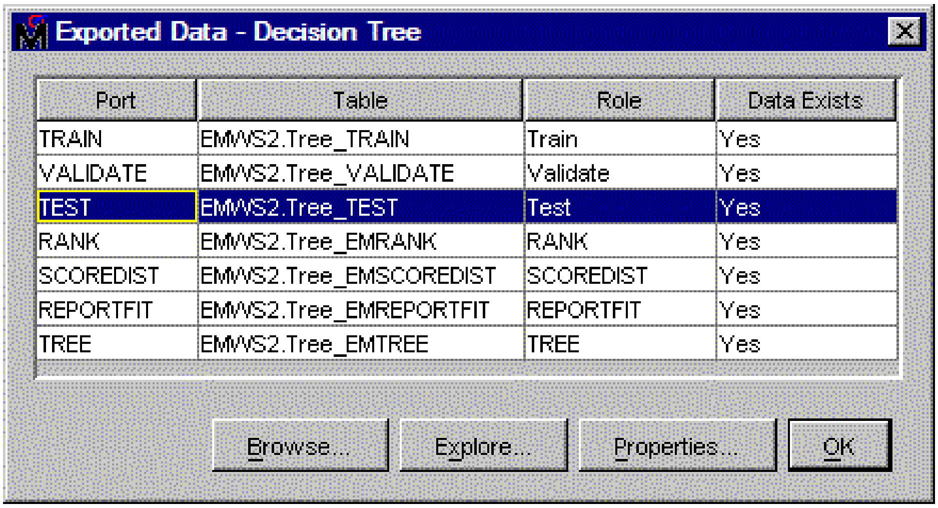
To specify a graph title, subtitle, or footnote, select the corresponding check box to enable the **Text** field. Enter your text in the field. Select the corresponding **Text options** button to specify the text font, size, style, color, and other text parameters.

Decision Tree Interactive Training

To launch an interactive training session in SAS Enterprise Miner, click the ... button at the right of the Interactive property in the properties panel. For more information about the Interactive Decision Tree on page 813 application, select **Help** ⇨ **Interactive Decision Tree Help** from the main menu of the Interactive Decision Tree window.

Decision Tree Node Output Data Sources

After the **Decision Tree** node has run, with the node selected in the Diagram Workspace, select the ... button to the right of the Exported Data property in the node properties panel. This opens the Exported Data — Decision Tree window.



Exported Data - Decision Tree

Port	Table	Role	Data Exists
TRAIN	EMWS2.Tree_TRAIN	Train	Yes
VALIDATE	EMWS2.Tree_VALIDATE	Validate	Yes
TEST	EMWS2.Tree_TEST	Test	Yes
RANK	EMWS2.Tree_EMRANK	RANK	Yes
SCOREDIST	EMWS2.Tree_EMSCOREDIST	SCOREDIST	Yes
REPORTFIT	EMWS2.Tree_EMREPORTFIT	REPORTFIT	Yes
TREE	EMWS2.Tree_EMTREE	TREE	Yes

You can select any output data source that has existing data, and examine it further.

- Click **Browse** to open a window where you can browse the data set.
- Click **Explore** to open the Explore window, where you can sample and plot the data.
- Click **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

Tree Methodology

Missing Values

If the value of the target variable is missing, the observation is excluded from training and evaluating the decision tree model.

The search for a split on an input uses observations whose values are missing on the input (provided that you did set the Missing Values property to **Use** in search). All such observations are assigned to the same branch. The branch might or might not contain other observations. The resulting branch maximizes the worth of the split.

For splits on a categorical variable, this amounts to treating a missing value as a separate category. For numerical variables, it amounts to treating missing values as having the same unknown nonmissing value.

One advantage of using missing data during the search is that the worth of the split is computed with the same number of observations for each input. Another advantage is that an association of the missing values with the target values can contribute to the predictive ability of the split.

When a split is applied to an observation in which the required input value is missing, surrogate splitting rules can be considered before assigning the observation to the branch for missing values.

A surrogate splitting rule is a backup to the main splitting rule. For example, the main splitting rule might use COUNTY as input and the surrogate might use REGION. If the COUNTY is unknown and the REGION is known, the surrogate is used.

If several surrogate rules exist, each surrogate is considered in sequence until one can be applied to the observation. If none can be applied, the main rule assigns the observation to the branch that is designated for missing values.

The surrogates are considered in the order of their agreement with the main splitting rule. The agreement is measured as the proportion of training observations that the surrogate rule and the main rule assign to the same branch. The measure excludes the observations to which the main rule cannot be applied. Among the remaining observations, those on which the surrogate rule cannot be applied count as observations not assigned to the same branch. Thus, an observation that has used a missing value on the input in the surrogate rule but not the input in the primary rule counts against the surrogate.

The Number of Surrogate Rules property determines the number of surrogates sought. A surrogate is discarded if its agreement is less than or equal to the largest proportion of observations in any branch. As a consequence, a node might have fewer surrogates specified than the number in the Number of Surrogate Rules property.

Unseen Categorical Values

A splitting rule that uses a categorical variable might not recognize all possible values of the variable. Some categories might not have been in the training data. Others might have been so infrequent in the within-node training sample that the ARBORETUM procedure excluded them. The MINCATSIZE= option in the TRAIN statement specifies the minimum number of occurrences required for a categorical value to participate in the search for a splitting rule. Splitting rules treat unseen categorical values as they would missing values.

Method of Split Search

For a specific node and input, the **Decision Tree** node seeks the split that maximizes the measure of worth associated with the splitting criterion specified. Some measures are based on a node impurity measure while others are based on p-values of a statistical test. P-values can be adjusted for the number of branches and input values, the depth of the node in the tree and the number of independent input variables for which candidate splits exist in the node.

The measure of worth depends on the criterion property that is selected. The ENTROPY, GINI, and VARIANCE reduction criteria measure worth as follows:

```
I(node) - sum over branches b of P(b) I(b)
```

Here, I(node) denotes the entropy, Gini, or variance measure in the node, and P(b) denotes the proportion of observations in the node that is assigned to branch b. If prior probabilities are specified, then the proportions P(b) are modified accordingly.

- For ENTROPY, $$I(\text{node}) = -\sum_{\textit{all classes}} P_{class} \log_2 P_{class}$$
- For GINI,
$$I(\text{node}) = \left(1 - \sum_{i}^{\textit{classes}} \left(\frac{\text{number of class } i \text{ cases}}{\text{all cases in node}}\right)^2\right)$$
- For VARIANCE, $$I(\text{node}) = \sum_{\textit{obs } i} (Y_i - \overline{Y})^2$$

Here, Y-bar is the average Y in the node.

The Chi-square test and F test criteria use the -log(p-value) measure. For these criteria, the best split is the one with the smallest p-value. By default, the p-values are adjusted to take into account multiple testing. If the Bonferroni adjustment is applied before the split is selected, then the best split is the one with the smallest Bonferroni adjusted p-value.

For nodes with many observations, the algorithm uses a sample for the split search, for computing the worth, and for observing the limit on the minimum size of a branch. The Node Sample property specifies the size of the sample. The samples in different nodes are taken independently.

For binary, nominal, and ordinal targets, the sample is as balanced as possible. Suppose, for example, that a node contains 100 observations of one value of a binary target, 1000 observations of the other value, and Node Sample=200 or more. Then all 100 observations of the first target value are in the node sample.

For binary splits on binary or interval targets, the optimal split is always found. For other situations, the data is first consolidated, and then either all possible splits are evaluated or a heuristic search is used.

The consolidation phase searches for groups of values of the input that seem likely to be assigned the same branch in the best split. The split search regards observations in the same consolidation group as having the same input value. The split search is faster because fewer candidate splits need evaluating.

If, after consolidation, the number of possible splits is greater than the number specified in the Exhaustive property, then a heuristic search is used.

The heuristic algorithm alternately merges branches and reassigns consolidated groups of observations to different branches. The process stops when a binary split is reached. Among all candidate splits considered, the one with the best worth is chosen. The heuristic algorithm initially assigns each consolidated group of observations to a different branch, even if the number of such branches is more than the limit allowed in the final split. At each merge step, the two branches that degrade the worth of the partition the least are merged. After two branches are merged, the algorithm considers re-assigning consolidated groups of observations to different branches. Each consolidated group is considered in turn, and the process stops when no group is re-assigned.

When using the Chi-square test or F test criteria, the p-value of the selected split on an input is subjected to more adjustments: Kass after or before choosing the number of branches, Depth, and Effective number of inputs. If the adjusted p-value is greater than or equal to the worth value, the split is rejected.

Automatic Pruning and Subtree Selection

After a node is split, the newly created nodes are considered for splitting. This recursive process ends when no node can be split.

The reasons a node will not split are as follows:

- The node contains too few observations, as specified in the Leaf Size property option.
- The number of nodes in the path between the root node and the given node equals the number that is specified in the Maximum Depth property.
- No split exceeds the threshold worth requirement that is specified in the F test or Chi-square significance level value.

The last reason is the most informative: either all the observations in the node contain nearly the same target value, or no input is sufficiently predictive in the node.

A tree adapts itself to the training data and, generally, does not fit as well when applied to new data. Trees that fit the training data too well often predict too poorly to use on new data.

A primary consideration when developing a tree for prediction is deciding how large to grow the tree or (what amounts to the same in the end) which nodes to prune off the tree.

After the sequence of subtrees is established, the **Decision Tree** node uses one of three methods to select which subtree to use for prediction. You set the desired subtree method by using the **Method** property. The **Method** property can be set to **Assessment**, **Largest**, or **N**.

If the **Method** property under Subtree grouping is set to **Assessment**, then the **Decision Tree** node uses the smallest subtree with the best assessment value. This smallest subtree in turn depends on the **Assessment Measure** property value. The assessment is based on the validation data when available. (This differs from the construction of the sequence of subtrees that only uses the training data.)

If the **Method** property under Subtree grouping is set to **Largest**, then the **Decision Tree** node uses the largest subtree after it prunes the nodes that do not increase the assessment based on the training data. For nominal targets, the largest subtree in the sequence might be much smaller than the original unpruned tree because a splitting rule might have a significant split without increasing the number of observations that are correctly classified.

Comparison with Other Tree Methods

CHAID

The following discussion provides a brief description of the CHAID (chi-square automatic interaction detection) algorithm for building decision trees, including how the CHAID algorithm differs from the **Decision Tree** node and how it can be approximated.

For CHAID, the inputs are either nominal or ordinal. Many software applications accept interval inputs and automatically group the values into ranges before growing the tree.

The splitting criteria is based on p-values from the F-distribution (interval targets) or Chi-square distribution (nominal targets). The p-values are adjusted to accommodate multiple testing.

A missing value can be treated as a separate value. For nominal inputs, a missing value constitutes a new category. For ordinal inputs, a missing value is free of any order restrictions.

The search for a split on an input proceeds stepwise. Initially, a branch is allocated for each value of the input. Branches are alternately merged and re-split as seems warranted by the p-values. The original CHAID algorithm by Kass stops when no merge or re-splitting operation creates an adequate p-value. The final split is adopted. A common alternative, sometimes called the exhaustive method, continues merging to a binary split and then adopts the split with the most favorable p-value among all splits the algorithm considered.

After a split is adopted for an input, its p-value is adjusted, and the input with the best adjusted p-value is selected as the splitting variable. If the adjusted p-value is smaller than a threshold that you specified, then the node is split.

Tree construction ends when all the adjusted p-values of the splitting variables in the unsplit nodes are above the user-specified threshold.

The CHAID algorithm differs from the **Decision Tree** node in a number of ways:

- The **Decision Tree** node seeks the split minimizing the adjusted p-value, whereas the original KASS algorithm does not.
- The CHAID exhaustive method is similar to the **Decision Tree** node heuristic method, except that the exhaustive method "re-splits" and the **Decision Tree** node "re-assigns".
- CHAID software discretizes interval inputs; the **Decision Tree** node sometimes consolidates observations into groups.
- The **Decision Tree** node searches on a within-node sample, unlike CHAID.

How to Approximate the CHAID Method with the Decision Tree Node

To approximate CHAID, the interval inputs should first be discretized into a few dozen values.

You should then set the following **Decision Tree** node properties:

- For nominal targets:
  - Set the **Nominal Criterion** property to **PROBCHISQ**.
- For interval targets:
  - Set the **Interval Criterion** property to **PROBF**.
- For either nominal or interval targets:
  - To avoid automatic pruning, set the **Method** property to **Largest**.
  - Set the Chi-square or F test **Significance Level** to 0.05 (or whatever significance level seems appropriate).
  - Set the **Maximum Branch** property to the maximum number of categorical values in an input.
  - Set the **Number of Surrogate Rules** property to 0.
  - To force a heuristic search, set the **Exhaustive** property to 0.
  - Set the **Leaf Size** to 1.
  - Set the **Split Size** property to 2.
  - Set the **Bonferroni Adjustment** property to **Yes**, and the **Time of Kass Adjustment** property to **After**.

  The CHAID method does not apply to ordinal targets. The methodology for ordinal targets included in CHAID software is fundamentally different, and originated a decade after the CHAID work.

Classification and Regression Trees

The following discussion provides a description of the Breiman, Friedman, Olshen, and Stone (BFOS) Classification and Regression Trees method for building decision trees. To learn more about this method, read *Classification and Regression Trees* by L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone.

The **Decision Tree** node options that are necessary to approximate the BFOS method are also provided in the following discussion.

The available splitting criteria are as follows:

- For interval targets, reduction in variance and reduction in least-absolute-deviation.

- For nominal targets, Gini impurity and twoing.
- For ordinal targets, ordered twoing.
- For binary targets, Gini, twoing, and ordered twoing create the same splits. Twoing and ordered twoing were used infrequently.

The BFOS method does an exhaustive search for the best binary split. Linear combination splits are also available. Using a linear combination split, an observation is assigned to the "left" branch when a linear combination of interval inputs is less than some constant. The coefficients and the constant define the split. The BFOS method for searching for linear combination splits is heuristic and might not find the best linear combination.

When splits are being created, observations with a missing value in the splitting variable (or variables in the case of linear combination) are omitted. Surrogate splits are also created and used to assign observations to branches when the primary splitting variable is missing. If missing values prevent the use of the primary and surrogate splitting rules, then the observation is assigned to the largest branch (based on the within node training sample).

When a node contains many training observations, a sample is taken and used for the split search. The samples in different nodes are independent.

For nominal targets, prior probabilities and misclassification losses are recognized. A misclassification loss matrix can be incorporated in the splitting criteria. The tree is grown to overfit the data. A sequence of subtrees is formed using a cost-complexity measure. The subtree with the best fit to validation data is selected. Cross validation is available when validation data is not.

For nominal targets, class probability trees are an alternative to classification trees. Trees are grown to produce distinct distributions of class probabilities in the leaves. Trees are evaluated in terms of an overall Gini index. Neither misclassification costs nor rates are used.

How to Approximate the BFOS Classification and Regression Method with the Decision Tree Node

Trees created by using the **Decision Tree** node should be very similar to trees grown by using the BFOS Classification and Regression methods without linear combination splits or twoing or ordered twoing splitting criteria, and without incorporating a profit/loss matrix into the tree construction.

Twoing, Ordered Twoing, and the least-absolute-deviation (LAD) splitting criteria are not available in the **Decision Tree** node. Also unavailable are linear combination splits and cross validation.

With the exception of the items noted above, the SAS Enterprise Miner CART methodology is complete and exact. SAS Enterprise Miner CART results can vary from other CART methodologies that use approximations.

The BFOS method recommends using validation data unless the data set contains too few observations. The **Decision Tree** node is intended for large data sets, so first use a predecessor **Data Partition** node to divide the input data source into training and validation data.

You should then set the following options in the **Decision Tree** node:

- For nominal targets:
  - Set the **Nominal Criterion** property to **Gini**.
- For interval targets:

  - Set the **Interval Criterion** property to **Variance**.
- For nominal or interval targets:
  - Set the **Maximum Branch** property to 2.
  - Set the **Missing Values** property to **Largest Branch**.
  - Set the **Number of Surrogates Rules** property to 5.
  - Set the **Method** property to **Assessment** (for class probabilities, set the **Assessment Measure** property to **Average Square Error**).
  - Set the **Node Sample** property to a number greater than or equal to all observations or whatever setting is deemed appropriate for the methodology of BFOS.
  - Set the **Exhaustive** property to a very large number to enumerate all possible splits. For example, 2,000,000,000 (2 billion) might suffice.
  - Import a validation data set into the **Decision Tree** node. The validation data set is used in conjunction with the Assessment subtree method.
  - When using prior probabilities and misclassification losses, the BFOS method incorporates these into the split search. The Decision Tree incorporates them the same way when the property "incorporate priors/profits in the split search" is selected.
  - SAS Enterprise Miner does not specify equal priors by default. You must use the Decision Tree Properties Panel to specify equal priors.

The BFOS classification and regression method of handling ordinal targets is unavailable in the **Decision Tree** node.

Normalization of Category Values

Categorical values (class variable level values) in SAS Enterprise Miner are normalized as follows:

- left-justified
- uppercase
- truncated to a length of 32 characters (after left-justification)

This implies, for example, that values such as "YES", "yes", "Yes", and " yes" are all considered the same. Likewise, "a big bright brilliantly crimson red ball" and "a big bright brilliantly crimson red cap" are also considered identical, because they are not unique within the first 32 characters.

Normalized values are stored in the DMDB catalog. The modeling tools work with normalized class values for both modeling as well as scoring.

New variable names that are constructed from category values (such as P_xxx) are based on normalized values.

The normalization process is aimed to provide consistent behavior in handling category values throughout the product.

References

Breiman, L. “Random Forests.” 2001. *Machine Learning* 45: 5–32.

Breiman, L 2002. “Manual on Setting Up, Using, and Understanding Random Forests V3.1.” unpublished manuscript, available at http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf, University of California – Berkeley.

Breiman, L 2003. “Manual on Setting Up, Using, and Understanding Random Forests V4.0.” unpublished manuscript, available at http://oz.berkeley.edu/users/breiman/Using_random_forests_v4.0.pdf, University of California – Berkeley.

Chapter 50
Dmine Regression Node

Dmine Regression Node

Overview of the Dmine Regression Node

The Dmine Regression node performs the following tasks:

- Computes a forward stepwise least-squares regression. In each step, an independent variable is selected that contributes maximally to the model R-square value.
- Optionally computes all 2-way interactions of classification variables.
- Optionally uses AOV16 variables to identify non-linear relationships between interval variables and the target variable.
- Optionally uses group variables to reduce the number of levels of classification variables.
- For a binary target (class response variable), a fast algorithm for (approximate) logistic regression is computed. The independent variable is the prediction from the former least squares regression. Since only one regression variable is used in the logistic regression, only two parameters are estimated, the intercept and slope. The range of predicted values is divided into a number of equidistant intervals (knots), on which the logistic function is interpolated.

See the “Predictive Modeling” on page 149 section for information that applies to all of the predictive modeling nodes.

Dmine Regression Node Data Set Requirements

The Dmine Regression node performs a regression analysis on data sets that have a binary or interval level target variable. If you want to create a regression model on data that contains a nominal or ordinal target, then you should use the Regression node.

The manner is which missing values are handled depends on the type of variable.

- Missing values in the categorical input variable are treated as an additional category.
- Missing values in an interval input variable are replaced by the mean of that variable. If there is a variable that is assigned the role of FREQ, the weighted mean of the interval input variable is used.
- Observations that have missing values in the target variables are excluded from the analysis.

Dmine Regression Node Properties

Dmine Regression Node General Properties

The following general properties are associated with the Dmine Regression node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Decision Tree node added to a diagram has a Node ID of DMineReg. The second Dmine Regression node added to a diagram has a Node ID of DMineReg2, and so on.
- **Imported Data** — accesses the Imported Data — Dmine Regression window. The Imported Data — Dmine Regression window contains a lists of the ports that provide data sources to the Dmine Regression node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data— Dmine Regression window. The Exported Data— Dmine Regression window contains a list of the output data ports that the Dmine Regression node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Dmine Regression Node Train Properties

The following train properties are associated with the Dmine Regression node:

- **Variables** — specifies the properties of each variable in the data source that you want to use. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can set a variable's report status to **Yes** or **No**.
- **Maximum Variable Number** — Use the Maximum Variable Number property to specify the upper bound for the number of independent variables that you want to allow to be selected for the model. The upper bound is for the number of rows and columns of the X'X matrix of the regression problem. Permissible values are positive integers. The default setting for the Maximum Variable Number property is 3000.

Dmine Regression Node Train Properties: R-Square Options

- **Minimum R-Square** — specifies a lower bound for the individual R-square value of a variable to be eligible for the model selection process.
- **Stop R-Square** — specifies a lower bound value for the incremental model R-square value.

Dmine Regression Node Train Properties: Created Variables

- **Use AOV16 Variables** — when set to **Yes**, this property bins interval variables into sixteen equally-spaced groups to help identify non-linear relationships with the target.
- **Use Group Variables** — when set to **Yes**, this property tries to reduce the levels of each class variable to a group variable, based on the relationship with the target.
- **Use Interactions** — when set to **Yes**, this property enables the inclusion of all possible 2-way interactions of class variables in the model.
- **Print Option** — selects a print option that controls how much output detail is included.
  - **Default** — suppresses some details of outputs.
  - **All** — all output details print.
  - **None** — suppresses all printed outputs.
- **Use SPDE Library** — Use this property to indicate whether you want to use the multithreading processing capabilities provided by SAS Scalable Performance Data Engine. The default setting is **Yes**.

Dmine Regression Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

How the Dmine Regression Node Groups Class Inputs

The group class variables method can be viewed as an analysis of the ordered cell means to determine if specific levels of a class input can be collapsed into a single group. For example, suppose that you have a four level class input that explains 20% of the total variation in the target (R-square value = .20). The node first orders the input levels by the mean responses. The ranking determines the order that the node considers the input levels for grouping. The node always works from top to bottom when considering the levels that can be collapsed.

Target Mean	Input Level
90	C
85	B
50	D
42	A

The node combines the first two levels (C and B) into a single group if the reduction in the explained variation is less than or equal to 5% of the target variable (R-square value of at least .19). If the C and B levels can be combined into a group, then the node determines if CB and D can be collapsed. If C and B cannot be combined, then the node determines if B and D can be combined into one group. The node stops combining levels when the R-square threshold is not met for all possible ordered, adjacent levels or when it reduces the original variable to two groups.

How the AOV16 Variables Identify Non-Linear Relationships with the Target

The following example explains how the AOV16 variables identify nonlinear relationships with the target. Suppose that you have some data where Y represents income and X is age. The age variable has 5 levels with replicate income measurements at each age level. You could perform a simple linear regression using the following code:

```
PROC GLM DATA=WORK.INCOME;
   MODEL Y = X;
RUN;
```

In this case the covariate (regressor) X would support 1 degree of freedom. You could also run the following model:

```
PROC GLM DATA=WORK.INCOME;
   CLASS X;
   MODEL Y = X;
RUN;
```

In this case, X would support 4 degrees of freedom and the source of variation explained by X would be equivalent to a fourth degree polynomial model:

```
PROC GLM DATA=WORK.INCOME;
   MODEL Y = X X*X  X*X*X  X*X*X*X;
RUN;
```

Thus the AOV16, which can account for at most 15 degrees of freedom, can be thought of as a way of explaining a single degree covariate in a nonlinear fashion. Since the covariate (input) typically has more than 16 levels, the above regression won't work out exactly but the spirit of the idea is the same.

Dmine Regression Node Results

You can open the Results window of the Dmine Regression node by right-clicking the node and selecting **Results** from the menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Dmine Regression node properties configuration when the node was last run.
  - **Run Status** — displays the status of the Dmine Regression node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — displays a table of the variables in the training data set.
  - **Train Code** — displays the code that Enterprise Miner used to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — the SAS log of the Dmine Regression run.
  - **Output** — displays a report that includes:
    - variable summary
    - model events summary
    - table of predicted and decision variables
    - table of variable R-Square values
    - table of effects chosen for the target variable
    - ANOVA table for the target variables
    - table of effects not chosen for the target variable
    - estimating logic table
    - classification tables for train, validate, and test data
    - fit statistics tables for train, validate, and test data
    - event classification table
    - assessment score rankings for train, validate, and test data
    - assessment score distributions

- **Flow Code** — the SAS code used to produce the output that the Dmine Regression node passes on to the next node in the process flow diagram.

- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 .

- **Assessment**
  - **Fit Statistics** — displays a table of fit statistics from the model.
  - **Classification Chart** — displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represent the percentage of total observations.
  - **Score Rankings Overlay** — several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles and observations in a decile are used to calculate the statistics that are plotted in deciles charts. The Score Rankings Overlay plot displays both train and validate statistics on the same axis.

    By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays

    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response
    - Total Profit
    - Expected Profit
  - **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

    For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

    The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores.

For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are

- Percentage of Events — Categorical Targets
- Number of Events — Categorical Targets
- Cumulative Percentage of Events — Categorical Targets
- Expected Profit — Categorical Targets
- Report Variables — Categorical Targets
- Mean for Predicted — Interval Targets
- Max. for Predicted — Interval Targets
- Min. for Predicted — Interval Targets

- **Model** — graphs and tables with information about the variables in the model. The available graphs and tables are
  - **Effects in the Model** — a histogram ranking variable effects by their sequential R-Square scores.
  - **Group Variables** — displays a read-only table of:
    - Name
    - Group
    - Variable
    - Level
  - **AOV16 Variables** — displays a read-only table of:
    - Name
    - Group
    - Variable
    - Bin Cutoff
  - **Group Interactions** — displays a read-only table of:
    - Name
    - Group
    - Variable 1
    - Level 1
    - Variable 2
    - Level 2
- **Table** — displays a table that contains the underlying data that is used to produce a chart.
- **Plot** — use the Graph Wizard to modify an existing Results plot or create a Results plot of your own.

Chapter 51
DMNeural Node

DMNeural Node

Overview of the DMNeural Node

The DMNeural node enables you to fit an additive nonlinear model that uses the bucketed principal components as inputs to predict a binary or an interval target variable.

The algorithm that is used in DMNeural network training was developed to overcome the following problems of the common neural networks for data mining purposes. These problems are likely to occur especially when the data set contains highly collinear variables.

- Nonlinear estimation problem — The nonlinear estimation problem in common neural networks is seriously underdetermined, which yields to highly rank-deficient Hessian matrices and results in extremely slow convergence of the nonlinear optimization algorithm. In other words, the zero eigenvalues in a Hessian matrix correspond to long and very flat valleys in the shape of the objective function. The traditional neural network approach has serious problems to decide when an estimate is close to an appropriate solution and the optimization process can be prematurely terminated. This is overcome by using estimation with full-rank Hessian matrices of a few selected principal components in the underlying procedure.
- Computing time — Each function call in common neural networks corresponds to a single run through the entire training data set; normally many function calls are needed for convergence of the nonlinear optimization. This requires a tremendous calculation time to get an optimized solution for data sets that have a large number of observations. In DMNeural network training of the DMNeural node, we obtain a set of grid points from the selected principal component and a multidimensional

frequency table from the training data set for nonlinear optimization. In other words, segments of the data are trained instead of the entire data, and the computing time is reduced dramatically.

- Finding global optimal solution — For the same reason, common neural network algorithms often find local rather than global optimal solutions and the optimization results are very sensitive with respect to the starting point of the optimization. However, the DMNeural network training can find a good starting point that is less sensitive to the results, because it uses well specified objective functions that contain a few parameters and can do a very simple grid search for the few parameters.

In the DMNeural training process, a principal components analysis is applied to the training data set to obtain a set of principal components. Then a small set of principal components is selected for further modeling. This set of components shows a good prediction of the target with respect to a linear regression model with an R2 selection criterion. The algorithm obtains a set of grid points from the selected principal component and a multidimensional frequency table from the training data set. The frequency table contains count information of the selected principal components at a specified number of discrete grid points.

In each stage of the DMNeural training process, the training data set is fitted with eight separate activation functions. The DMNeural node selects the one that yields the best results. The optimization with each of these activation functions is processed independently. The following table lists the eight activation functions that are used in the DMNeural training process:

Table of Activation Functions

Function Name	**Function Expression of Input x**
SQUARE	(A + Bx)x
TANH	A * tanh(Bx)
ARCTAN	A * arctan(Bx)
LOGIST	$\frac{\exp(ax)}{1+\exp(bx)}$
GAUSS	A * exp(-[Bx]^2)
SIN	A * sin(Bx)
COS	A * cos(Bx)
EXP	A * exp(Bx)

In the DMNeural node, the link function depends on the type of the target variable. By default, the IDENT and LOGIST functions are used for a binary target and an interval target, respectively.

Table of Link Functions

Function Name	Function Expression of Input x
INDET	x
LOGIST	$\frac{\exp(x)}{1+\exp(x)}$

In the first stage, the response variable is used as the target variable. Starting with the second stage, the algorithm uses the residuals of the model from the previous stage as the new target variables. In other words, model estimation is an additive stage-wise process. The response variable y is modeled in the first stage only. For each of the other stages, the residuals from the previous stage are modeled. The selection of the best activation function at each stage in the training process is based on the smallest SSE (Sum of Squared Error) or the smallest misclassification rate. In the final stage, an additive nonlinear model is generated as the final predicted model.

See the Predictive Modeling section for information that applies to all of the predictive modeling nodes.

Variable Requirements for the DMNeural Node

The DMNeural node requires one target variable, either binary or interval, and at least two input variables to perform the DMNeural network training.

You may also specify the following variables:

- Cost — contains the cost of a decision. You should first assign the cost model role to the appropriate variables when you create the data source. You can then assign a cost variable to each decision when you define a profit matrix with costs in the target profile for the target. For more information about defining a profit matrix with costs, see the Target Profiler section.
- Frequency — a variable that represents frequency of occurrences in each observation. Observations that have a missing or negative value for the frequency variable are excluded from the analysis. You can assign the freq model role to the appropriate variable when you create the data source.

Missing Values in the DMNeural Node

Observations that contain missing values in the target variable are excluded from the analysis. However, the predicted values of these observations are still computed.

Observations that contain missing values in the input variables are imputed automatically by the node:

- For numeric variables, missing values are imputed by the mean of the variables.
- For class variables, missing values are treated as an additional category.

DMNeural Node Properties

DMNeural Node General Properties

The following general properties are associated with the AutoNeural node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first DMNeural node added to a diagram hasa Node ID of DMNeural. The second DMNeural node added to a diagram has a Node ID of DMNeural2, and so on.
- **Imported Data** — accesses the Imported Data — DMNeural window. The Imported Data — DMNeural window contains a lists of the ports that provide data sources to the DMNeural node. Click the ellipsis button ... to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — DMNeural window. The Exported Data — DMNeural window contains a list of the output data ports that the DMNeural node creates data for when it runs. Click the ellipsis button ... to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

DMNeural Node Train Properties

The following train properties are associated with the DMNeural node:

- **Variables** — specifies the properties of each variable in the data source that you want to use. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can set a variable's report status to **Yes** or **No**.

DMNeural Node rain Properties: DMNeural Network

- **Lower Bound R2** — Use the Lower Bound R2 property of the DMNeural Network node to specify the lower R-square bound that you want to use to select a small set of principal components at each stage of the optimization process. Permissible values are real numbers between 0 and 1. The default value is 5.0E-5.

- **Max Component** — Use the Max Component property of the DMNeural Network node to specify the maximum number of principal components that you want to use to predict target variable values. Permissible values are integers between 2 and 6. The default value is 3.
- **Max EigenVector** — Use the Max EigenVector property of the DMNeural Network node to specify the upper bound that you want to use for the number of eigenvectors that are available for selection. Permissible values are integers greater than or equal to 2. The default value is 400.
- **Max Function Call** — Use the Max Function Call property of the DMNeural Network node to specify the upper bound that you want to use for the number of function calls in each optimization. Permissible values are integers greater than or equal to 1. The default value is 500.
- **Max Iteration** — Use the Max Iteration property of the DMNeural Network node to specify the upper bound that you want to use for the number of iterations performed during each optimization. Permissible values are integers greater than or equal to 1. The default value is 200.
- **Max Stage** — Use the Max Stage property of the DMNeural Network node to specify an upper bound for the number of stages utilized in the DMNeural optimization. Permissible values are integers between 1 and 10. The default value is 3.

DMNeural Node Train Properties: Convergence Criteria

- **Absolute Gradient** — Use the Absolute Gradient property of the DMNeural Network node to set the Gradient convergence criterion. Permissible values are real numbers greater than 0. The default value is 5.0E-4.
- **Gradient** — Use the Gradient property of the DMNeural Network node to set the Gradient convergence criterion. Permissible values are real numbers greater than 0. The default value is 1.0E-8.

DMNeural Node Train Properties: Model Criteria

- **Selection** — Use the Selection property of the DMNeural Network node to specify a criterion where, for the best activation function at each stage in the training process.
  - **Default** — The default setting is the objective function in the optimization criterion that you specified in the Optimization property.
  - **SSE** — the sum of squared errors.
  - **ACC** — the number of correctly classified observations divided by the total number of training observations.
- **Optimization** — Use the Optimization property of the DMNeural node to specify the objective function that you want to be optimized in the iterated network training process. The choices are
  - **SSE** (default) — The sum of squared errors is minimized.
  - **ACC** — The correct classification rate, that is, the number of correctly classified observations divided by the total number of training observations, is maximized.
- **Print Option** — Use the Print Option property of the DMNeural Network node to specify the level of detail that for node output.
  - **Default** — prints the standard node output.

- **PAll** — prints an extended version of the standard node output. This is the default setting.
- **PShort** — prints an abbreviated set of node outputs.
- **Noprint** — no print output is produced.

DMNeural Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

DMNeural Node Results

You can open the Results window of the DMNeural node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select View from the main menu to view the following information in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the DMNeural node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the DMNeural node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — opens a window and displays (read-only) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — the SAS log of the DMNeural node run.
  - **Output** — the SAS output of the DMNeural node run displays the following:
    - variable summary
    - model events summary

- table of predicted and decision variables
- component selection table for SS(y) and R2
- PROC DMNEURL summary
- response profile for the target variable
- table of PROC DMNEURL variable statistics
- stagewise goodness-of-fit criterion tables
- stagewise component selection tables
- summary table across stages
- fit statistics for the target variable
- classification table for the target variable
- event classification table
- assessment score rankings
- assessment score distribution table

- **Flow Code** — the SAS code used to produce the output that the DMNeural node passes on to the next node in the process flow diagram.

- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications. SAS Code contains DMNeural values of input variables only. This include DMNeural of unknown levels (if specified) and DMNeural of specific levels. If no DMNeural values were generated, the SAS Code menu item is dimmed and unavailable.
  - **PMML Code** — the DMNeural node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — displays the following fit statistics:
    - _ERR_ — Error Function
    - _SSE_ — Sum of Squared Errors
    - _MAX_ — Maximum Absolute Error
    - _DIV_ — Divisor for ASE
    - _NOBS_ — Sum of Frequencies
    - _WRONG_ — Number of Wrong Classifications
    - _DISF_ — Frequency of Classified Cases
    - _MISC_ — Misclassification Rate
    - _ASE_ — Average Squared Error
    - _RASE_ — Root Average Squared Error
    - _AVERR_ — Average Error Function
    - _DFT_ — Total Degrees of Freedom
    - _DFM_ — Model Degrees of Freedom
    - _DFE_ — Degrees of Freedom for Error

    - _MSE_ — Mean Squared Errors
    - _RMSE_ — Root Mean Squared Errors
    - _NW_ — Number of Weight
    - _FPE_ — Final Prediction Error
    - _RFPE_ — Root Final Prediction Error
    - _AIC_ — Akaike's Information Criterion
    - _SBC_ — Schwarz Bayesian Criterion

  - **Classification Chart** — displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into.

  - **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles and observations in a decile are used to calculate the statistics that are plotted in deciles charts. The Score Rankings Overlay plot displays both train and validate statistics on the same axis.

    By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any):

    - posterior probability of target event
    - number of events
    - cumulative and noncumulative lift values
    - cumulative and noncumulative % response
    - cumulative and noncumulative % captured response
    - gain
    - actual profit or loss
    - expected profit or loss

  - **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used. For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores.

- **Model**

  - **Stagewise Optimization Statistics** — In each stage of the DMNeural training process, the training data set is fitted with eight separate activation functions. The Stagewise Optimization Statistics window displays a lattice of plots that show model fit statistics of different activation functions at each stage of the training process. The fit statistics that are displayed in the lattice graph are Sum of Squared Errors and Accuracy Percent.

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph wizard to modify an existing results plot or to create a results plot of your own.

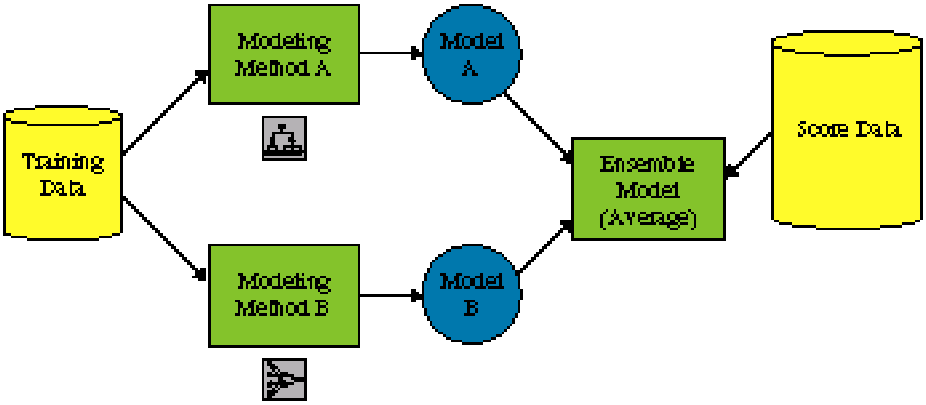
Chapter 52
Ensemble Node

Ensemble Node

Overview of the Ensemble Node

The Ensemble node creates new models by combining the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple predecessor models. The new model is then used to score new data. The SAS Enterprise Miner Ensemble node supports group processing options of index, stratify (looping over variables), cross-validation, target, bagging, and boosting.

One common ensemble approach is to use multiple modeling methods, such as a neural network and a decision tree, to obtain separate models from the same training data set. The component models from the two complementary modeling methods are integrated by the Ensemble node to form the final model solution.



It is important to note that the ensemble model can only be more accurate than the individual models if the individual models disagree with one another. You should always compare the model performance of the ensemble model with the individual models. You can compare models in a Model Comparison node.

The Ensemble node must be placed in a process flow diagram after a modeling node. The Ensemble node does not have to be the immediate successor node to one of the modeling nodes; for example, you might follow a modeling node with the MultiPlot node and then the Ensemble node with no irregularities.

When you run a process flow diagram with multiple models connected to an Ensemble node, only the ensemble model results are assessed. You can create assessment charts, such as lift and profit charts in a Model Comparison node. Because the ensemble model is represented in SAS DATA step format instead of PROC format, some fit statistics that are usually available in PROC output, such as the root mean square error, are not calculated or displayed in the Results window.

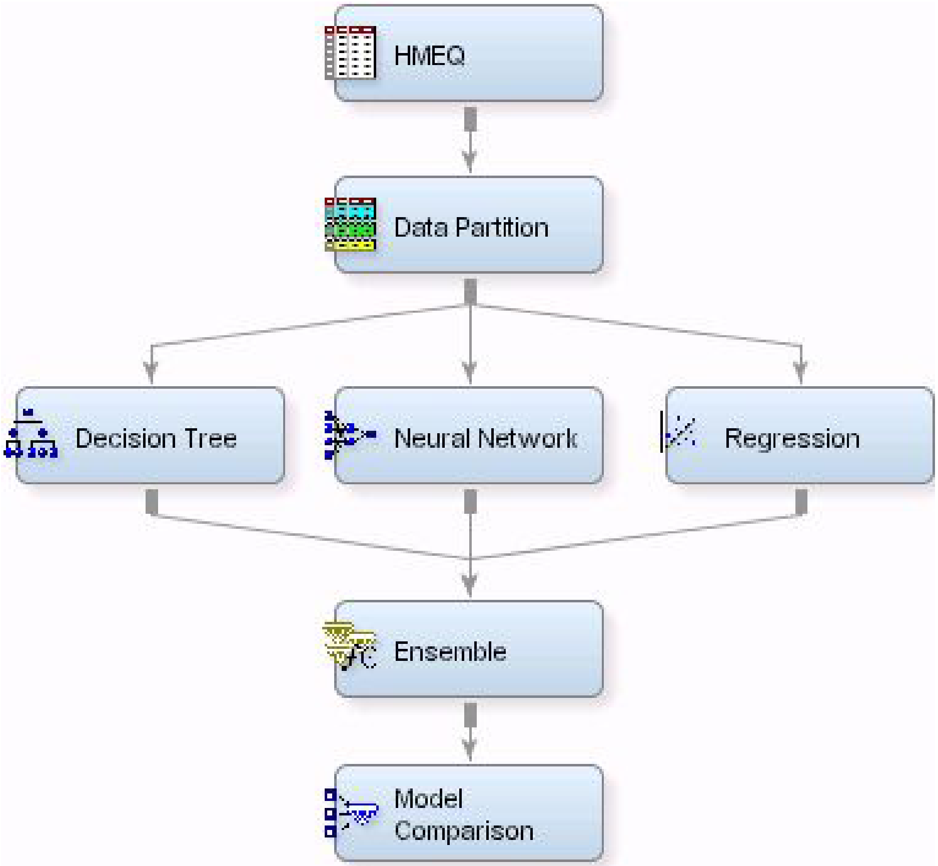
Note: In SAS Enterprise Miner 12.3, the Ensemble node only supports the modeling nodes that generate the score code in DATA step format. For example, the MBR (Memory-Based Reasoning) node is not supported.

See the Predictive Modeling section for information that applies to all of the predictive modeling nodes.

Combing Models Using the Ensemble Node

Combined models are created using the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple models. You may want to create a combined model to improve the stability of disparate nonlinear models, such as those created from the Neural Network and Decision Tree nodes.

The following display shows an example flow which combines decision tree and neural network models:



In this example flow, each modeling node creates a separate model using the same training data. The Ensemble node co-mines the posterior probabilities (for class targets) or the predicted values (for interval targets) from the component models to create the ensemble model. The ensemble model is then stored as a single model entry in the Model Comparison node.

The Ensemble node can read only one model from a given modeling node. In order to combine two models that were created from the same modeling method (for example, two different decision tree models), you must define two separate modeling nodes in the diagram workspace.

Combination Methods in the Ensemble Node

The Ensemble node use the following methods to combine results from different modeling nodes. You use the Predicted Values, Posterior Probabilities, and Voting Posterior Probabilities properties to specify the method.

- **Average** — takes the average of the posterior probabilities (for categorical targets) regardless the target event level, or of the predicted values (for interval targets) from different models as the prediction from the Ensemble node.
- **Maximum** — takes the maximum of the posterior probabilities (for categorical targets) or of the predicted values (for interval targets) from different models as the prediction from the Ensemble node.
- **Voting** — This method is available for categorical targets only. When you use the voting method to compute the posterior probabilities, two methods are available for voting the posterior probabilities: Average and Proportion.

The Average method for voting posterior probabilities uses the posterior probabilities from the models that predict the same target event. For example, if models M1, M2, M3 predict the event level J1, and model M4 predicts the event level J2, the posterior probability for J1 in the Ensemble node would be computed by averaging the posterior probabilities of J1 from models M1, M2, and M3. Model M4 is ignored.

The Proportion method for voting posterior probabilities ignores the posterior probabilities that are generated from the individual models. Instead, the Proportion method computes the posterior probability for a target value J1 based on the proportion of individual models that predict the same event level. For example, if individual models M1, M2, and M3 predict the target value J1, and model M4 predicts the target value J2, the posterior probability for J1 in the Ensemble node would be 3/4.

Note: If only one model is being ensembled, then no model combination is performed.

Ensemble Node Data Set Requirements

The Ensemble node supports only one target variable. The target variable must be modeled by a predecessor modeling node.

Ensemble Node Properties

Ensemble Node General Properties

The following general properties are associated with the Ensemble node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Ensemble node added to a diagram will have a Node ID of Ensembl. The second Ensemble node added to a diagram will have a Node ID of Ensembl2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — Ensemble window. The Imported Data — Ensemble window contains a lists of the ports that provide data sources to the Ensemble node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — the Exported Data property provides access to the Exported Data — Ensemble window. The Exported Data — Ensemble window contains a list of the output data ports that the Ensemble node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Ensemble Node Train Properties

The following train properties are associated with the Ensemble node:

- **Variables** — specifies the properties of each variable in the data source that you want to use. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can set a variable's report status to **Yes** or **No**.

Ensemble Node Train Properties: Interval Target

- **Predicted Values** — Use the Predicted Values property of the Ensemble node to specify the function that you want to use to combine models for interval targets. The function choices are **Average** and **Maximum**. The default setting is **Average**.

Ensemble Node Train Properties: Class Target

- **Posterior Probabilities** — Use the Posterior Probabilities property of the Ensemble node to specify the method that you want to use to combine probabilities for class targets. The choices are **Average**, **Maximum**, and **Voting**. The default setting is **Average**.

- **Posterior Probabilities for Voting** — Use the Posterior Probabilities for Voting property of the Ensemble node to specify the method that you want to use to compute the posterior probabilities when the function to combine models is voting, with class targets (binary, ordinal, or nominal). The method choices are **Proportion** and **Average**. The default setting is **Average**.

Ensemble Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.

- **Last Error** — displays the error message from the last run.

- **Last Status** — displays the last reported status of the node.

- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.

- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Ensemble Node Results

Ensemble Results Menu

You can open the Results window of the Ensemble node by right-clicking the node a and selecting **Results**. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following in the Results window.

- **Properties**
  - **Settings** — displays a window with a read-only table of the Ensemble node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Ensemble node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Ensemble run.
  - **Output** — the SAS output of the Ensemble run. The Ensemble SAS Output includes a variable summary, a model events summary, a decision matrix, a table of predicted and decision variables, a list of imported models to be combined, fit statistics for train and validate data sets, classification tables for train and validate data sets, decision tables for train and validate data sets, an event classification table, assessment score rankings for train, validate, and test data, and assessment score distributions.
  - **Flow Code** — the SAS code used to produce the output that the Ensemble node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Ensemble node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — The Fit Statistics Table in the Ensemble Results window displays the following statistics for the train, validate, and test data sets (when applicable).
    - _ASE_ — Average Squared Error
    - _AVERR_ — Average Error Function
    - _DISF_ — Frequency of Classified Cases
    - _DIV_ — Divisor for _ASE_
    - _ERR_ — Error Function
    - _MAX_ — Maximum Absolute Error

  - _MISC_ — Misclassification Rate
  - _NOBS_ — Sum of Frequencies
  - _RASE_ — Root Average Squared Error
  - _SSE_ — Sum of Squared Errors
  - _WRONG_ — Number of Wrong Classifications

- **Classification Chart** — The Classification Table chart displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represent the percentage of total observations.

- **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

  By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations.

  The vertical axis displays the following values, and their mean, minimum, and maximum (if any):

  - posterior probability of target event
  - number of events
  - cumulative and noncumulative lift values
  - cumulative and noncumulative % response
  - cumulative and noncumulative % captured response
  - gain
  - actual profit or loss
  - expected profit or loss

- **Score Rankings Matrix** — The score ranking matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. Plots can also be created for report variables.

  Y-axis plots that are available in the Score Rankings Matrix Chart are as follows:

  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response

- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. For interval targets, observations are grouped into bins, based on the actual predicted values of the target.

The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores.

You can select the score distribution statistic that you want to chart. The default chart choice is Percentage of Events. Other chart choices are

- Percentage of Events
- Number of Events
- Cumulative Percentage of Events

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Modifying Ensemble Results Plots and Tables

You can modify and enhance Results graph attributes, whether you want to graphically explore effects of different variable roles, add or change response variables, or modify graph attributes to emphasize particular results or enhance appearance for presentation purposes.

For more information on manipulating plot variable roles and response variables, see the section in the Enterprise Miner User Interface Help on the Data Options Dialog window. For more information on modifying individual plot attributes, see the section on the Graph Properties window.

Chapter 53
Gradient Boosting Node

Gradient Boosting Node

Overview of the Gradient Boosting Node

The Gradient Boosting Node is on the Model tab of the Enterprise Miner tools bar. This node uses a partitioning algorithm described in "A Gradient Boosting Machine," and "Stochastic Gradient Boosting" by Jerome Friedman. A partitioning algorithm is one that searches for an optimal partition of the data defined in terms of the values of a single variable. The optimality criterion depends on how another variable, the target, is distributed into the partition segments. The more similar the target values are within the segments, the greater the worth of the partition. Most partitioning algorithms further partition each segment in a process called, recursive partitioning. The partitions are then combined to create a predictive model. The model is evaluated by goodness-of-fit statistics defined in terms of the target variable. These statistics are different than the measure of worth of an individual partition. A good model may result from many mediocre partitions.

Gradient boosting is a boosting approach that resamples the analysis data set several times to generate results that form a weighted average of the re-sampled data set. Tree boosting creates a series of decision trees which together form a single predictive model. A tree in the series is fit to the residual of the prediction from the earlier trees in the series. The residual is defined in terms of the derivative of a loss function. For squared error loss with an interval target the residual is simply the target value minus the predicted value. Each time the data is used to grow a tree and the accuracy of the tree is computed. The successive samples are adjusted to accommodate previously computed inaccuracies. Because each successive sample is weighted according to the classification accuracy of previous models, this approach is sometimes called stochastic gradient boosting. Boosting is defined for binary, nominal, and interval targets.

Like decision trees, boosting makes no assumptions about the distribution of the data. For an interval input, the model only depends on the ranks of the values. For an interval target, the influence of an extreme value theory depends on the loss function. The Gradient Boosting node offers a Huber M-estimate loss which reduces the influence of extreme target values. Boosting is less prone to overfit the data than a single decision tree, and if a decision tree fits the data fairly well, then boosting often improves the fit.

See the “Predictive Modeling” on page 149 section for information that applies to all of the predictive modeling nodes.

Gradient Boosting Algorithm

The algorithm behind the Gradient Boosting Node uses algorithms described in "A Gradient Boosting Machine," and "Stochastic Gradient Boosting" by Jerome Friedman.

Gradient Boosting Node Properties

Gradient Boosting Node General Properties

The following general properties are associated with the Gradient Bossting node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Gradient Boosting node that is added to a diagram will have a Node ID of Boost. The second Gradient Boosting node added to a diagram will have a Node ID of Boost2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Gradient Boosting window. The Imported Data — Gradient Boosting window contains a list of the ports that provide data sources to the Gradient Boosting node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Gradient Boosting window. The Exported Data — Gradient Boosting window contains a list of the output data ports that the Gradient Boosting node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.

  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Gradient Boosting Node Train Properties

The following train properties are associated with the Gradient Boosting node:

- **Variables** — Select the ... button to open the Variables - Gradient Boosting table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use and Report variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - Apply — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

Gradient Boosting Node Train Properties: Series Options

- **N Iterations** — Use the N Iterations property to specify the number of terms in the boosting series. For interval and binary targets, the number of iterations equals the trees. For a nominal target, a separate tree is created for each target category in each iteration series.
- **Seed** — Use the Seed property to specify the seed for generating random numbers. The Training Proportion property uses this value to select a training sample at each iteration.
- **Shrinkage** — Use the Shrinkage property to specify how much to reduce the prediction of each tree.
- **Train Proportion** — Use the Train Proportion property to specify the proportion of training observations to train a tree with. A different training sample is taken in each iteration. Trees trained in the same iteration have the same training data.

Gradient Boosting Node Train Properties: Splitting Rule

- **Huber M-Regression** — Use the Huber M-regression loss function instead of square error loss with an interval target. The Huber loss function is less sensitive to extreme target values. The Huber threshold value must be between 0.6 and 1. A value close to one, such as 0.9, is reasonable. HUBER=NO requests using square error loss instead of Huber loss. The default value is No. The HUBER option is ignored unless the target has interval measurement level. See Friedman, (1999), p. 1197 for more details on the Huber M-regression loss function.

- **Maximum Branch** — Use the Maximum Branch property to specify the maximum number of branches that you want a splitting rule to produce from one node. This property restricts the number of subsets that a splitting rule can produce to the specified number or fewer. Permissible values for the Maximum Branch property are integers between 2 and 100. The minimum value of 2 results in binary trees. The default value for the Maximum Branch property is 2.
- **Maximum Depth** — Use the Maximum Depth property to specify the maximum depth of a node that will be created. The depth of a node equals the number of splitting rules needed to define the node. The root node has depth zero. The children of the root node are the first generation and have depth one.
- **Minimum Categorical Size** — Use the Minimum Categorical Size property to specify the minimum number of training observations that a categorical value must have before the category can be used in a split search. Categorical values that appear in fewer than the number specified (n) are regarded as if they were missing. If the Missing Values property is set to Use in Search, the categories occurring in fewer than n observations are merged into the pseudo category for missing values for the purpose of finding a split. Otherwise observations with infrequent categories are excluded from the split search. The policy for assigning such observations to a branch is the same as the policy for assigning missing values to a branch. Permissible values are integers greater than or equal to 1. The default value for the Minimum Categorical Size property is 5.
- **Re-use Variable** — Use the Re-use Variable property to specify how many splitting rules in a path may use the same variable.

  When a splitting rule is considered for a node, the worth of the rule is multiplied by n if the splitting variable has already been used in a primary splitting rule in an ancestor node. For n > 1, once a variable is used in a primary split, it is more likely to appear in a rule in a descendent node because it gets an advantage competing for splitting in the descendent node. The tree ends up using fewer variables, especially correlated variables. Reducing the number of variables in a tree may make the tree easier to interpret or to deploy. Eliminating correlated variables in the tree results in a more accurate measure of importance of those that are in the tree.

  An appropriate value for n depends on context. A value of two or three might be reasonable. The default value for n is one, giving no advantage or disadvantage for reusing a variable in a path. Set n = 0 to avoid using the same variable more than once in a path.
- **Categorical Bins** — Use the Categorical Bins property to specify the number of preliminary bins to collect categorical input values when preparing to search for a split. If an input variable has more than n categories in the node, then the split search uses the most frequent n − 1 categories, and regards the remaining categories as a single pseudo category. The count of categories is done separately in each node. The default value is 30.
- **Interval Bins** — Use the Interval Bins property to specify the number of preliminary bins to consolidate interval input values into. The width equals (max(x) − min(x))/n, where max(x) and min(x) are the maximum and minimum of the input variable values in the training data in the tree node being searched. The width is computed separately for each input and each node. The search algorithm ignores the Interval Bins property if the number of distinct input values in the node smaller than n. The default value of n is 100.

- **Missing Values** — Use the Missing Values property to specify how splitting rules handle observations that contain missing values for a variable. If a surrogate rule can assign an observation to a branch, then it does, and the missing value policy is ignored for the specific observation.

  The default value is Use in search. Select from the following available missing value policies:

  - **Use in search** — uses missing values during the split search.
  - **Largest branch** — assigns the observations that contain missing values to the branch with the largest number of training observations.
  - **Most correlated branch** — assigns an observation with missing values to the most correlated branch.
- **Performance** — Use the Performance property to specify where to put the working copy of the training data. This property affects the speed of computations with no impact on the results.
  - **Disk** — Requests the working copy of the training data to be stored in a disk utility file. Storing the copy on disk may free a considerable amount of memory for calculations, possibly shortening the execution time.
  - **RAM** — Requests the working copy of the training data to be stored in memory if enough memory is available for it and a minimum number of calculations.

Gradient Boosting Node Train Properties: Node

- **Leaf Fraction** — Use the Leaf Fraction property to specify the smallest number of training observations a new branch may have, expressed as the proportion of the number N of available training observations in the data. N may be less than the total number of observations in the data set because observations with a missing target value are excluded. The default is 0.1.
- **Number of Surrogate Rules** — Use the Number of Surrogate Rules property to specify the maximum number of surrogate rules that Gradient Boosting node seeks in each non-leaf node. The first surrogate rule is used when the main splitting rule relies on an input whose value is missing. If the first surrogate also relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main and all of the surrogate's rules from applying to an observation, then the main rule assigns the observation to the branch it has designated as receiving missing values. Permissible values are nonnegative integers. The default value for the Number of Surrogate Rules property is 0.
- **Split Size** — Use the Split Size property to specify the smallest number of training observations that a node must have before it is eligible to be split. Permissible values are integers greater than or equal to 2. The default value is determined based on the data.

Gradient Boosting Node Train Properties: Split Search

- **Exhaustive** — Use the Exhaustive property to specify the highest number of candidate splits that you want to find in an exhaustive search. If more candidates should be considered, a heuristic search is used instead. The exhaustive method of searching for a split examines all possible splits. If the number of possible splits is greater than n, then a heuristic search is done instead of an exhaustive search. The exhaustive and heuristic search methods only apply to multi-way splits, and to binary splits on nominal targets with more than two values. The Exhaustive property applies to multi-way splits and to binary splits on nominal targets with more than two values.

Permissible values are integers between 0 and 2,000,000,000. The default setting for the Exhaustive property is 5000.

- **Node Sample** — Use the Node Sample property to specify the maximum within-node sample size n that you want to use to find splits. If the number of training observations in a node is larger than n, then the split search for that node is based on a random sample of size n. Permissible values are integers greater than or equal to 2. The default value for the Node Sample property is 20000.

Gradient Boosting Node Train Properties: Subtree

- **Assessment Measure** — Use the Assessment Measure property to specify the method that you want to use to select the best tree, based on the validation data. If no validation data is available, training data is used. The available assessment measurements are
  - **Decision** (default setting) — The Decision method selects the tree that has the largest average profit and smallest average loss if a profit or loss matrix is defined. If no profit or loss matrix is defined, the value of model assessment measure will be reset in the training process, depending on the measurement level of the target. If the target is ordinal, the measure is set to Decision. If the target is interval, the measure is set to Average Square Error. If the target is categorical, the measure is set to Misclassification.
  - **Average Square Error** — The Average Square Error method selects the tree that has the smallest average square error.
  - **Misclassification** — The Misclassification method selects the tree that has the smallest misclassification rate.

Gradient Boosting Node Score Properties

The following score properties are associated with the Gradient Boosting node:

- **Subseries** — Use the Subseries property to specify how many iterations in the series to use in the final model. For a binary or interval target, the number of iterations is the number of trees. For a nominal target with k categories, k > 2, each iteration contains k trees.
  - **Best Assessment Value** — The Best Selection Value selects the smallest subseries with the best assessment value.
  - **Complete Series** — The Complete Series selects the complete or longest series.
  - **N Iterations** — The N Iterations selects the subseries containing N iterations.
- **Number of Iterations** — Specifies the specific number of iterations to be used in the final model. This value is used when the Subseries property is set to the N Iterations setting.
- **Create H Statistic** — Specifies whether the H statistic is calculated for every primary splitting variable. The H statistic indicates interaction importance for all variables that are identified as primary splitting variables during the boosting procedure. With many splitting variables, the matrix of variable interactions can become very large. Calculating the H statistic to measure interaction importance among all splitting variables can be extremely time-consuming. For this reason, the default setting of this property is No.
- **Variable Selection** — Use the Variable Selection property to specify whether variable selection should be performed based on importance values. If the Variable Selection property is set to Yes, all variables that have an importance value greater

than or equal to 0.05 will have the variable role set to Input. All other variables will be set to Rejected. The default setting for the Variable Selection property is Yes.

Gradient Boosting Node Report Properties

The following report properties are associated with the Gradient Boosting node:

- **Observation Based Importance** — Use the Observation Based Importance property to specify whether observation-based importance statistics should be generated.
- **Number Single Var Importance** — Use the Number Single Var Importance property to specify the number of variables that one way importance statistics should be generated.

Gradient Boosting Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Gradient Boosting Node Results

You can open the Results window of the Gradient Boosting Node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Gradient Boosting Node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Gradient Boosting Node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Gradient Boosting Node . The table includes columns to see the variable name, the variable role, the variable level, and the model used. (Name, Use, Report, Role, and Level).
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**

  - **Log** — the SAS log of the Gradient Boosting Node run.
  - **Output** — the SAS output of the Gradient Boosting Node run.
  - **Flow Code** — the SAS code used to produce the output that the Gradient Boosting Node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Gradient Boosting Node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — a table of the fit statistics from the model.
  - **Classification Chart: <TARGET>** — a bar chart that shows the correct classification of the values of the target variable.
  - **Score Rankings Overlay: <TARGET>** — opens the Score Rankings Overlay chart.
  - **Score Distribution: <TARGET>** — opens the Score Distribution chart.
- **Model**
  - **Subseries Plot** — opens the Subseries Plot.
  - **Variable Importance** — opens the Variable Importance table.
  - **Observation Based Importance Statistics** — opens the Observation Based Importance Statistics table. This menu item is available when the Observation Based Importance property is set to Yes.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Select a Chart Type wizard to modify an existing Results plot or create a Results plot of your own.

Gradient Boosting Node Details

Missing Values

See the Missing Values on page 766 section of the Decision Tree documentation for details about the recursive partitioning computation.

Variable Importance

The Gradient Boosting node provides two approaches to evaluating the importance of a variable: split-based and observation-based. The split-based approach uses the reduction in the sum of squares from splitting a node, summing over all nodes. The observation-based approach uses the increase in a fit statistic due to making the observation values of a variable uninformative. A detailed explanation of each approach is given in separate sections below. Measures of variable importance generally underestimate the importance of correlated variables. Two correlated variables could make a similar contribution to a model. The total contribution is usually divided between them, and neither variable acquires the rank it deserves. Eliminating either variable generally increases the contribution attributed to the other.

Split-Based Variable Importance

The split-based approach to variable importance fully credits both correlated variables when using surrogate rules. When the primary splitting rule uses one of two highly correlated variables, a surrogate splitting rule will use the other variable. Both variables get about the same credit for the reduction in residual sums of squares in the split of that node. The overall importance of correlated variables are about the same and in the correct relation to other variables.

The observation-based variable importance is misleading when some variables are correlated. Consider using the split-based importance with surrogates first to discover superfluous correlated variables and eliminating them before relying on observation-based importance.

The split-based approach to evaluating the importance of a variable utilizes the idea that the reduction in the sum of squares due to the model may be expressed as a sum over all nodes of the reduction in the sum of squares due to splitting the data in the node. The credit for the reduction in a particular node goes to the variable used to split the node. (More than one variable gets credit when surrogate rules exist.) The formula for variable importance sums the credits over all splitting rules, scales the sums so that the largest sum is one, and takes a square-root to revert to linear units.

The split-based approach uses statistics already saved in a node and does not need to read the data again. The relative importance computed with the training data and again computed with the validation data. If the validation data indicates a much lower importance than the training data, then the variable is over-fitting. The over-fitting usually occurs in an individual node that uses the variable to split with. The validation statistics in the branches will differ substantially from the training data statistics in such a node.

See the “Variable Importance” on page 808 section of the Decision Tree documentation for details about the recursive partitioning computation.

Observation-Based Variable Importance

The observation-based approach to evaluating the importance of a variable entails reading a data set and applying the model several times to each observation, first to compute the standard prediction of the observation, and then once for each variable or pairs of variables being evaluated to compute a prediction in which variables being evaluated are made uninformative. The difference between the standard prediction and the prediction using an uninformative rendering of a variable (or pair of variables) is a measure of the influence of the variable (or pair of variables). A plot of all observations of the differences versus the value of the variable can show which values influence the prediction the most. The difference in a fit statistic computed first the standard way and then computed using the uninformative rendering of the variable is a measure of the overall importance of the variable.

To make a variable uninformative the Gradient Boosting node replaces its value in a given observation with the empirical distribution of the variable, and replaces the standard prediction with the expected prediction integrated over the distribution. In simpler words, to make a variable uninformative imagine making several copies of a given observation, altering the values of the variable being evaluated, and then taking the average of the standard predictions of these copies. The choice of altered values follows the empirical distribution: each value that appears in the input data set also appears among the imaginary copies of the observation, and appears with the same frequency. Notice that the uninformative prediction of an observation depends on the other observations in the input data set because of the empirical distribution.

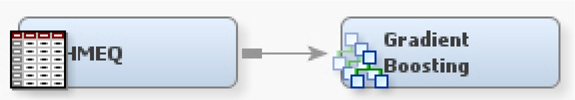
Gradient Boosting Node Example

Use the Enterprise Miner Data Source Wizard to create the Home Equity data source. The example data is located in the SAS sample library SAMPSIO. Use the SAS sample Home Equity data set, SAMPSIO.HMEQ.

Configure the variables in the SAMPSIO.HMEQ data set as follows:

- Set the role of the variable BAD as the Target variable.
- Set the level of the variable BAD to Binary.
- Save the SAMPSIO.HMEQ data set using the role of either Train or Raw.

Create a new process flow diagram, then drag the Home Equity (HMEQ) data source onto the diagram workspace. Drag a Gradient Boosting node from the Model tab of the node toolbar and connect this node to the HMEQ data source node. Select the Gradient Boosting node and set the Observation Based Importance property to Yes. Use the default property settings for the other properties.

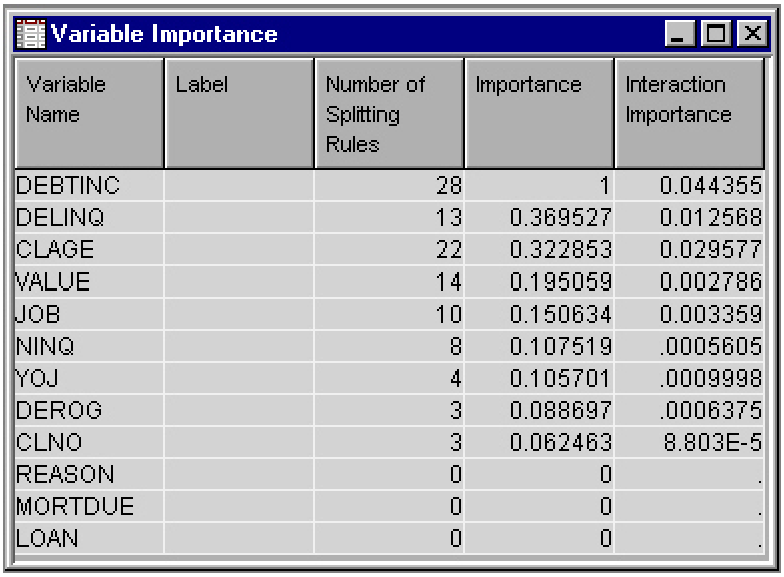


Right-click the Gradient Boosting node, then select **Run** to run the process flow diagram. When the Run Status window indicates that the run is completed, select the **Results** button to open the Gradient Boosting Results window.

Examine the Variable Importance table. The table contains the split-based measure of relative importance of each input variable in the selected subtree.

The table contains the following variables:

- Variable Name — the input variable name.
- Label — the input variable label.
- Number of Splitting Rules — the number of splitting rules using this variable.
- Importance — the relative importance computed with the training data. This measures the relative influence of individual input variables on the variation of $\hat{F}(x)$ over the joint input variable distribution. See Friedman (1999), pp. 1217-1219 for more details about relative importance. In the example, DEBTINC, DELINQ and CLAGE have the top 3 highest relative importance values.
- Interaction Importance — the interaction importance for each variable.



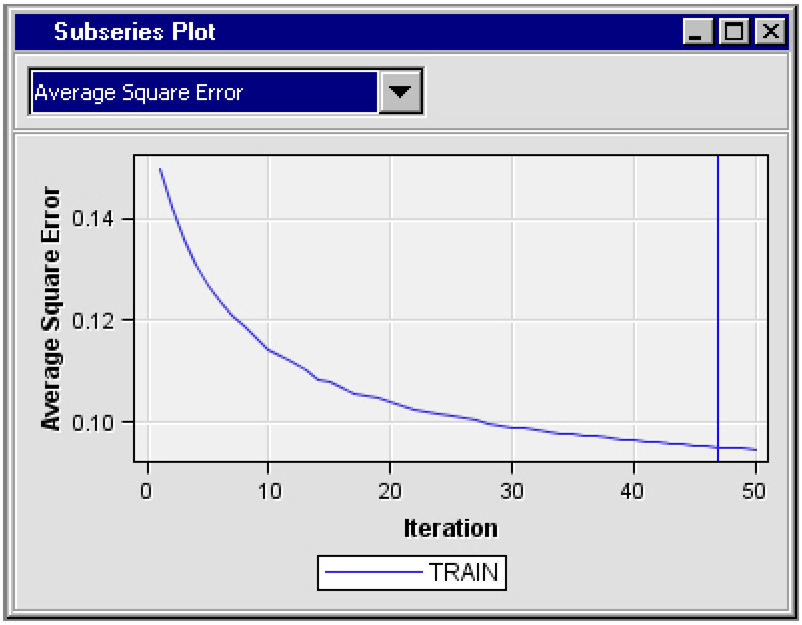
Variable Importance

Variable Name	Label	Number of Splitting Rules	Importance	Interaction Importance
DEBTINC		28	1	0.044355
DELINQ		13	0.369527	0.012568
CLAGE		22	0.322853	0.029577
VALUE		14	0.195059	0.002786
JOB		10	0.150634	0.003359
NINQ		8	0.107519	.0005605
YOJ		4	0.105701	.0009998
DEROG		3	0.088697	.0006375
CLNO		3	0.062463	8.803E-5
REASON		0	0	.
MORTDUE		0	0	.
LOAN		0	0	.

Examine the Subseries Plot.

The following plots against Iteration are available:

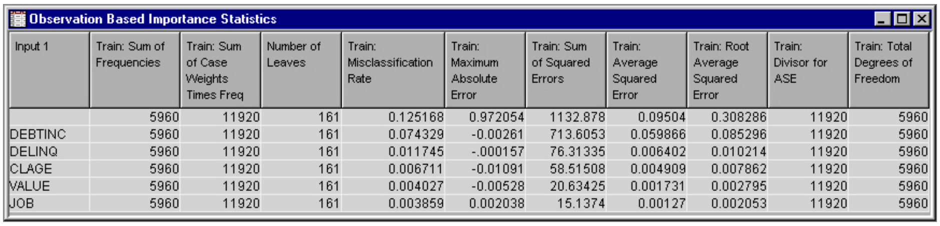
- Average Square Error
- Number of Leaves
- Misclassification Rate
- Maximum Absolute Error
- Sum of Squared Errors
- Average Square Error
- Root Average Squared Error



Examine the Observation Based Importance table by selecting **View** ⇨ **Model** ⇨ **Observation Based Importance Statistics**

The table contains the following variables:

- Input1
- Sum of Frequencies
- Sum of Case Weights Times Freq
- Number of Leaves
- Misclassification Rate
- Maximum Absolute Error
- Sum of Squared Errors
- Average Squared Error
- Root Average Squared Error
- Divisor for ASE
- Total Degrees of Freedom



Observation Based Importance Statistics

Input 1	Train: Sum of Frequencies	Train: Sum of Case Weights Times Freq	Number of Leaves	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error	Train: Divisor for ASE	Train: Total Degrees of Freedom
	5960	11920	161	0.125168	0.972054	1132.878	0.09504	0.308286	11920	5960
DEBTINC	5960	11920	161	0.074329	-0.00261	713.6053	0.059866	0.085296	11920	5960
DELINQ	5960	11920	161	0.011745	-.000157	76.31335	0.006402	0.010214	11920	5960
CLAGE	5960	11920	161	0.006711	-0.01091	58.51508	0.004909	0.007862	11920	5960
VALUE	5960	11920	161	0.004027	-0.00528	20.63425	0.001731	0.002795	11920	5960
JOB	5960	11920	161	0.003859	0.002038	15.1374	0.00127	0.002053	11920	5960

References

Friedman, Jerome H “Greedy Function Approximation: A Gradient Boosting Machine.” 2001. *The Annals of Statistics* 29: 1189–1232.

Friedman, Jerome H “Stochastic Gradient Boosting.” 2002. *Computations Statistics & Data Analysis* 38: 367–378.

Barry de Ville 2006. *Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner*. Cary, NC: .

Chapter 54
Interactive Decision Tree Application

Interactive Decision Tree Application

Overview of the Interactive Decision Tree Application

The Tree Desktop Application that was included with previous versions of SAS Enterprise Miner was replaced in SAS Enterprise Miner 7.1 by the Interactive Decision Tree application. The Interactive Decision Tree application is integrated with the SAS Enterprise Miner user interface, requires no additional installation, and is available through the Decision Tree modeling node. Users of SAS Enterprise Miner 7.1 and all later versions who start the software using Java Web Start have the full use of the Interactive Decision Tree application. A switch targets feature allows users to select a new dependent variable in a tree leaf and make new splits based on the new target. This feature can be handy when designing decision trees for segmentation strategies.

Interactive Decision Tree and Target Variables with Missing Values

The SAS Enterprise Miner **Decision Tree** node uses complete case analysis with respect to target variables. Complete case analysis means that SAS Enterprise Miner only uses observations that contain target variable values for model building. Observations that contain missing values for target variables are excluded from the analysis. In cases where the training data contains many missing values for various target levels, it is possible for complete case analysis to exclude so many observations (due to missing target levels) that only one target level can be modeled. In such cases, consider cleansing the training data, or imputing or replacing the missing target level values.

Interactive Decision Tree and Sampling

When you create a standard SAS Enterprise Miner decision tree, the decision tree will form splits based on all of the input data that is allocated for the node to use. In contrast, when you perform interactive decision tree training, you might potentially be training the interactive decision tree using a sample of the input data. SAS Enterprise Miner determines whether to sample interactive decision tree data based on the number of rows and columns in the input data source's training data. The SAS Enterprise Miner interactive decision tree data sampling algorithm performs a random sample of the input training and validation data by default, is automatic, and does not require user input for activation.

However, some users might wish to override default SAS Enterprise Miner interactive decision tree sampling strategies. SAS Enterprise Miner provides two macros that you can issue with your project start code that will modify interactive decision tree input data sampling behaviors:

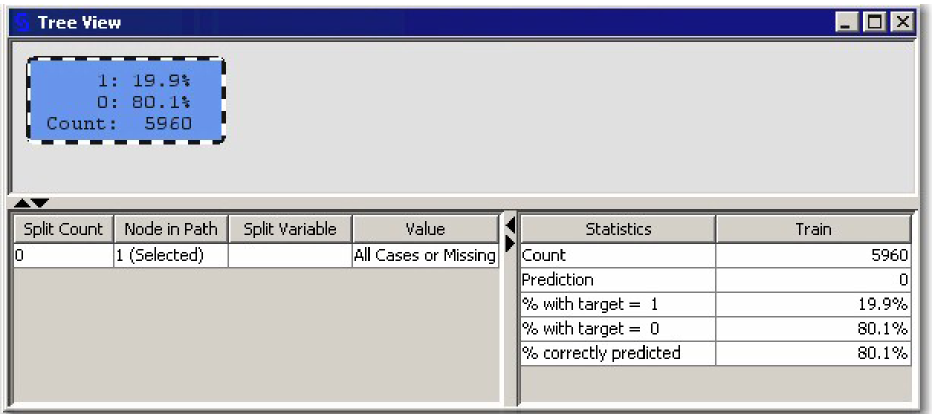
```
%let EM_INTERACTIVE_TREE_MAXOBS= <max-number-of-observations-in-sample>;

%let EM_INTERACTIVE_TREE_SAMPLEMETHOD=<RANDOM | FIRSTN | STRATIFY>;
```

The first macro specifies the maximum number of observations that can exist in an Interactive Decision Tree node sample. You use this macro if you want to manually control the sample size. Otherwise, SAS Enterprise Miner will use its own algorithms to perform sampling for your interactive decision tree.

The second macro specifies the sampling methodology that will be used to create an Interactive Decision Tree node sample. You can use this macro if you want to manually control the methodology that SAS Enterprise Miner uses to create interactive decision tree samples. By default, SAS Enterprise Miner uses random sampling for interactive decision trees. You can use the macro to choose between RANDOM, STRATIFY, and FIRSTN sample creation. You use the EM_INTERACTIVE_TREE_MAXOBS macro to specify the number of observations for any of the sampling strategies.

Launching the Interactive Decision Tree Application

To launch an interactive training session in SAS Enterprise Miner, click the ... button at the right of the Decision Tree node's Interactive property in the Properties panel. By default, the Interactive Decision Tree window displays a Tree View and a split pane to help identify information and statistics about the highlighted node. The **Rules Pane** on the left displays the split count from the root node, the node ID for the current and each predecessor node, the variable used for this particular splitting decision, and what values or criteria were used in the split. The **Statistics Pane** on the right displays summary statistics related to the training at each node. The panes can be hidden entirely by clicking the black triangle that is pointing down on the horizontal divider. To hide the **Rules Pane**, click the black triangle that is pointing to the left on the vertical divider. To hide the **Statistics Pane**, click the black triangle that is pointing to the right on the vertical divider. Unless you have previously run your process flow diagram and automatically generated a decision tree, the Tree View will contain only the root node as depicted below:



```
1: 19.9%
0: 80.1%
Count:  5960
```

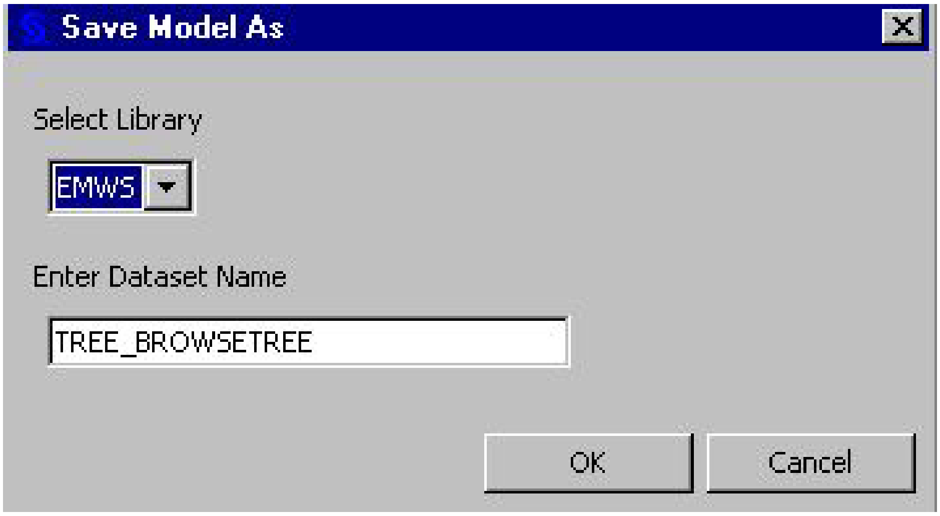
Split Count	Node in Path	Split Variable	Value
0	1 (Selected)		All Cases or Missing

Statistics	Train
Count	5960
Prediction	0
% with target = 1	19.9%
% with target = 0	80.1%
% correctly predicted	80.1%

Interactive Decision Tree Application Menus

The Interactive Decision Tree window provides a main menu with the following menu items:

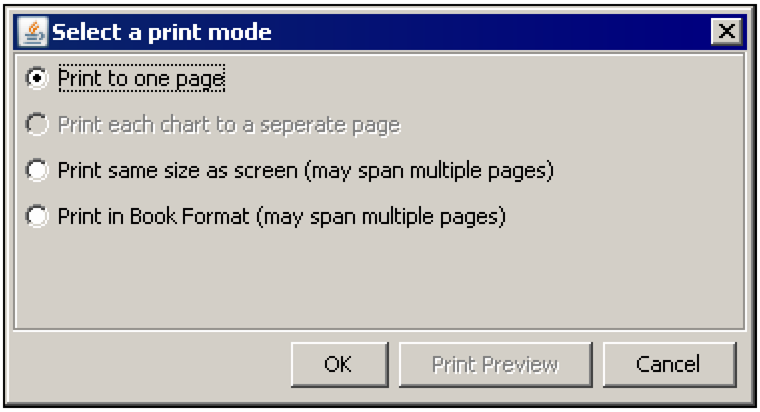
- **File**
  - **Save** — saves the decision tree to a file that is designated in the **Properties** menu item.
  - **Save As** — opens a dialog box that enables you to select a library and a data set name and then saves the decision tree data using the information provided.



Select Library: EMWS

Enter Dataset Name: TREE_BROWSETREE

  - **Save View Settings as Default** — enables you to change which views are displayed by default when you open the Interactive Decision Tree window. The initial default settings are for the Tree View and the Tree Map to be displayed.
  - **Reset Tree Data Model** — reverts the decision tree to the last saved version.
  - **Print** — prints the contents of the Tree View. The Select a print mode dialog box enables you to choose from four printing modes:


Select a print mode

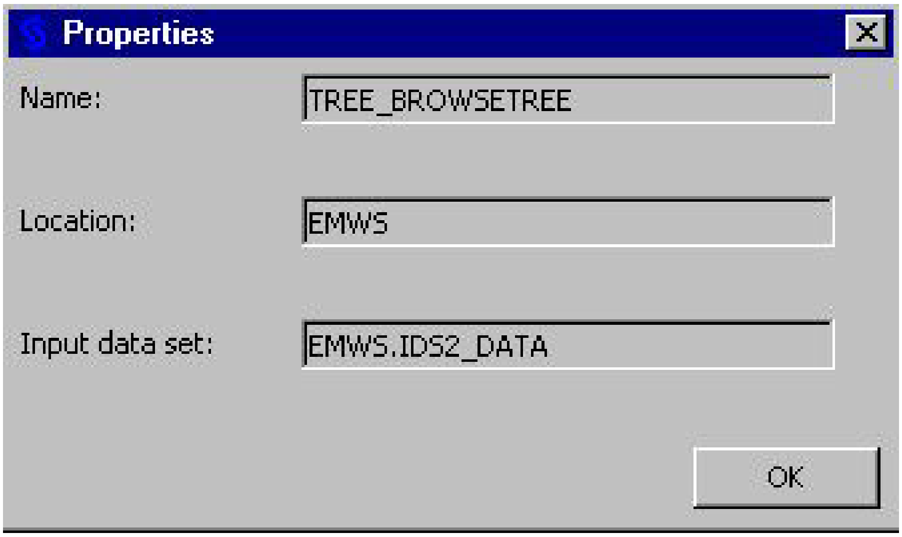
● Print to one page

○ Print each chart to a seperate page

○ Print same size as screen (may span multiple pages)

○ Print in Book Format (may span multiple pages)

- **Properties** — provides the name, location, and input data set name for the decision tree:


Properties

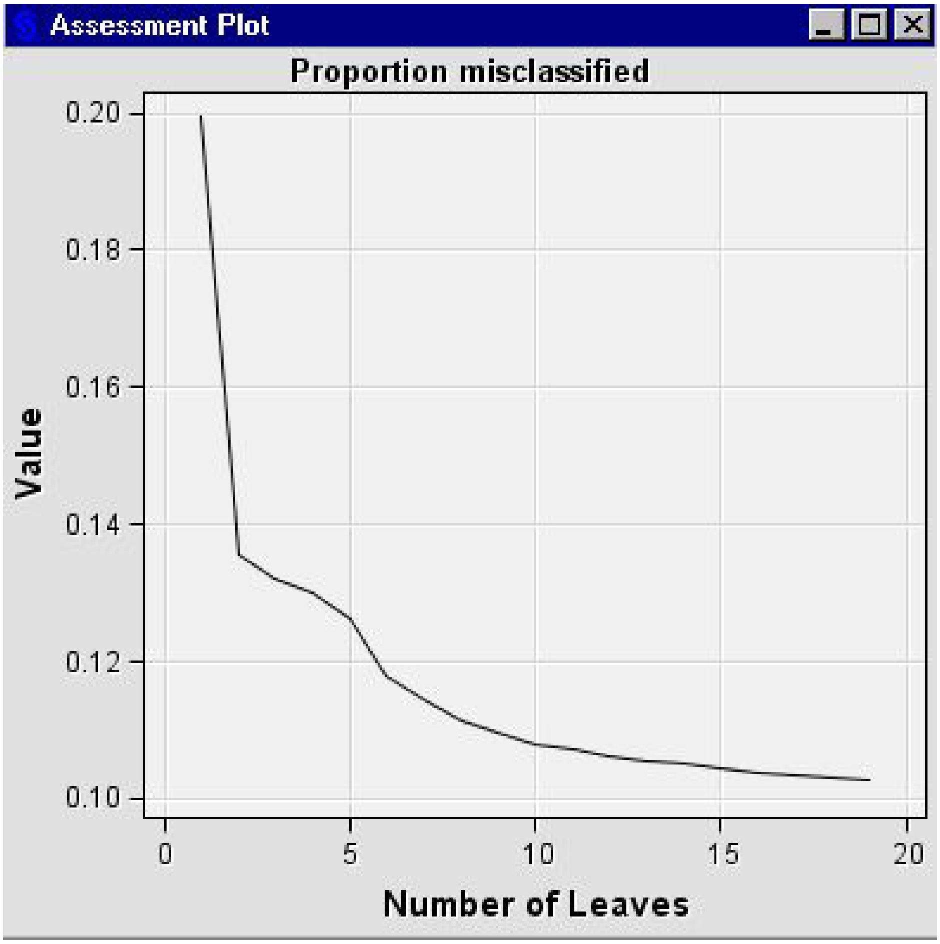
Name: TREE_BROWSETREE

Location: EMWS

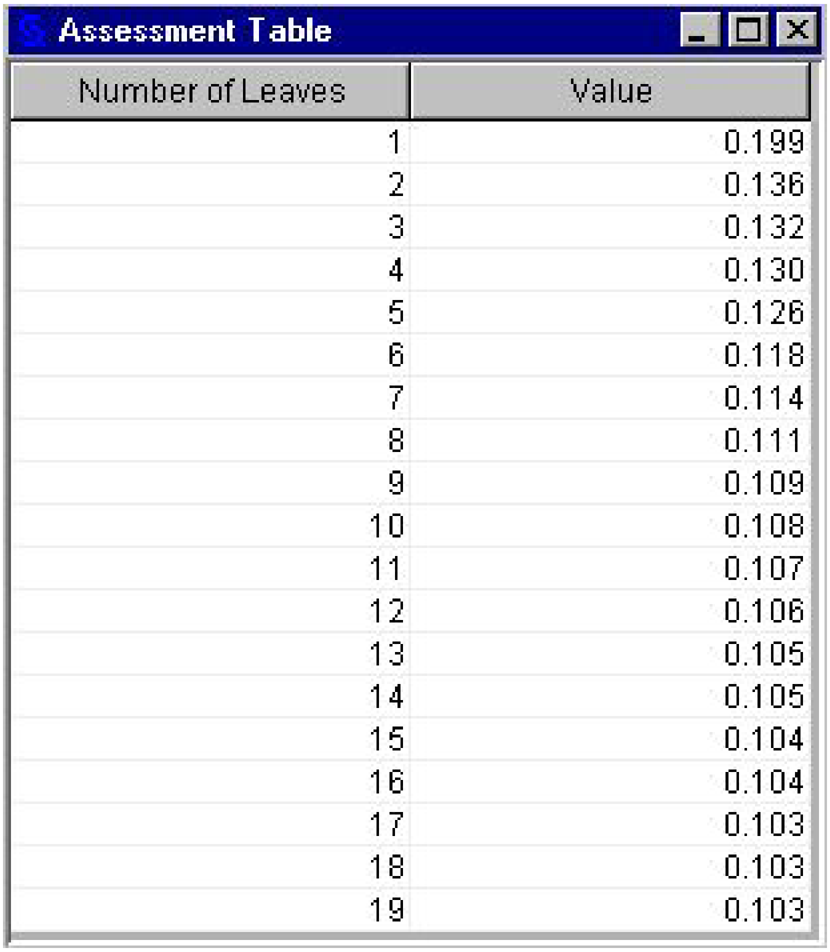
Input data set: EMWS.IDS2_DATA

- **Exit** — closes the Interactive Decision Tree application.

- **Edit**
  - **Copy** — creates an image of the tree diagram and places it in the clipboard. This enables you to paste a copy of the tree in other applications.
  - **Node Statistics** — displays the Node Statistics dialog box. The Node Statistics dialog box enables you to select which statistics will be displayed by the node text and tooltip.
- **View**
  - **Subtree Assessment Plot** — plots the assessment criterion versus the number of leaves. The assessment criterion for a categorical target is the proportion misclassified. The assessment criterion for an interval target is the average square error. The Assessment Plot is unavailable when there are multiple targets.

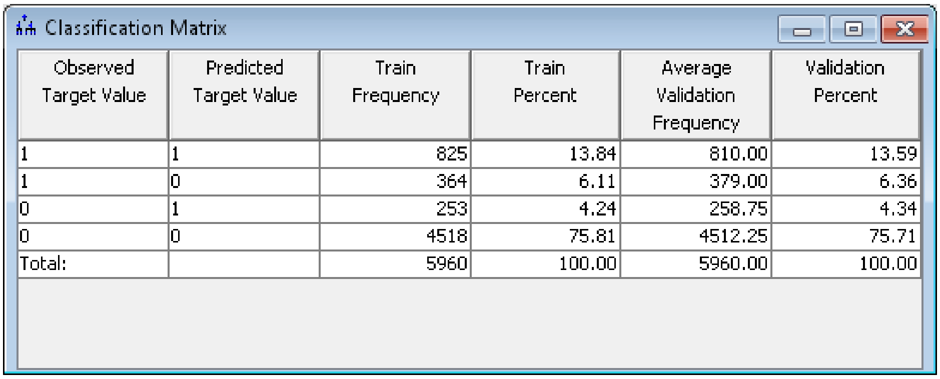


- **Subtree Assessment Table** — displays a table containing the assessment criterion value and the number of leaves. That is, the table information corresponds to the information displayed in the Assessment Plot. The Assessment Table is unavailable when there are multiple targets.



Number of Leaves	Value
1	0.199
2	0.136
3	0.132
4	0.130
5	0.126
6	0.118
7	0.114
8	0.111
9	0.109
10	0.108
11	0.107
12	0.106
13	0.105
14	0.105
15	0.104
16	0.104
17	0.103
18	0.103
19	0.103

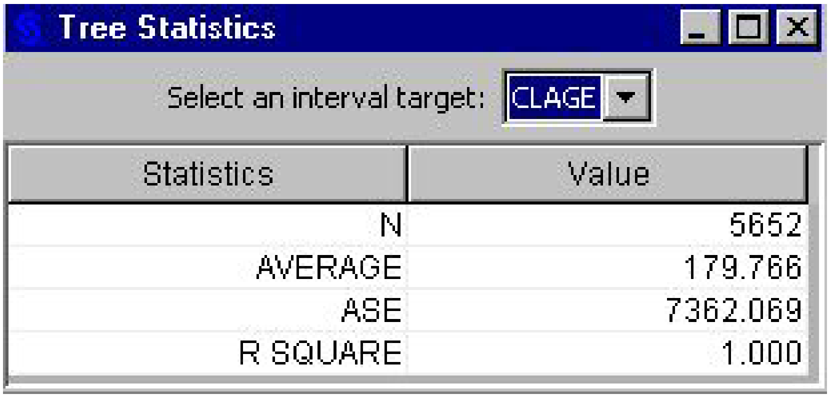
- **Classification Matrix** — displays a classification matrix for categorical targets. The matrix displays the total number and percentage of observations that were correctly and incorrectly classified by the decision tree.



Observed Target Value	Predicted Target Value	Train Frequency	Train Percent	Average Validation Frequency	Validation Percent
1	1	825	13.84	810.00	13.59
1	0	364	6.11	379.00	6.36
0	1	253	4.24	258.75	4.34
0	0	4518	75.81	4512.25	75.71
Total:		5960	100.00	5960.00	100.00

Note that **Validation Frequency** column is affected by the **Number of Repeats** property in the **Decision Tree** node. If you perform cross validation and set the **Number of Repeats** property to a value greater than 1, then the Classification Matrix displays the **Average Validation Frequency**. If the **Number of Repeats** property is equal to 1, then the Classification matrix displays the **Validation Frequency**.

- **Tree Statistics** — displays a table of statistics for decision trees with interval targets. The Tree Statistics table displays the number of observations (N), the average of the interval target (AVERAGE), the average squared error (ASE), and the R-square (R SQUARE) of the model.

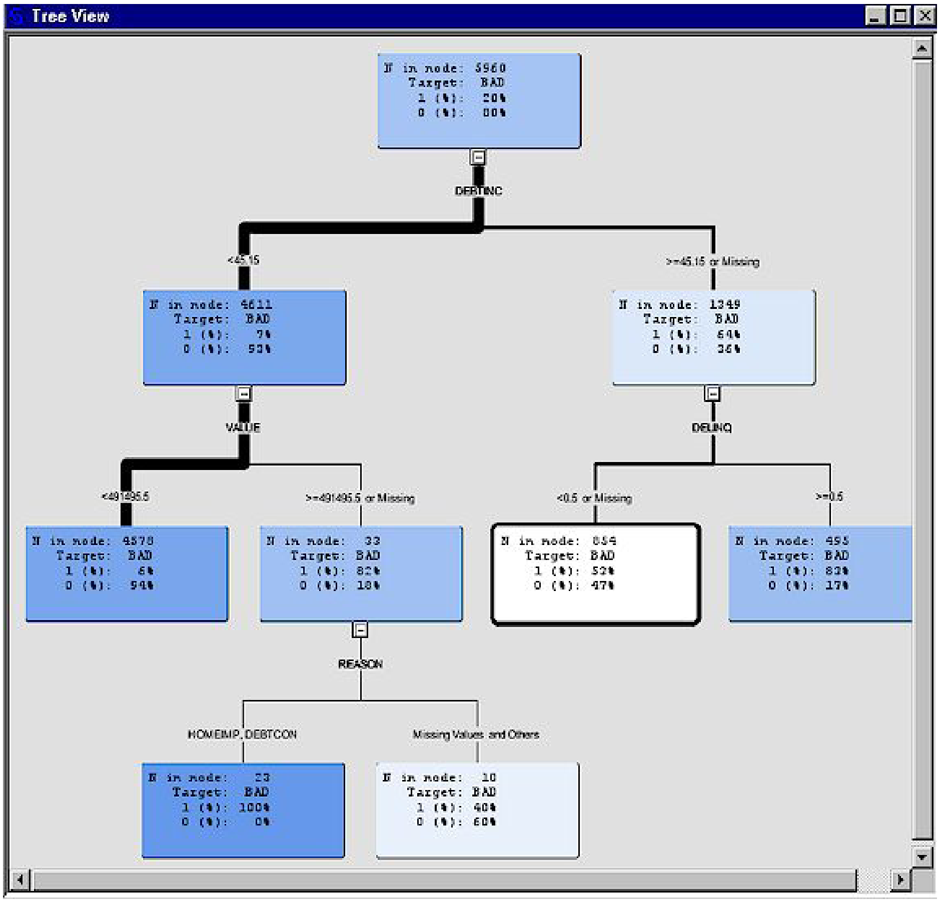


Statistics	Value
N	5652
AVERAGE	179.766
ASE	7362.069
R SQUARE	1.000

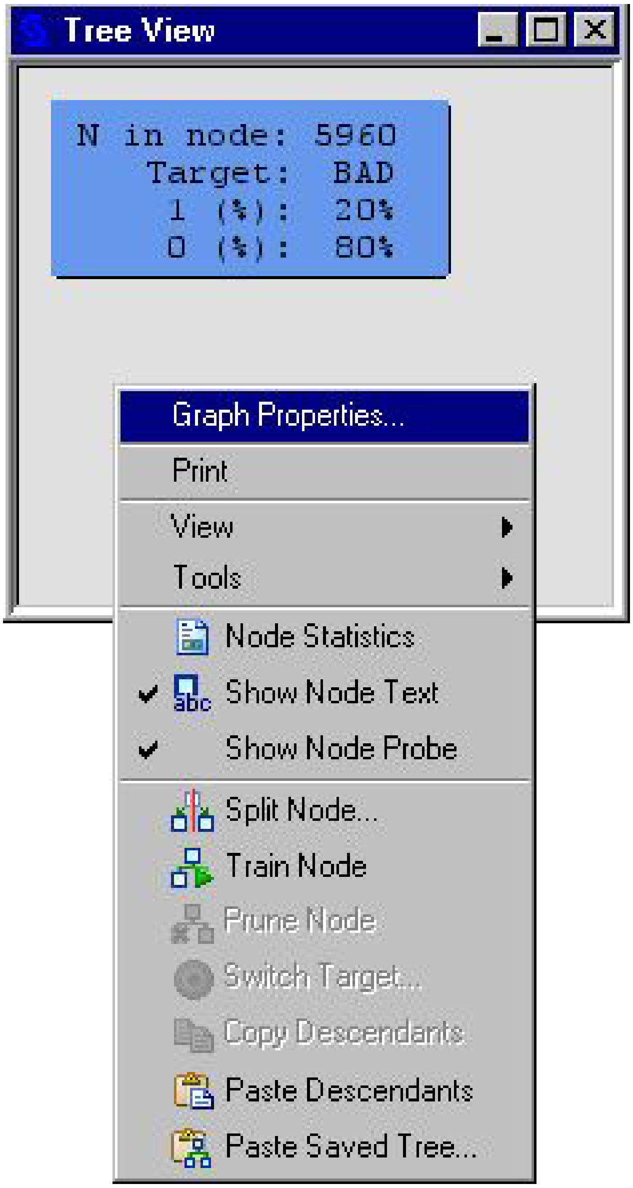
- **Tree** — displays a graphical representation of the decision tree model. A tree contains the following items:
  - **Root node** — the top node of a vertical tree or the left-most node of a horizontal tree that contains all observations.
  - **Internal nodes** — non-terminal nodes that contain the splitting rule. This includes the root node.
  - **Leaf nodes** — terminal nodes that contain the final classification for a set of observations.

  A default tree has the following properties:

  - It is displayed in vertical orientation.
  - The nodes are colored by the percentage of a categorical target value or the average of an interval target.
  - The line width is proportional to the ratio of the number of observations in a branch to the number of observations in the root node.
  - The line color is constant.



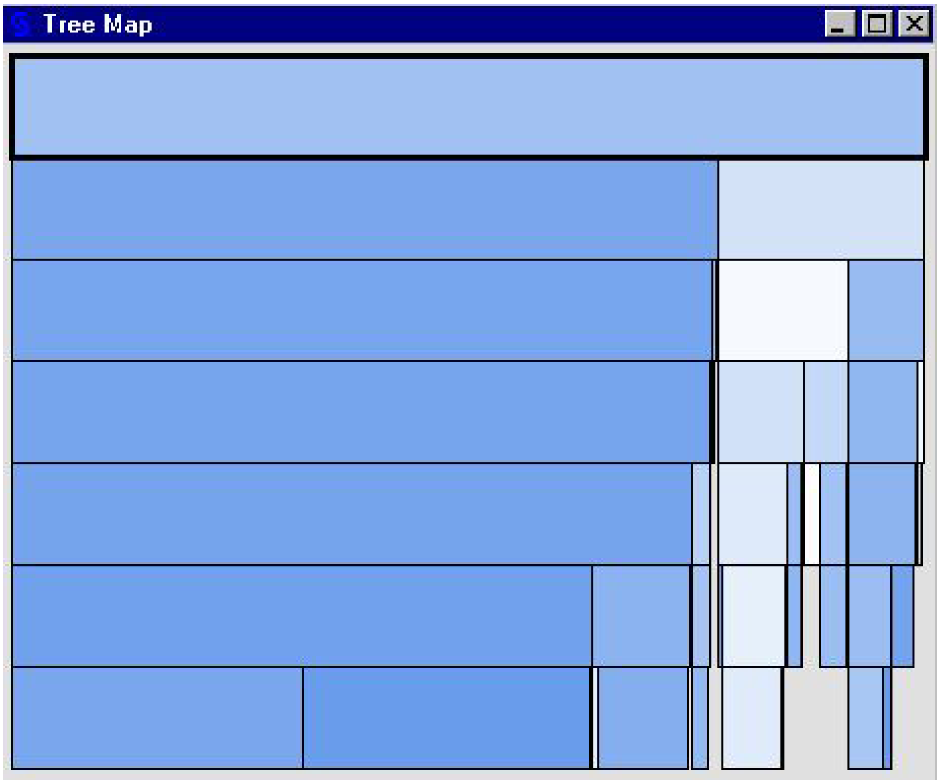
Right-click in the Tree View window and a pop-up menu appears:



```
N in node: 5960
   Target:  BAD
   1 (%):   20%
   0 (%):   80%
```

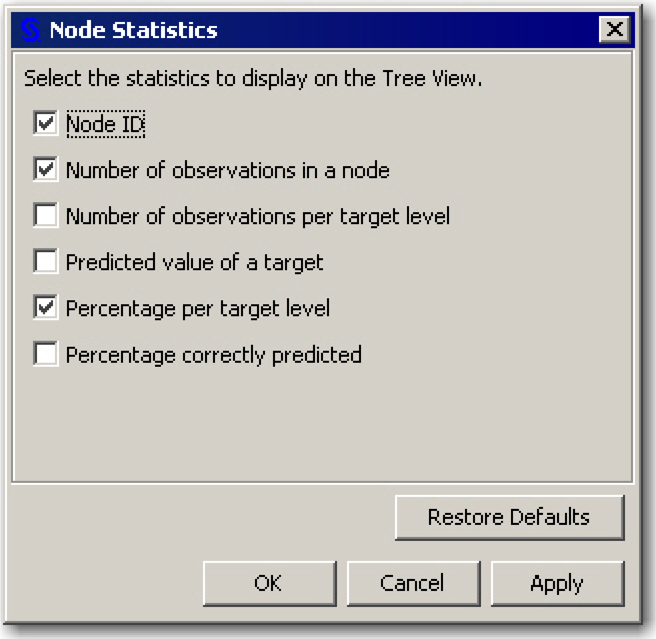
The Graph Properties, View items, and Tools items enable you to control the appearance of the Tree View. These properties are documented in Tree Properties Window on page 761.

- **Tree Map** — represents a compact graphical display of the tree. This view maintains the top-to-bottom, left-to-right orientation of the traditional Tree view while using node width to represent node size. The following display shows an example of the Tree Map view:



The root node is at the top of the tree. It contains all the observations in the data set. All other nodes contain a percentage of the total observations in the data. The node width is proportional to the number of observations in the node.

By default, the statistics for the node are displayed when you place the mouse pointer on a node in the Tree Map view. To display different statistics, right-click in the Tree Map window and select Node Statistics:



Select the statistics to display on the Tree View.

☑ Node ID

☑ Number of observations in a node

□ Number of observations per target level

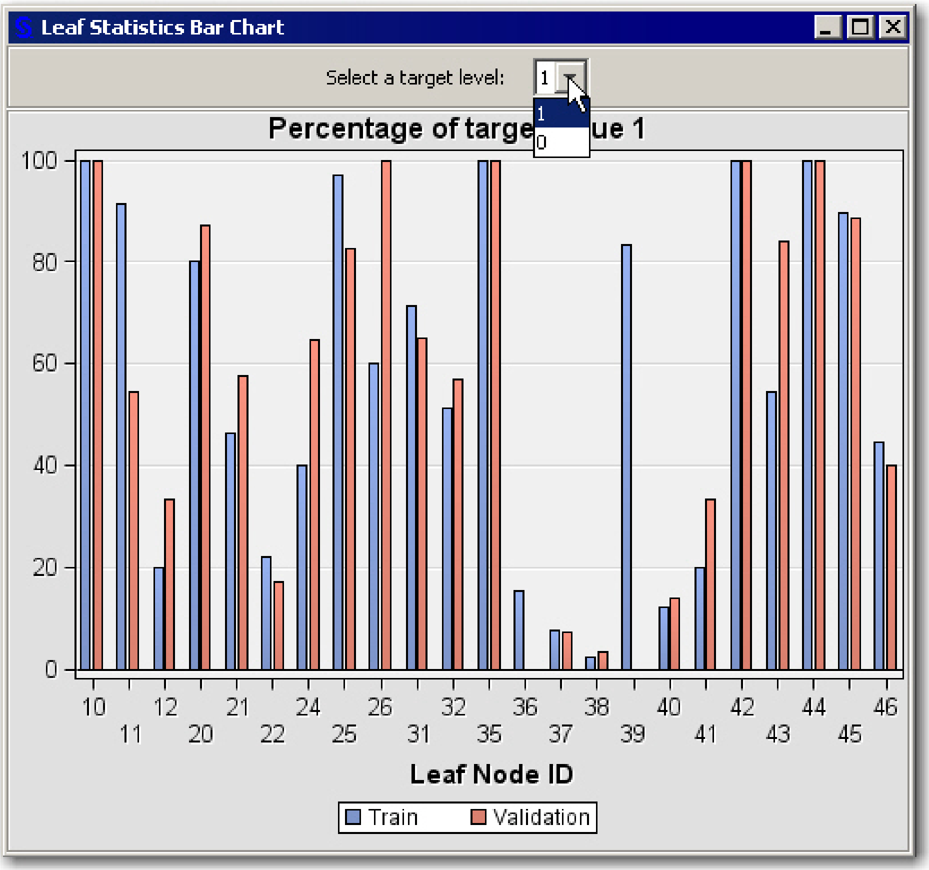
□ Predicted value of a target

☑ Percentage per target level

□ Percentage correctly predicted

- **Local Tree** — displays a partial view of the decision tree. The Local Tree view functions like the Tree view, but is rooted with a selected node from the Tree View and descends only one level.

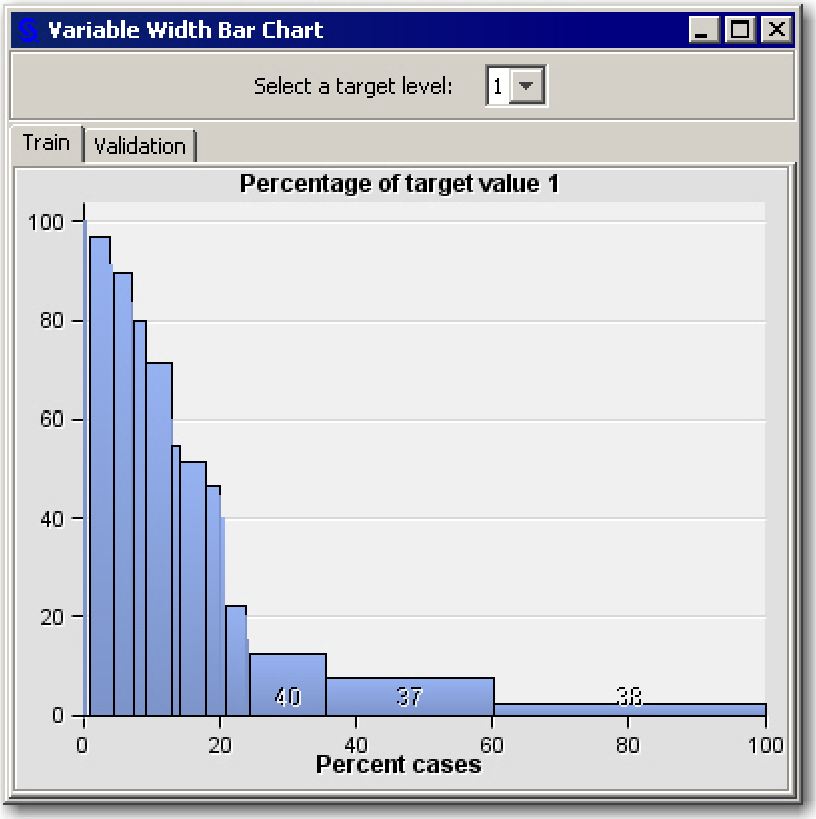
- **Leaf Statistics Table** — provides summary statistics for the leaves in the currently selected tree. The information that is provided in the table depends on whether the target is categorical or interval. When the target is categorical, the statistics reported include the Node ID, Predicted Target Value, Number of Cases, and Percentage of Correct Predictions. When the target is an interval variable, the reported statistics include the Node ID, Number of Cases, Average Target, and the Square Root of Average Square Error.
- **Leaf Statistics Bar Chart** — displays two bars for each leaf. One bar is for the training data and the other is for the validation data. The height of the bar can be represented two ways. For categorical targets, the height is the proportion of cases in the leaf with a specific target category. For interval targets, the height is the average of target values in a leaf. The leaves are ordered in ascending order of leaf node ID. If you right-click the chart, a pop-up menu appears that provides options that enable you to modify the appearance of the graph.



When there is a single target, a combo box appears at the top of the chart that enables you to designate which target value to graph. When there are multiple targets, a combo box appears at the top of the chart that allows you change which target to graph. Click on the down arrow to switch the target or target value and select the desired target or target value. The chart will be updated automatically for the new target variable.

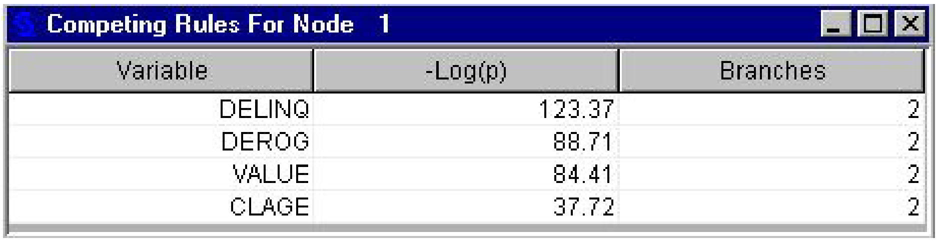
- **Variable Width Bar Chart** — displays a bar for each leaf in the training data set. The height of the bar can be represented two ways. For categorical targets, the height is the proportion of cases in the leaf with a specific target category. For interval targets, the height is the average of target values in a leaf. The leaves are ordered by descending values of the percentage of events in the training data set. The bar width is proportional to the number of training cases in a leaf.

A separate tab is given for the training and for the validation data. If you right-click the chart, a pop-up menu appears that provides options that enable you to modify the appearance of the graph.



When there is a single target, a combo box appears at the top of the chart that enables you to designate which target value to graph. When there are multiple targets, a combo box appears at the top of the chart that allows you change which target to graph. Click on the down arrow to switch the target or target value and select the desired target or target value. The chart will be updated automatically for the new target variable.

- **Competing Rules** — displays the alternative unused splitting variables that are saved in the primary selected node.



Variable	-Log(p)	Branches
DELINQ	123.37	2
DEROG	88.71	2
VALUE	84.41	2
CLAGE	37.72	2

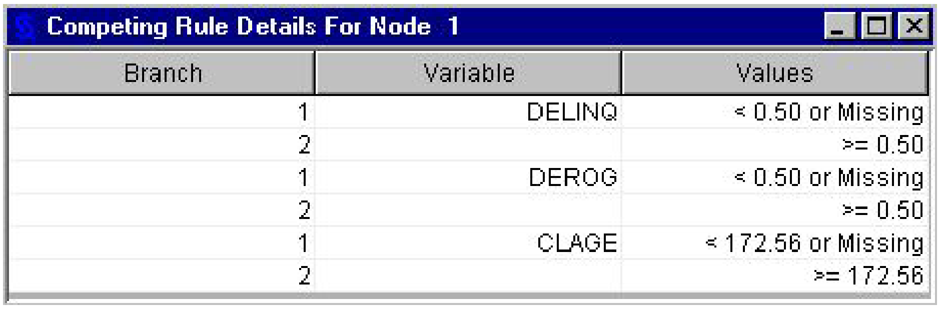
The number of the alternative splitting rules that are displayed in a competing rules table is determined by the value that you entered for the Number of Rules property. The default value for the Number of Rules property is five rules.

The measure of worth indicates how well a variable divides the data into each class. For the chi-square or F test splitting criteria, the opposite of the base-10 logarithm of the Worth is displayed. That is, for categorical targets, LOGWORTH = - log(chi-square p-value); for interval targets, LOGWORTH = - log(F test p-value). For the Gini, Entropy, or Variance Reduction splitting criteria, the worth is displayed. Good splitting variables have larger values of LOGWORTH or WORTH.

The Branches column displays the number of groups that the data is divided into according to the corresponding splitting rule.

If no nodes are selected or if the primary selected node is a leaf, no variables are listed.

- **Competing Rules Details** — displays the values of the primary splitting variables for each branch and the alternative unused rules details that are saved in the primary selected node.

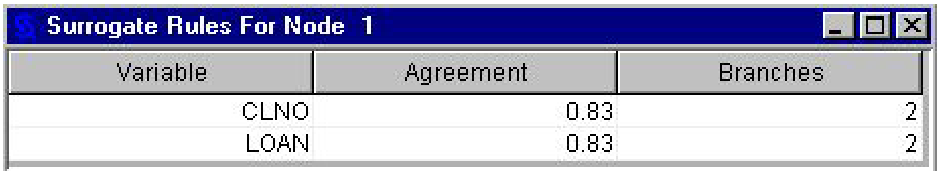


Competing Rule Details For Node 1

Branch	Variable	Values
1	DELINQ	< 0.50 or Missing
2		>= 0.50
1	DEROG	< 0.50 or Missing
2		>= 0.50
1	CLAGE	< 172.56 or Missing
2		>= 172.56

If no nodes are selected or if the primary selected node is a leaf, no variables are listed.

- **Surrogate Rules** — displays a list of surrogate rule variables that are saved in the primary selected node. The view is available only if you specified that surrogate rules should be saved in each node.



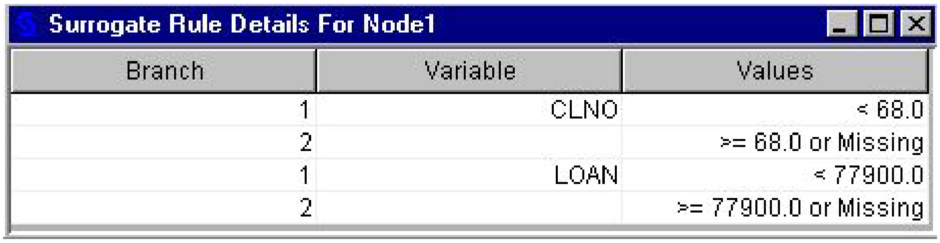
Surrogate Rules For Node 1

Variable	Agreement	Branches
CLNO	0.83	2
LOAN	0.83	2

The Branches column displays the number of groups that data is divided into according to the corresponding surrogate rule.

If no nodes are selected or if the primary selected node is a leaf, no variables are listed.

- **Surrogate Rules Details** — The Surrogate Rules Details view displays the values of surrogate rules for each branch that are saved in the primary selected node. The view is available only if you specified that surrogate rules should be saved in each node.



Surrogate Rule Details For Node1

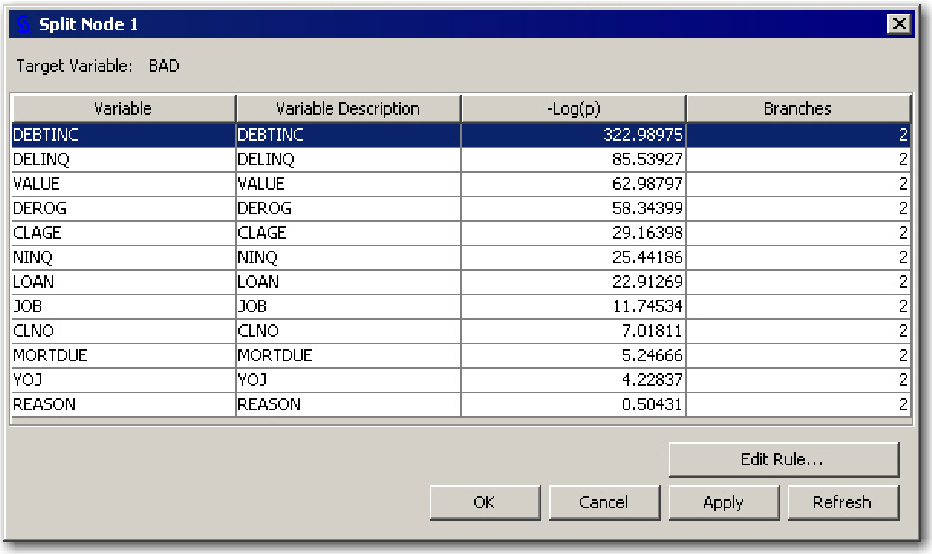
Branch	Variable	Values
1	CLNO	< 68.0
2		>= 68.0 or Missing
1	LOAN	< 77900.0
2		>= 77900.0 or Missing

If no nodes are selected or if the primary selected node is a leaf, no variables are listed.

- **Variable Importance** — displays a list of variables in the order of their importance in the tree. The Variables table lists the input variable name (Variable), the number of nodes that use the input variable as the primary splitting rule (Nodes), and the measure of importance that is computed from the training data set (Relative Importance). See "Variable Importance" on page 749 for details about how the Relative Importance statistic is computed. Variable usage is unavailable when there are multiple targets.

- **Split Rule Text** — displays a read-only window describing the split rule using SAS syntax. This includes all spit rules up to and including the selected node.

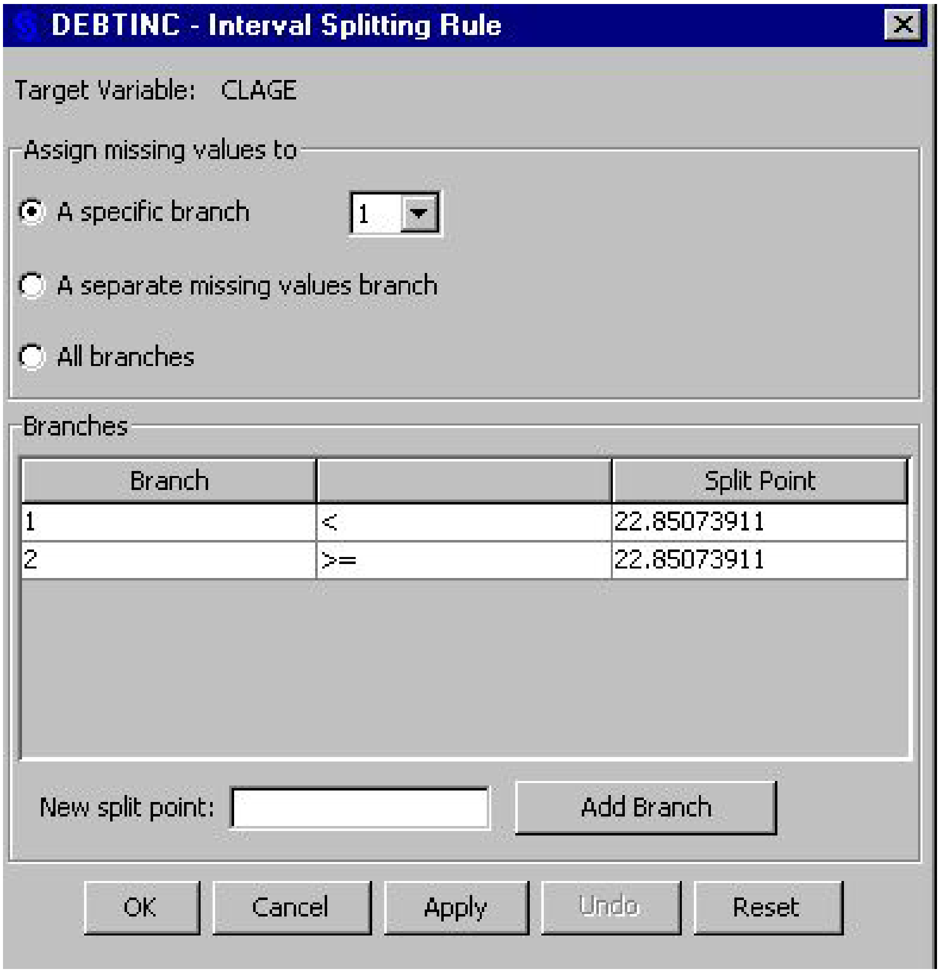
- **Action**
  - **Split Node** — opens the Split Node dialog box that enables you to select the splitting variable and to edit the splitting rule for that variable.



Target Variable: BAD

Variable	Variable Description	-Log(p)	Branches
DEBTINC	DEBTINC	322.98975	2
DELINQ	DELINQ	85.53927	2
VALUE	VALUE	62.98797	2
DEROG	DEROG	58.34399	2
CLAGE	CLAGE	29.16398	2
NINQ	NINQ	25.44186	2
LOAN	LOAN	22.91269	2
JOB	JOB	11.74534	2
CLNO	CLNO	7.01811	2
MORTDUE	MORTDUE	5.24666	2
YOJ	YOJ	4.22837	2
REASON	REASON	0.50431	2

When you click the **Edit Rule** button, a dialog box appears that enables you to edit the splitting rule:



Target Variable: CLAGE

Assign missing values to

● A specific branch 1

○ A separate missing values branch

○ All branches

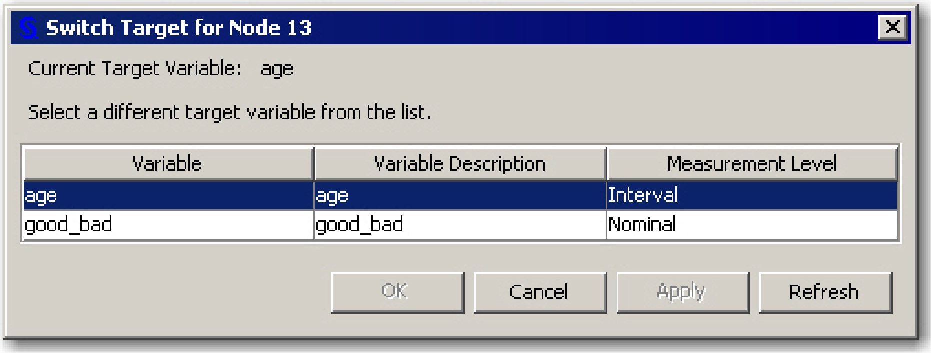
Branches

Branch		Split Point
1	<	22.85073911
2	>=	22.85073911

New split point:

There is a slightly different version of the dialog box for each of the three different types of variables: interval, nominal, and ordinal.

- **Train Node** — trains the selected node using the property settings in the Properties panel of the **Decision Tree** node.
- **Prune Node** — removes all branches and leaves that descend from the selected node.
- **Switch Target** — enables you to switch targets for any leaf. After you click on a leaf and select **Switch Target** from the **Train** menu, a Switch Target for Node # window appears.



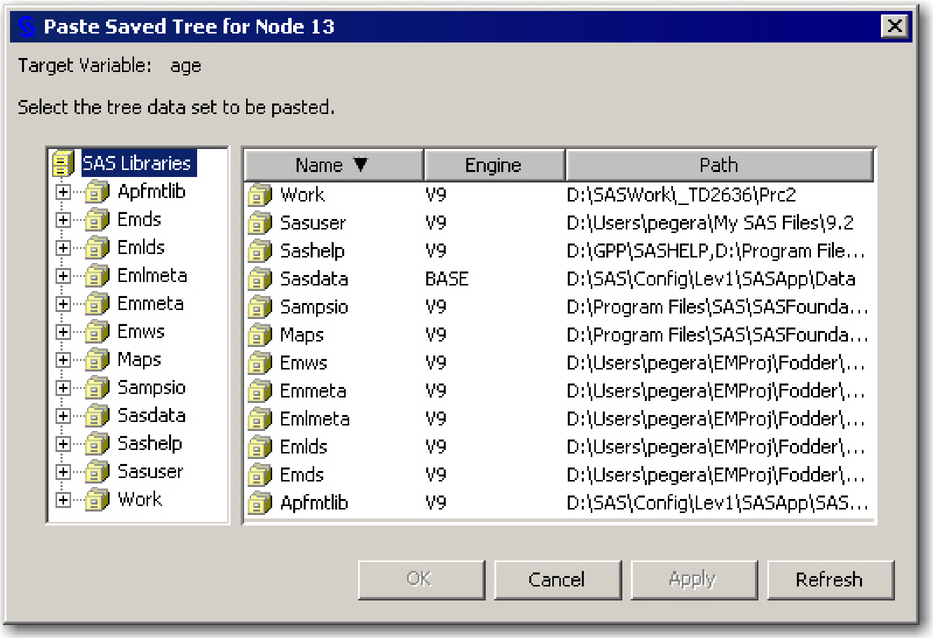
Current Target Variable: age

Select a different target variable from the list.

Variable	Variable Description	Measurement Level
age	age	Interval
good_bad	good_bad	Nominal

Any subsequent splits that are applied to that leaf will segment the new target variable conditional on all preceding nodes in the tree.

- **Copy Descendants** — copies a selected node's descendant branches, nodes, and leaves to the clipboard.
- **Paste Descendants** — pastes descendant branches, nodes, and leaves that have been copied to the clipboard to a selected node.
- **Paste Saved Tree** — enables you to choose a saved tree and to paste it to the Tree View.



Paste Saved Tree for Node 13

Target Variable: age

Select the tree data set to be pasted.

SAS Libraries
- Apfmtlib
- Emds
- Emlds
- Emlmeta
- Emmeta
- Emws
- Maps
- Sampsio
- Sasdata
- Sashelp
- Sasuser
- Work

Name ▼	Engine	Path
Work	V9	D:\SASWork_TD2636\Prc2
Sasuser	V9	D:\Users\pegera\My SAS Files\9.2
Sashelp	V9	D:\GPP\SASHELP,D:\Program File...
Sasdata	BASE	D:\SAS\Config\Lev1\SASApp\Data
Sampsio	V9	D:\Program Files\SAS\SASFounda...
Maps	V9	D:\Program Files\SAS\SASFounda...
Emws	V9	D:\Users\pegera\EMProj\Fodder\...
Emmeta	V9	D:\Users\pegera\EMProj\Fodder\...
Emlmeta	V9	D:\Users\pegera\EMProj\Fodder\...
Emlds	V9	D:\Users\pegera\EMProj\Fodder\...
Emds	V9	D:\Users\pegera\EMProj\Fodder\...
Apfmtlib	V9	D:\SAS\Config\Lev1\SASApp\SAS...

- **Window**
  - **Cascade Views** — causes all open views to be displayed in overlapping or cascading windows.
  - **Tile Views** — causes all open views to be displayed as nonoverlapping tiles of the Interactive Decision Tree window.
  - **Close All Views** — closes all views that are currently open in the Interactive Decision Tree window.

Interactive Decision Tree Toolbar

The Interactive Decision Tree window also provides a toolbar that is populated with icons that provide one-click access to some of the more frequently used menu items.

These icons are identified as follows:

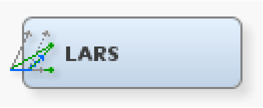
- 💾 — Save
- ↶ — Reset Data Tree
- 🖨 — Print
- ⎘ — Copy

- — Node Statistics
- — Split Node
- — Train Node
- — Prune Node
- — Switch Targets
- — Copy Descendants
- — Paste Descendants
- — Paste Saved Tree

Chapter 55

LARs Node

LARs Node



Overview of the LARs Node

General Information

The LARs node is located on the Model tab of the Enterprise Miner toolbar. LARs is an abbreviation for the Least Angle Regression algorithm that this data mining tool uses.

Data mining databases usually contain a large number of potential model inputs (independent or explanatory variables) that can be used to predict the value of a given target (a dependent or response variable). The LARs node can perform both variable selection and model-fitting tasks. When used for variable selection, the LARs node selects the variables in a continuous fashion, as coefficients for each selected variable grow from zero to the variable's least square estimates.

The LARs node can produce models that range from simple intercept models to least square regression models with many input variables. You specify the model selection criteria that you want to use to choose the optimal model. The LARs node sets the role of input variables that were excluded in the optimal model to rejected. Input variables that the LARs node rejects can be passed to subsequent modeling nodes, but their rejected status means that they will not be used as model inputs in successor nodes.

When using the LARs node to perform model fitting, LARs uses criteria from either least angle regression or the LASSO regression to choose the optimal model.

See the "Predictive Modeling" on page 149 section for information that applies to all of the predictive modeling nodes.

Least Angle Regression (LAR)

Least angle regression was introduced by Efron et. al (2004). Not only is the LARS algorithm useful for selecting the best-fitting model, but with a small modification, you can use LARs to efficiently produce LASSO solutions. Like the forward selection method used with regression models, the LAR algorithm produces a sequence of regression models. One model parameter is added with each step. The sequence of models terminates at the full least squares solution after all parameters have entered the model.

The algorithm starts by centering the covariates and response. Then the covariates are scaled so that they all have the same corrected sum of squares. Initially all coefficients are zero, as is the predicted response. The predictor that is most correlated with the current residual is identified, and a step is taken in the direction of this predictor. The length of the step determines the coefficient of this predictor. The step length is chosen so that some other predictor and the current predicted response have the same correlation with the current residual.

At this point, the predicted response moves in the direction that is equiangular between these two predictors. Moving in the equiangular direction ensures that the two predictors continue to have a common correlation with the current residual. The predicted response moves in this direction, until a third predictor has the same correlation with the current residual as the two predictors that are already in the model. A new direction is determined that is equiangular between these three predictors. The predicted response moves in this direction until a fourth predictor that has the same correlation with the current residual joins the set. This process continues until all predictors are in the model.

Lasso Selection (LASSO)

LASSO (Least Absolute Shrinkage and Selection Operator) selection arises from a constrained form of ordinary least squares, where the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter.

More precisely, let X = (x_1, x_2, . . . , x_m) denote the matrix of covariates, and let y denote the response, where the xi's have been centered and scaled to have unit standard deviation and mean zero, and y has mean zero. Then for a given parameter t, the LASSO regression coefficients β = (β_1, β_2, . . . , β_m) are the solution to the constrained optimization problem, as follows:

$$\text{minimize} \left\| y - X\beta \right\|^2 \text{ subject to} \sum_{j=1}^{m} \left| \beta_j \right| \leq t$$

If the LASSO parameter *t* is small enough, some of the regression coefficients will be exactly zero. This means that you can view the LASSO algorithm as a way to select a subset of the regression coefficients for each LASSO parameter. If you increase the LASSO parameter in discrete steps, you can obtain a sequence of regression coefficients, where the nonzero coefficients at each step correspond to selected parameters. Early implementations of LASSO selection (Tibshirani 1996) used quadratic programming techniques to solve the constrained least squares problem for each LASSO parameter of interest. Later, Osborne, Presnell, and Turlach (2000) developed a “homotopy method” that generates the LASSO solutions for all values of the parameter t. Efron et. al (2004) derived a variant of the least angle regression algorithm that can be used to obtain a sequence of LASSO solutions, from which all other LASSO solutions can be obtained by linear interpolation. LASSO can be viewed as an iterative procedure that makes either a single addition to or a single deletion from the set of nonzero regression coefficients at any step.

The LARs node offers model selection criteria that include SBC, BIC, AIC, AICC, CP, Validate ASE, and CV Press. The current Enterprise Miner LARs node processes interval and binary target variables.

Adaptive LASSO

Adaptive LASSO variable selection is a modification of the LASSO selection. In Adaptive LASSO, selection weights are applied to each of the parameters in forming the LASSO constraint.

More precisely, let $X=(X_1,X_2,\ldots X_p)$ denote the matrix of covariates, and let y denote the response where the X's have been centered and scaled to have unit standard deviation and mean zero. Suppose you can find a suitable estimator β^{hat} of the parameters in the true model, and you define a weight vector by

$$w = \frac{1}{\left|\hat{\beta}\right|^{\gamma}}$$ where $\gamma \geq 0$.

Then the Adaptive LASSO regression coefficients $\beta=(\beta_1,\beta_2,\ldots\beta_p)$ are the solution to the constrained optimization problem:

$$\text{minimize} \left\|y - X\beta\right\|^2 \text{ subject to} \sum_{j=1}^{m} \left|w_j \beta_j\right| \leq t$$.

If the target variable is binary, a logistic regression (PROC LOGISTIC) is run to obtain initial estimates that are then fed into PROC GLMSELECT using the INEST suboption of the SELECTION option on the model statement. Another logistic regression transforms the output from PROC GLMSELECT back to a probability between 0 and 1. If the target variable is interval, no INEST specification is required, and the starting parameters are the solution to the unconstrained least squares problem as the estimators for β^{hat} .

LARs Node and Missing Values

The LARs node handles missing values as follows:

- Observations that have missing values for target variables are excluded from the analysis.
- Missing values for categorical input variables are treated as an additional category.

The LARs node can be run before any other analysis, and the variables that LARs selects can be passed to any Enterprise Miner node or SAS System procedure.

LARs Node Properties

LARs Node General Properties

The following general properties are associated with the LARs node:

- **Node ID** — the Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first LARs node added to a diagram will have a Node ID of LARS. The second LARs node added to the diagram will have a Node ID of LARS2.

- **Imported Data** — The Imported Data property provides access to the Imported Data — LARs window. The Imported Data — LARs window contains a list of the ports that provide data sources to the LARs node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — LARs window. The Exported Data — LARs window contains a list of the output data ports that the LARs node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

LARs Node Train Properties

The following train properties are associated with the LARs node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the LARs node. Select the ... button to open a window containing the variables table. You can set the variable status to either **Use** or **Don't Use** in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.

LARs Node Train Properties: Modeling Techniques

- **Use Class Inputs** — the LARs node cannot select or exclude all levels of nominal variables simultaneously. When you set the Include Class Variables property to **No**, only interval input variables are used in the analysis. The default value for the Include Class Variables property is **Yes**.
- **Intercept** — specifies whether the intercept is included in the model fitting. The default value is **Yes**.
- **Variable Selection Method** — specifies the method used to perform model selection.

The following methods are available:

  - **LASSO** — The LASSO method adds and deletes parameters based on a version of ordinary least squares where the sum of the absolute regression coefficients is constrained.
  - **LAR** — The Least Angle Regression method starts with no effects in the model and then adds effects. The LAR parameter estimates at any step are more compact than the corresponding least squares parameter estimates. LAR is the default variable selection method.
  - **Adaptive LASSO** — The adaptive LASSO method adds and deletes parameters based on a version of ordinary least squares where the sum of the absolute regression coefficients is constrained subject to an adaptive weight for penalizing the coefficients.
  - **None** — No variable selection is performed. The ordinary least squares regression model is fitted.

- **Model Selection Criterion** — specifies the statistical criterion that you want to use in order to select the best candidate model. The model that yields the optimal value for the specified criterion is chosen. If the optimal value for the chosen criterion occurs for one or more models at more than one step, then the model that has the smallest number of parameters is chosen. The default value for the Model Selection Criterion property is SBC.

  The available Model Selection Criterion choices are as follows:

  - **SBC** — Schwarz Bayesian information criterion
  - **BIC** — Sawa Bayesian information criterion
  - **AIC** — Akaike's information criterion
  - **AICC** — corrected Akaike's information criterion
  - **CP** — Mallow C(p) statistic
  - **Validation** — average squares error for the validation data
  - **Cross Validation** — predicted residual sum of square with k-fold cross-validation

- **Path Stopping Criterion** — specifies the statistical criterion that you want to use in order to stop the selection process. The default value for the Path Stopping Criterion property is Maximum Steps.

  The available Path Stopping Criterion choices are as follows:

  - **None** — no path stopping criterion. None is the default setting for the Path Stopping Criterion property.
  - **SBC** — Schwarz Bayesian information criterion. SBC is the default model selection criterion.
  - **BIC** — Sawa Bayesian information criterion
  - **AIC** — Akaike's information criterion
  - **AICC** — corrected Akaike's information criterion
  - **CP** — Mallow C(p) statistic
  - **Validation** — average squares error for the validation data
  - **Cross Validation** — predicted residual sum of square with k-fold cross-validation

  - **Maximum Steps** — stops the algorithm after the number of steps that are specified in the Maximum Steps property.
- **Maximum Steps** — specifies the number of steps to perform when the Path Stopping Criterion property is set to Maximum Steps. The Maximum Steps property accepts positive integer values, and the default value is 200.

LARs Node Train Properties: Cross-Validation Options

Cross-validation properties are only activated if either the **Model Selection Criteria** property or the **Path Stopping Criteria** property is set to **Cross Validation**.

- **Cross-Validation** — specifies whether to perform cross-validation on the training data, and if so, the type of cross-validation to use. The default value for the Cross-Validation property is Random.

  The available types of cross-validation are as follows:

  - **Random** — assigns each training observation randomly to one of the n parts.
  - **Split** — requests that the ith part consists of training observations i, i + n, i + 2n, ...
  - **Block** — forms parts using n blocks of consecutive training observations.
- **CV Fold** — specifies the number of parts that the training data is subdivided into for the cross-validation. An integer value less than the number of observations in a training data set should be specified. The default value is 5.
- **Seed** — specifies an integer used to start the pseudo-random number generator for random cross-validation. The default value is 12345.

LARs Node Train Properties: Report

- **Details** — specifies the level of detail produced in output reports.

  The following are the available values for the Details property:

  - **All** — displays both Steps and Summary information. The default value for the Details property is Summary.
  - **Steps** — displays entry and removal statistics for the top 10 candidates for inclusion or exclusion at each step.
  - **Summary** — displays a table that contains ANOVA and fit statistics, as well as parameter estimates.

LARs Node Score Properties

The following score property is associated with the LARs node:

- **Excluded Variables** — when using LARs to perform variable selection, specifies what to do with variables that have been excluded from the model.
  - **Reject** — the role of excluded variables is set to Rejected.
  - **Hide** — excluded variables are dropped from the metadata that is exported by the node.
  - **None** — the role of excluded variables remains the same.

LARs Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

LARs Node Results

You can open the Results window of the LARs node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration settings for the LARs node. The information was captured when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the LARs node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set that was submitted to the LARs node. The table includes columns for the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the LARs node run.
  - **Output** — the SAS output of the LARs run. Typical SAS output includes information such
    - Variable Summary by Role
    - Observation Profiles
    - Class-Level Information and Dimensions
    - Stepwise Analysis of Variance (ANOVA) tables for the target variable and the Parameter Estimates Table for the Chosen Effects
    - Summary Table for the Variable Selection Method
    - Analysis of Variance (ANOVA) tables and the Parameter Estimates Table for the Chosen Effects of the Optimal Model
    - Fit Statistics
    - Assessment Score Rankings

  - **Flow Code** — the SAS code used to produce the output that the LARs node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS code that was created by the LARs node. The SAS code can be modified for use outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the LARs node does not generate PMML code.
- **Model**
  - **Coefficient Paths** — displays a line plot of the change in coefficient values across successive steps. The optimal model is indicated by a vertical line.
  - **Iteration Plot** — displays a line plot of the stepwise change of various fit statistic values. The optimal model is indicated by a vertical line that intersects the smallest value of the model selection criterion.
  - **Selected Variables**— displays a read-only table that provides selected variable names, class levels, and coefficients.
  - **Parameter Estimate (Absolute Values)** — displays a bar chart of the absolute parameter estimates against the selected variables. Bars are color coded by the algebraic sign of the parameter estimate for that variable.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — You can use the Graph wizard to modify an existing Results plot or to create a Results plot of your own.

LARs Node Example

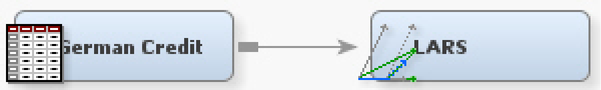
Create the Data Source

This example uses the sample SAS data set SAMPSIO.DMAGECR. You must use the SAMPSIO.DMAGECR data set to create an Enterprise Miner Data Source. Right-click the Data Sources folder in the Project Navigator and select **Create Data Source** to launch the Data Source wizard.

1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter `SAMPSIO.DMAGECR` in the Table field and click **Next**.
3. Continue to the Metadata Advisor step and choose the **Advanced Metadata Advisor**.
4. In the Column Metadata window, set the role of the variable AMOUNT to **Target**, set the role of CHECKING, INSTALLP, and SAVINGS to **Interval**, and set the role of GOOD_BAD to **Rejected**. Click **Next**.
5. There is no Decision Processing. Click **Next**.
6. Click **Finish**.

Place the Nodes on the Diagram Workspace

Drag the DMAGECR data source that you just created from the **Data Sources** folder of the Project Navigator onto the Diagram Workspace. Next, drag a LARs node from the **Model** folder onto the Diagram Workspace and connect the two nodes.



Configure the LARs Node

Click the LARs node in the Diagram Workspace to select it. Then, set the **Use Class Inputs** property in the LARs node Properties panel to **No**.

Run the LARs Node

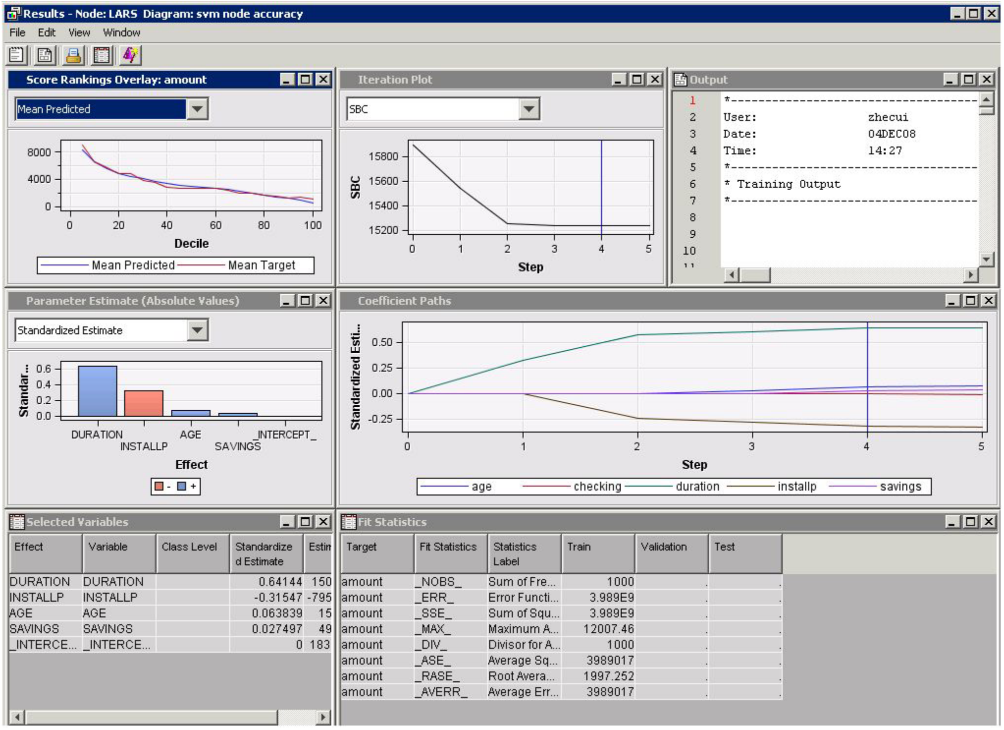
Right-click the LARs node in the Diagram Workspace and select **Run**.

Open the LARs Node Results Window

After the LARs node successfully runs, select **Results** when the Run Status window reports that the run has completed.

Examine the Results

The Results — LARS window opens. The Results — LARS window displays a line plot of the scoring ranking overlay, a bar chart of the absolute value of the standardized coefficients, a line plot of a variety of fit statistics, a table of the selected variables and their coefficients, a line plot of the coefficient paths, a table of the fit statistics for the target variables and the SAS output.



```
*-----------------------------------------
User:                     zhecui
Date:                     04DEC08
Time:                     14:27
*-----------------------------------------
* Training Output
*-----------------------------------------
```

Effect	Variable	Class Level	Standardized Estimate	Estim
DURATION	DURATION		0.64144	150
INSTALLP	INSTALLP		-0.31547	-795
AGE	AGE		0.063839	15
SAVINGS	SAVINGS		0.027497	49
_INTERCE...	_INTERCE...		0	183

Target	Fit Statistics	Statistics Label	Train	Validation	Test
amount	_NOBS_	Sum of Fre...	1000	.	.
amount	_ERR_	Error Functi...	3.989E9	.	.
amount	_SSE_	Sum of Squ...	3.989E9	.	.
amount	_MAX_	Maximum A...	12007.46	.	.
amount	_DIV_	Divisor for A...	1000	.	.
amount	_ASE_	Average Sq...	3989017	.	.
amount	_RASE_	Root Avera...	1997.252	.	.
amount	_AVERR_	Average Err...	3989017	.	.

- **Score Ranking Overlay Plot** — displays the comparison of the means of the predicted and the target variables at a series of percentiles of the training data and the validation data. It also displays the maximum and minimum values of the predicted and the target variables.
- **Parameter Estimates Plot** — This bar chart displays the absolute values of the standardized coefficients. In this plot, we can see that only AGE, DURATION, INSTALLP, and SAVINGS have nonzero coefficients. The sign of INSTALLP is minus, and the signs of all the other nonzero coefficients are plus.

- **Iteration Plot** — This plot displays the stepwise change of different fit statistics during the steps in the LAR algorithm. The pattern is obvious. The SBC value decreases quickly, and then rises gradually. The minimum value is realized during step 4. Therefore, the optimal model, based on the SBC criterion, is the model at step 4. The vertical line in the plot indicates the step that corresponds to the optimal model.
- **Selected Variables Table** — lists the variables that were chosen as model predictors based on SBC criterion scores and parameter estimates of the selected variables.
- **Coefficient Path Plot** — displays the stepwise change in value of input variable coefficients as variables enter and exit the model. The vertical line indicates the step that corresponds to the optimal model. In this example, only four predictor variables are identified in the optimal model. All other input variables have coefficients of zero.
- **Fit Statistics Table** — displays a list of training data fit statistics for the target variable AMOUNT.
- **SAS Output Window** — The SAS Output window for the LARs node displays the following charts:
  - **Variable Summary** — The Variable Summary of the LARS Output window breaks down the input variables by Role, Level, and Count.
  - **Predicted and Decision Variables** — This table provides the model fitting information.
  - **The GLMSelect Procedure** — GLMSelect is the SAS procedure that implements the LAR or LASSO variable selection. The GLMSelect output includes the ANOVA table and the parameter estimates table of the selected variables at each step as the variables enter or exit the model as well as the reprint of the ANOVA table and the parameter estimate table for the optimal model.

    Note: There is a selection summary table in the LARs Results that gives the sequence of the variables as they enter the model. During LASSO variable selection, variables can exit the model at some steps.
  - **Fit Statistics** — The set of fit statistics that are displayed in the Results window.
  - **Assessment Score Rankings** — the data that was used to generate the score ranking overlay plot.
  - **Assessment Score Distribution** — a stepwise summary of the model assessment scores.

References

Tibshirani, R “Regression Shrinkage and Selection Via the Lasso.” 1996. *Journal of the Royal Statistical Society: Series B* 58: 267–288.

Bradley, E., Hastie, T., Tibshirani, R., and Johnstone “Least Angle Regression.” 2004. *Annals of Statistics* 32 (2): 407–499.

Hastie, T., Cohen, R., and Friedman, J. 2004. *Manual of PROC GLMSELECT*. Cary, NC: SAS Press.

Chapter 56
Memory-Based Reasoning (MBR) Node

Memory-Based Reasoning (MBR) Node

Overview of the Memory-Based Reasoning Node

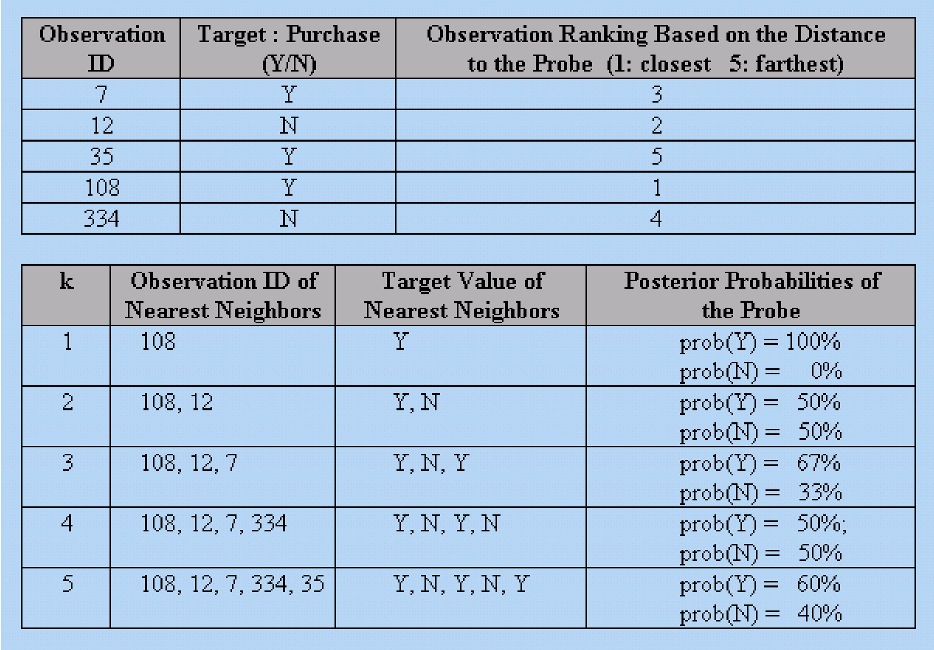
The Memory-Based Reasoning node belongs to the Model category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA). Memory-based reasoning is a process that identifies similar cases and applies the information that is obtained from these cases to a new record. In Enterprise Miner, the Memory-Based Reasoning (MBR) node is a modeling tool that uses a k-nearest neighbor algorithm to categorize or predict observations.

The k-nearest neighbor algorithm takes a data set and a probe, where each observation in the data set is composed of a set of variables and the probe has one value for each variable. The distance between an observation and the probe is calculated. The k observations that have the smallest distances to the probe are the k-nearest neighbor to that probe.

In Enterprise Miner, the k-nearest neighbors are determined by the Euclidean distance between an observation and the probe. Based on the target values of the k-nearest neighbors, each of the k-nearest neighbors votes on the target value for a probe. The votes are the posterior probabilities for the binary or nominal target variable.

The following display shows the voting approach of these neighbors for a binary target variable when different values of k are specified. In this example, observations 7, 12, 35, 108, and 334 are the five closest observations to the probe. Observations 108 and 35 have the shortest and the longest distances to the probe, respectively.

The k-nearest neighbors are first k observations that have the closest distances to the probe. If the value of k is set to 3, then the target values of the first three nearest neighbors (108, 12, and 7) are used. The target values for these three neighbors are Y, N, and Y. Therefore, the posterior probability for the probe to have the target value Y is 2/3 (67%).



Observation ID	Target : Purchase (Y/N)	Observation Ranking Based on the Distance to the Probe (1: closest 5: farthest)
7	Y	3
12	N	2
35	Y	5
108	Y	1
334	N	4

k	Observation ID of Nearest Neighbors	Target Value of Nearest Neighbors	Posterior Probabilities of the Probe
1	108	Y	prob(Y) = 100% prob(N) = 0%
2	108, 12	Y, N	prob(Y) = 50% prob(N) = 50%
3	108, 12, 7	Y, N, Y	prob(Y) = 67% prob(N) = 33%
4	108, 12, 7, 334	Y, N, Y, N	prob(Y) = 50%; prob(N) = 50%
5	108, 12, 7, 334, 35	Y, N, Y, N, Y	prob(Y) = 60% prob(N) = 40%

If the target is an interval variable, then the average of the target values of the k-nearest neighbors is calculated as the prediction for the probe observation.

See the Predictive Modeling section for information that applies to all of the predictive modeling nodes.

Data Set Requirements of the Memory-Based Reasoning Node

The Memory-Based Reasoning node requires exactly one target variable. If more than one target variable is defined in a predecessor node of the process flow diagram, you must select one variable as the target in order to run the node. The target variables can be binary, nominal, or interval. You cannot model an ordinal target variable, but ordinal target variables can be modeled as interval targets.

The Memory-Based Reasoning node assumes that the variables with a model role of "input" are numeric, orthogonal to each other, and standardized. You may use the Princomp node to generate numeric, orthogonal, and standardized variables that can be used as inputs for the Memory-Based Reasoning node.

Memory-Based Reasoning Node Properties

Memory-Based Reasoning Node General Properties

The following general properties are associated with the Memory-Based Reasoning Node node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Memory-Based Reasoning node added to a diagram will have a Node ID of MBR. The second Memory-Based Reasoning node added to a diagram will have a Node ID of MBR2, and so on.

- **Imported Data** — the Imported Data property provides access to the Imported Data — Memory-Based Reasoning window. The Imported Data — Memory-Based Reasoning window contains a list of the ports that provide data sources to the Memory-Based Reasoning node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Exported Data** — the Exported Data property provides access to the Exported Data — Memory-Based Reasoning window. The Exported Data — Memory-Based Reasoning window contains a list of the output data ports that the Memory-Based Reasoning node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Memory-Based Reasoning Node Train Properties

The following train properties are associated with the Memory-Based Reasoning node:

- **Variables** — Use the Variables property to specify how to use the variables in your data source. Select the ... button to the right of the Variables property to open a variables table. In the table, you can set the Use status for each variable to either Yes (default) or No, and the Report status of each variable to **No** (default) or **Yes**.

- **Method** — Use the Method property of the Memory-Based Reasoning node to specify the following data representation that you want to use to store the training data observations and then retrieve the nearest neighbors.

  - **RD-Tree** — (default setting) The Reduced Dimensionality Tree method stores the training data set observations in memory. RD-Tree creates a binary tree by repeatedly partitioning the data set into subsets such that the number of observations in each of the subsets is small. RD-Tree splits a node along some dimension, usually the one that has the greatest variation for observations in that node. This split generally occurs at the median value of the node. RD-Tree normally works better than Scan up to a dimensionality of 100. When using RD-Tree on data sets with high dimensionality, using the Epsilon property may improve computational efficiency.

  - **Scan** — the Scan method retrieves a nearest neighbor by scanning naively through every observation in the data set and calculating its distance to a probe observation.

- **Number of Neighbors** — Use the Number of Neighbors property of the Memory-Based Reasoning node to specify k, the number of nearest neighbors that you want to use to categorize or predict observations. The Number of Neighbors property allows integer values larger than 1. The default setting is 16.
- **Epsilon** — Use the Epsilon property of the Memory-Based Reasoning node if you use the RD-Tree method to store the data and you want to specify a non-zero number to conduct an approximate nearest neighbor search where the nearest neighbors must be at most "epsilon" away from the actual nearest neighbors to terminate the search. Judicious use of the Epsilon property can result in significance performance improvements in cases with large dimensionality. The Epsilon property is a double with a minimum value of 0.0 and a default value of 0.0. The Epsilon property is not relevant for the Scan method.
- **Number of Buckets** — Use the Number of Buckets property of the Memory-Based Reasoning node if you are using the RD-Tree method and you want to specify the maximum number of buckets that you want to allow a leaf node to grow to before splitting into a branch with two new leaves. The Buckets property is an integer with a minimum value of 2 and a default value of 8. The Number of Buckets property is not relevant for the Scan method.
- **Weighted** — When you are using an interval target variable, the Memory-Based Reasoning node weights input variables by default. Each input variable is weighted by the absolute value of its correlation to the target variable. Setting the Weighted property to No suppresses input variable weighting. The default setting of the Weighted property is **Yes**.
- **Create Nodes** — Setting the Create Nodes property of the Memory-Based Reasoning node to **Yes** adds a variable called _NNODES_ to the Memory-Based Reasoning node output data set. The _NNODES_ variable shows the number of point comparisons that were performed during node calculations. The _NNODES_ information is often useful as a point of comparison. The default setting for the Create Nodes property is **No**.
- **Create Neighbor Variables** — Setting the Create Neighbor Variables property to Yes writes the nearest neighbors for each observation in the Memory-Based Reasoning node output score data set. The variables for the nearest neighbors are _N1, _N2, ..., _Nk, where k is the number of nearest neighbors specified. The values of these nearest neighbor variables are the values of the ID variable (if ID variable was specified) or positive integers corresponding to row numbers in the DATA= data set. The default setting for the Create Neighbor Variables property is **Yes**.

  Note: If an ID variable was not specified, then the values of the nearest neighbor variables _Ni are no longer valid if the DATA= data set is sorted.

Memory-Based Reasoning Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.

- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Memory-Based Reasoning Node Results

Results Window Menus

You can open the Results window of the Memory-Based Reasoning node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Memory-Based Reasoning node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Memory-Based Reasoning node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that Enterprise Miner used to train the node.
- **SAS Results**
  - **Log** — the SAS log of the Memory-Based Reasoning node run.
  - **Output** — the SAS output for the Memory-Based Reasoning node. The output contains a list of the predicted and decision variables, fit statistics, classification table, event classification table, and assessment statistics for each decile.
  - **Flow Code** — the Memory-Based Reasoning node does not produce SAS flow code.
- **Scoring**
  - **SAS Code** — SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the MBR node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — The Fit Statistics table for the AutoNeural node includes the following columns:
    - **Target** — name of the target variable.
    - **Fit Statistics** — each row in the Fit Statistics column contains a different statistic for model goodness-of-fit. The variable name that is used in SAS code for each fit statistic appears in the Fit Statistics column.
    - **Statistics Label** — an easy-to-understand label name for each fit statistic variable.

- **Train** — The fit statistic value for the target variable in the training data set.
- **Validation** — The fit statistic value for the target variable using the validation data set.
- **Test** — The fit statistic value for the target variable using the test data set.

Some fit statistics are not calculated for both interval and non-interval targets. The "Fit Statistics Variable Table" on page 847 provides a key to which fit statistics are calculated according to the fit statistics variable table.

- **Residual Statistics** — The Residual Statistics plot displays a box-and-whisker plot for the residual measurements when the target is interval.
- **Classification Chart** — The Classification Table chart displays a stacked bar chart of the classification results for a categorical target variable.
- **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

  The Score Rankings Overlay plot displays both train and validate statistics on the same axis. By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any).

  - posterior probability of target event
  - number of events
  - cumulative and noncumulative lift values
  - cumulative and noncumulative % response
  - cumulative and noncumulative % captured response
  - gain
  - actual profit or loss
  - expected profit or loss.
- **Score Rankings Matrix** — The score ranking matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. Plots can also be created for report variables. The y-axis choices for the Score Rankings Matrix Chart are

  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response.
- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values

on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. For interval targets, observations are grouped into bins, based on the actual predicted values of the target.

The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. Use the y-axis list to select the score distribution statistic that you want to chart. The default chart is Percentage of Events. The chart choices are

- Percentage of Events
- Number of Events
- Cumulative Percentage of Events.

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — Opens the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Modifying Results Tables and Plots

You can modify and enhance Results graph attributes, whether you want to graphically explore effects of different variable roles, add or change response variables, or modify graph attributes to emphasize particular results or enhance appearance for presentation purposes.

For more information about manipulating plot variable roles and response variables, see the Data Options Dialog topic in the Enterprise Miner User Interface Help. For more information about modifying individual plot attributes, see the Enterprise Miner User Interface Help section on the Graph Properties window.

Fit Statistics Variable Table

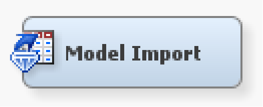
		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
AIC	Akaike's Information Criterion	Yes	Yes
APROF	Average Profit of the Target	Yes	No
ASE	Average Squared Error	Yes	Yes
AVERR	Average Error Function	Yes	Yes
DFE	Degrees of Freedom for Error	Yes	Yes

		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
DFM	Model Degrees of Freedom	Yes	Yes
DFT	Total Degrees of Freedom	Yes	Yes
DIV	Divisor for _ASE_	Yes	Yes
ERR	Error Function	Yes	Yes
FPE	Final Prediction Error	Yes	Yes
MAX	Maximum Absolute Error	Yes	Yes
MISC	Misclassification Rate	Yes	No
MSE	Mean Squared Error	Yes	Yes
NOBS	Sum of Frequencies	Yes	Yes
_NW_	Number of Estimated Weights	Yes	Yes
PROF	Total Profit	Yes	Yes
RASE	Root Average Squared Error	Yes	Yes
RFPE	Root Final Prediction Error	Yes	Yes
RMSE	Root Mean Squared Error	Yes	Yes
SBC	Schwarz's Bayesian Criterion	Yes	Yes
SSE	Sum of Squared Errors	Yes	Yes
SUMW	Sum of Case Weights Times Frequency	Yes	Yes
WRONG	Number of Wrong Classifications	Yes	No

Chapter 57

Model Import Node

Model Import Node



Overview of the Model Import Node

The Model Import Node is on the Model tab of the Enterprise Miner tools bar. You use the Model Import node to import and assess a model that was not created by one of the Enterprise Miner modeling nodes. You can then use the Model Comparison node to compare the user defined model(s) with a model(s) that you developed with an Enterprise Miner modeling node. This process is called integrated assessment.

The source for data that you can import include the following:

- Models or scored data sets that are represented as an Input Data Source node.
- Models that you developed in the SAS Code node, such as PROC LOGISTIC or PRINCOMP model, or a customized data step.
- A predicted model with prediction variables that is exported from an Enterprise Miner node.

See the "Predictive Modeling" on page 149 section for information that applies to all of the predictive modeling nodes.

Using the Model Import Node

Overview

The Model Import node must follow a node that exports a training, validation, test or raw data set. At least one of these data sets is required to generate assessment statistics. The export data set must contain a predicted values variable(s) and a target variable.

There are two ways that you can import a scored data set into your process flow and pass it to the Model Import node.

- Define an existing scored data set that contains predicted values for the target in the Input Data Source node and then pass it to the Model Import node.
- Write a SAS program in the SAS Code node to create a scored data set. Assign the target variable role to at least one variable and the prediction variable role to the prediction variables in an intermediate Metadata node, and then pass the scored data set to the Model Import node.

Defining the Scored Data Set in the Input Data Source Node

You can use the Input Data Source node to represent a scored data set that contains predicted values for the target variable. The scored data set can then be passed to the Model Import node for assessment in the Model Manager or in the Model Comparison node.

A SAS modeling procedure, such as PROC LOGISTIC, or a customized data set can be used to create the scored data set.

Creating the Scored Data Set in the SAS Code Node

You can use the SAS Code node to write a SAS program that creates a scored data set. The scored data set can then be exported to the Model Import node for assessment. You can create the scored data set from either a SAS modeling procedure or customized SAS DATA step code.

Model Import Node Properties

Model Import Node General Properties

The following general properties are associated with the Model Import node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Model Import node added to a diagram will have a Node ID of MdlImp. The second Model Import node added to a diagram will have a Node ID of MdlImp2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Model Import window. The Imported Data — Model Import window contains a list of the ports that provide data sources to the Model Import node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Model Import window. The Exported Data — Model Import window contains a list of the output data ports that the Model Import node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Model Import Node Train Properties

The following train properties are associated with the Model Import Node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the Model Import Node. Select the ... button to open a window containing the variables table. You can set the variable status to either Use or Don't Use in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Apply Decisions** — Use the Apply Decisions property to specify if prior probabilities that were defined in preceding nodes are applied to the imported model. When set to **No**, prior probabilities will be ignored even if they are defined in the preceding flow.

Model Import Node Train Properties: Predicted Variables

- Mapping Editor — Click the ... button to open an editor that enables you to specify the predicted variable that contains the predicted response and to specify the predicted variables that correspond to the posterior probabilities of the associated target levels.

- Import Type — specifies the source of the model to import and assess. Select Data to indicate that the predicted or posterior variables are in the incoming training data set. Select Registered Model to indicate that the predicted or posterior variables need to be computed by applying the model score code to the training data set.

- Model Name — Click the ... button to open a dialog that enables you to select a model that is registered in the metadata server.

- **Model Path** — displays the physical path to the model that is registered in the metadata server.

Model Import Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Model Import Node Results

You can open the Results window of the Model Import Node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Model Import Node Properties Panel. The information was captured when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Model Import Node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Model Import Node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Model Import Node run.
  - **Output** — the SAS output of the Model Import Node run.
  - **Flow Code** — the SAS code used to produce the output that the Model Import Node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the Model Import Node does not generate SAS Code. The SAS Code menu item is dimmed and unavailable in the Model Import Results window.
  - **PMML Code** — the Model Import Node does not generate PMML code.
- **Assessment**

- **Fit Statistics** — displays a table of fit statistics that includes the following:
  - Error Function
  - Sum of Squared Errors
  - Maximum Absolute Error
  - Divisor for Absolute Squared Error
  - Sum of Frequencies
  - Number of Wrong Classifications
  - Frequency of Classified Cases
  - Misclassification Rate
  - Average Squared Error
  - Root Average Squared Error
  - Average Error Function
- **Classification Chart** — The Classification chart displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represent the percentage of total observations.
- **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles and observations in a decile are used to calculate the statistics that are plotted in deciles charts. The Score Rankings Overlay plot displays both train and validate statistics on the same axis.

  By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations.

  The vertical axis displays the following values:
  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response
  - Total Profit
  - Expected Profit.
- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used. For categorical targets, observations are grouped into bins, based on the posterior

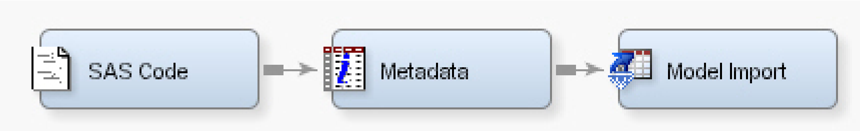
probabilities of the event level and the number of buckets. The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are

- Percentage of Events — for categorical target variables
- Number of Events — for categorical target variables
- Cumulative Percentage of Events — for categorical target variables
- Expected Profit — for categorical target variables
- Report Variables — for categorical target variables
- Mean for Predicted — for interval target variables
- Max. for Predicted — for interval target variables
- Min. for Predicted — for interval target variables

Model Import Node Examples

Example 1: Creating the Score Data Set in the SAS Code Node

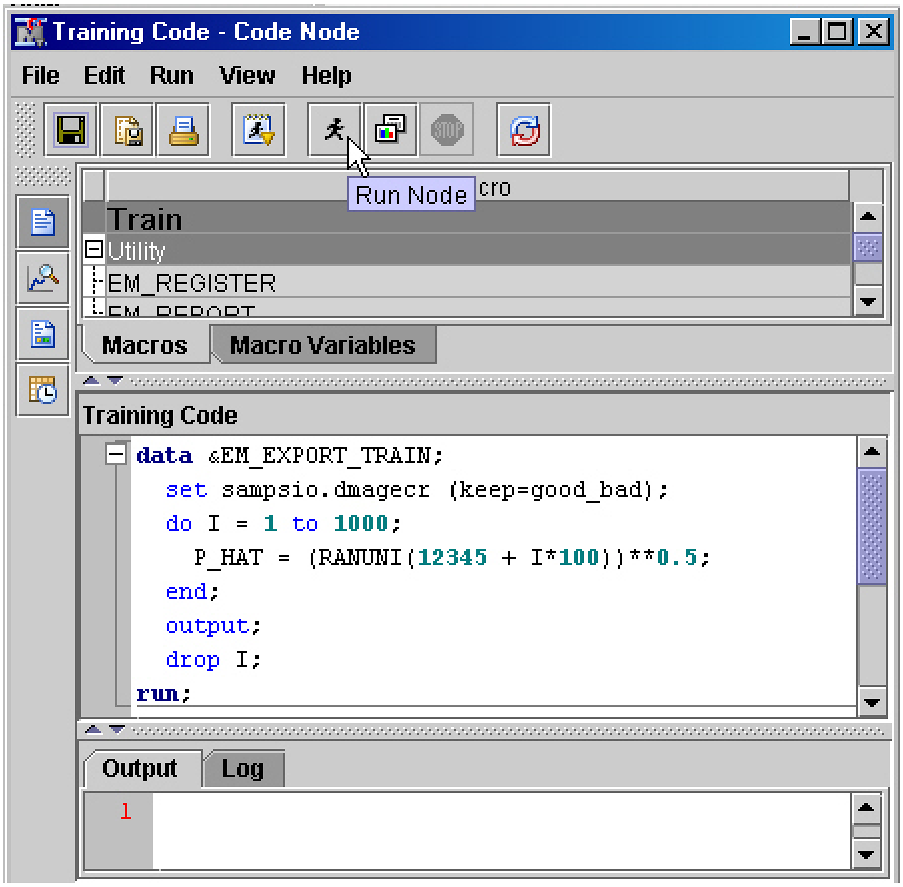
1. Create a new process flow diagram. Drag a SAS Code node from the Utility tab of the node toolbar onto the diagram workspace. Drag a Metadata node from the Utility tab, and a Model Import from the Model tab. Connect the nodes as shown below:



2. Select the SAS Code node and select the ... button to the right of the Code Editor property. Copy and paste the following on the SAS Code editor:

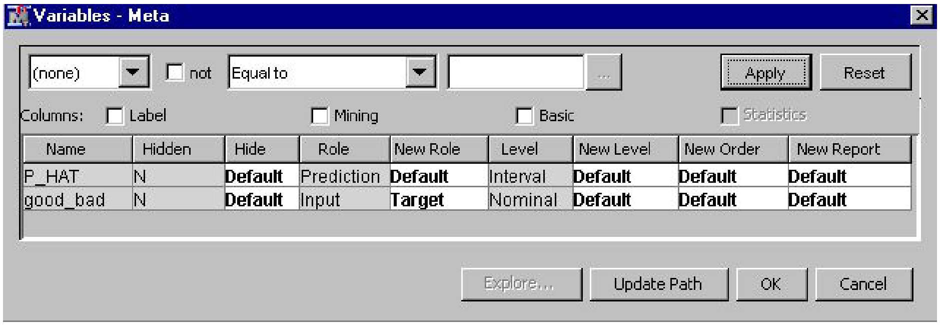
```
data &EM_EXPORT_TRAIN;
  set sampsio.dmagecr (keep=good_bad);
  do I = 1 to 1000;
    P_HAT = (RANUNI(12345 + I*100))**0.5;
  end;
  output;
  drop I;
run;
```

3. Click the **Run Node icon** on the SAS Code editor window.



```
data &EM_EXPORT_TRAIN;
  set sampsio.dmagecr (keep=good_bad);
  do I = 1 to 1000;
    P_HAT = (RANUNI(12345 + I*100))**0.5;
  end;
  output;
  drop I;
run;
```

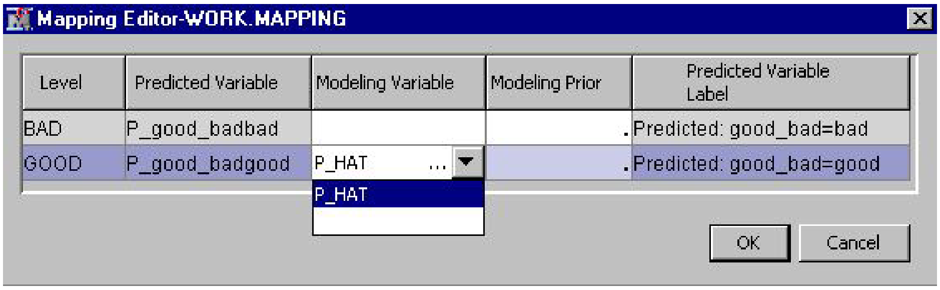
4. Close the SAS Code Node window. In the Metadata node's Variables properties, click on the **...** next to the Train property to open the Variables-Meta window. For the good_bad variable, set New Role to **Target**. Click **OK**.



Name	Hidden	Hide	Role	New Role	Level	New Level	New Order	New Report
P_HAT	N	Default	Prediction	Default	Interval	Default	Default	Default
good_bad	N	Default	Input	Target	Nominal	Default	Default	Default

5. Run the Metadata node.

6. In the Model Import property sheet, select the **...** button to the right of Mapping Editor. Set the **Modeling Variable** for the level GOOD to **P_HAT**. Click **OK**.



Level	Predicted Variable	Modeling Variable	Modeling Prior	Predicted Variable Label
BAD	P_good_badbad		.	Predicted: good_bad=bad
GOOD	P_good_badgood	P_HAT	.	Predicted: good_bad=good

7. Run the Model Import node.

8. From the Results window, examine the Output window. The Output window contains variable summary, model events, decision matrix, predicted and decision variables, fit statistics, classification table, event classification table, assessment score rankings and assessment score distribution.

Example 2: Defining the Scored Data Set in the Input Data Source Node

1. In the Metadata node's Variables properties, click on the ... next to the Train property to open the Variables-Meta window. For the good_bad variable, set New Role to **Target** and for PHAT, set New Role to **Prediction**. Click **OK**.

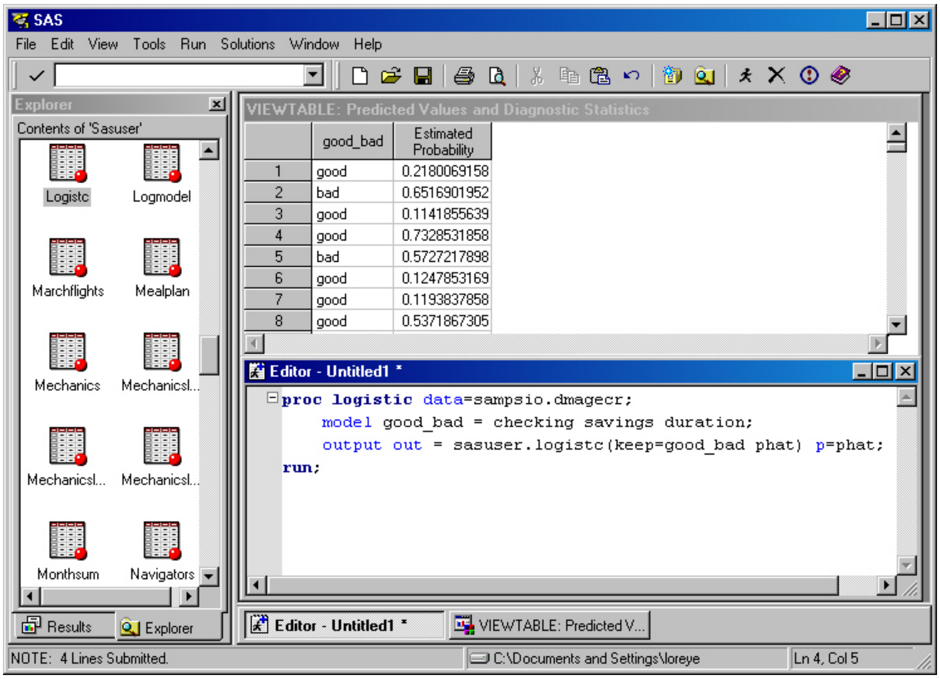
2. Run the Metadata node.

3. In the Model Import property sheet, click the button next to the Mapping Editor property. Set the **Modeling Variable** for the level GOOD to **P_HAT**.

4. Run the Model Import node.

5. Run the code below using SAS 9.2. The code will generate a SAS data set Sasuser.Logistc.

```
proc logistic data=sampsio.dmagecr;
    model good_bad = checking savings duration;
    output out = sasuser.logistc(keep=good_bad phat) p=phat;
run;
```



	good_bad	Estimated Probability
1	good	0.2180069158
2	bad	0.6516901952
3	good	0.1141855639
4	good	0.7328531858
5	bad	0.5727217898
6	good	0.1247853169
7	good	0.1193837858
8	good	0.5371867305

```
proc logistic data=sampsio.dmagecr;
    model good_bad = checking savings duration;
    output out = sasuser.logistc(keep=good_bad phat) p=phat;
run;
```

NOTE: 4 Lines Submitted. C:\Documents and Settings\loreye Ln 4, Col 5

6. Copy the Logistc.sas7bdat data set into a LIBNAME directory that you will use in the Enterprise Miner startup code. For example:

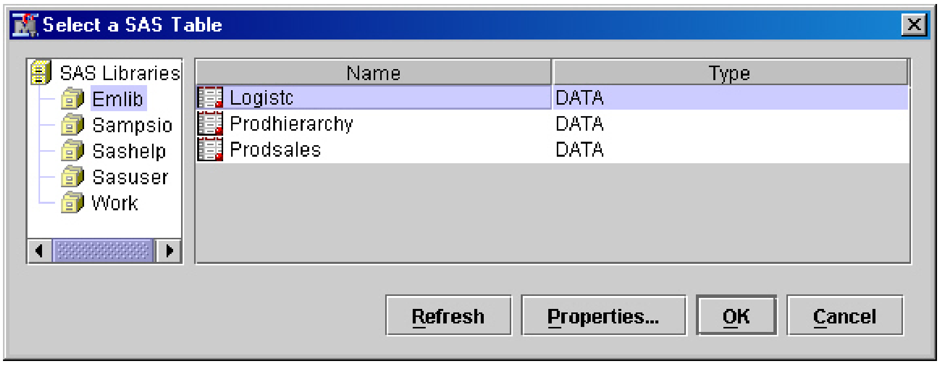
```
libname emlib '\\d17821\public\emexamples\data';
```

7. Add the LIBNAME statement in your Enterprise Miner project startup code and run the startup code.

8. Select **View** ⇨ **Explorere** from the Enterprise Miner window. On the Explorer window, select the Emlib library and verify that Logistc data set is listed.

9. Right-click the Data Sources folder on the Project panel and select **Create Data Source**. The Data Source Wizard Step 1 window opens.

   - Select **SAS Table** as the source and click **Next**. The Data Source Wizard Step 2 window opens.
   - Click **Browse** to open the Select a SAS Table window. Select the Emlib library and Logistc data set then click **OK**. Emlib.Logistc appear on the Table field.

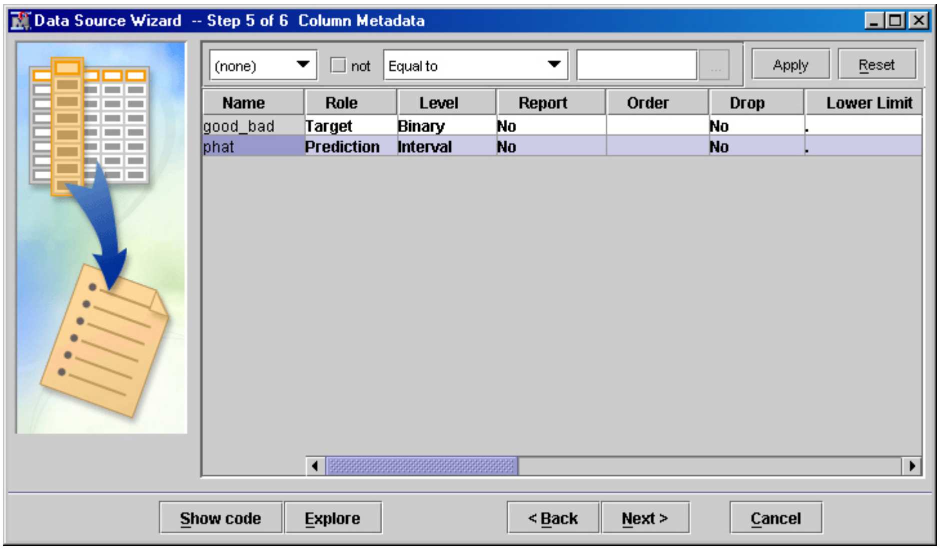


Select a SAS Table

SAS Libraries
- Emlib
- Sampsio
- Sashelp
- Sasuser
- Work

Name	Type
Logistc	DATA
Prodhierarchy	DATA
Prodsales	DATA

   - Click **Next** until the Data Source Wizard Column Metadata window opens. Set the role of good_bad to **Target** and phat to **Prediction**.



Data Source Wizard -- Step 5 of 6 Column Metadata

(none) ☐ not Equal to

Name	Role	Level	Report	Order	Drop	Lower Limit
good_bad	Target	Binary	No		No	.
phat	Prediction	Interval	No		No	.

   - Click **Next** until the Data Source Wizard Data Source Attributes window opens. Specify `Logistic Sample` in the Name field, set the Role to **Train** then click **Finish**.

10. Create a new process flow diagram, then drag the Logistic Sample data source onto the diagram workspace. Right-click the Logistic Sample data source and select **Run**.

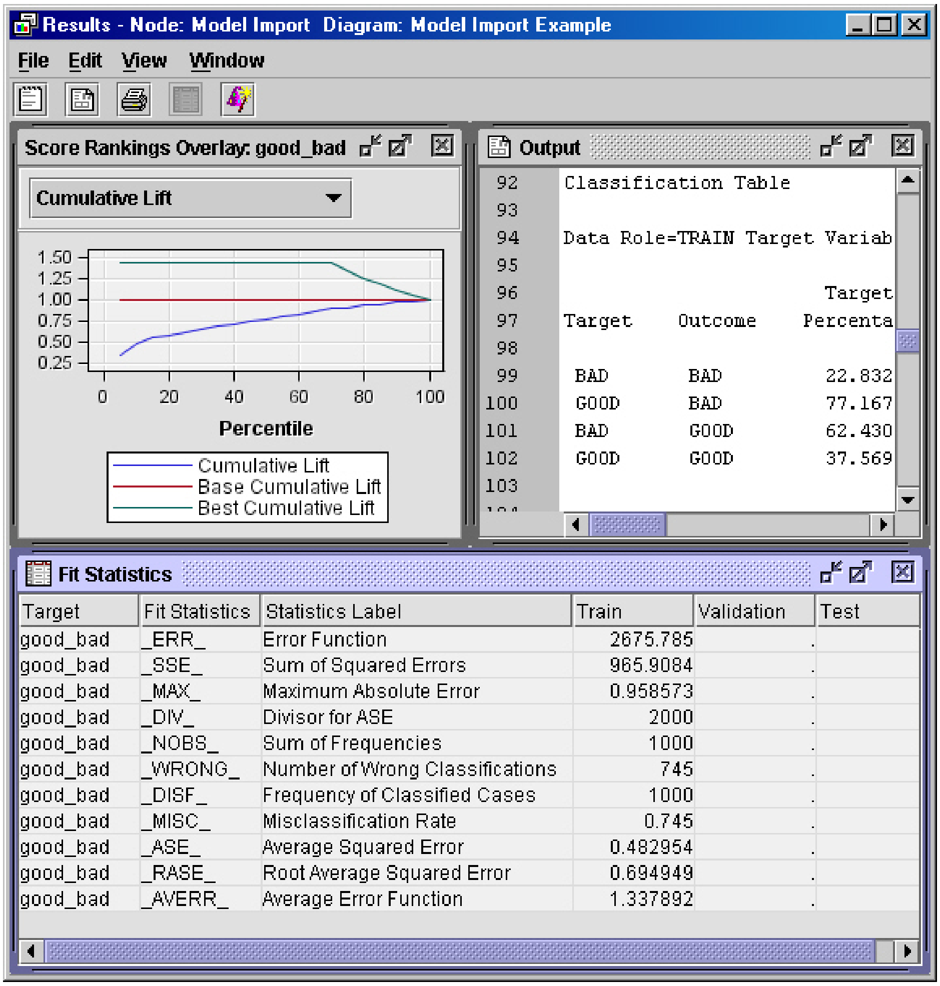
11. Drag a Model Import node from the Model tab of the node toolbar and connect this node to the Logistic Sample data source node.

12. Select the ... button to open the Mapping Editor.

13. Set the **Modeling Variable** for the level GOOD to **phat**.

14. Right-click the Model Import node and select **Run**.

15. Examine the Results window.



```
Classification Table

Data Role=TRAIN Target Variab

                                Target
Target     Outcome      Percenta

 BAD        BAD           22.832
 GOOD       BAD           77.167
 BAD        GOOD          62.430
 GOOD       GOOD          37.569
```

Target	Fit Statistics	Statistics Label	Train	Validation	Test
good_bad	_ERR_	Error Function	2675.785	.	
good_bad	_SSE_	Sum of Squared Errors	965.9084	.	
good_bad	_MAX_	Maximum Absolute Error	0.958573	.	
good_bad	_DIV_	Divisor for ASE	2000	.	
good_bad	_NOBS_	Sum of Frequencies	1000	.	
good_bad	_WRONG_	Number of Wrong Classifications	745	.	
good_bad	_DISF_	Frequency of Classified Cases	1000	.	
good_bad	_MISC_	Misclassification Rate	0.745	.	
good_bad	_ASE_	Average Squared Error	0.482954	.	
good_bad	_RASE_	Root Average Squared Error	0.694949	.	
good_bad	_AVERR_	Average Error Function	1.337892	.	

Chapter 58
Neural Network Node: Reference

Neural Network Node: Reference

Introduction

Before reading this topic, you should be familiar with the Predictive Modeling topic, which contains information that applies to all of the predictive modeling nodes. For general information regarding the use of neural networks, as well as an extensive bibliography, consult the online Neural Network FAQ (Frequently Asked Questions). You can use a web browser to read the FAQ at the URL: **`ftp://ftp.sas.com/pub/neural/FAQ.html`**.

There is an outstanding textbook by Bishop (1995) that is required reading for anyone seriously interested in neural networks.

Artificial neural networks were originally developed by researchers who were trying to mimic the neurophysiology of the human brain. By combining many simple computing elements (neurons or units) into a highly interconnected system, these researchers hoped to produce complex phenomena such as intelligence. In recent years, neural network researchers have incorporated methods from statistics and numerical analysis into their networks. While there is considerable controversy over whether artificial neural networks are really intelligent, there is no doubt that they have developed into very useful statistical models. More specifically, feedforward neural networks are a class of flexible nonlinear regression, discriminant, and data reduction models. By detecting complex nonlinear relationships in data, neural networks can help to make predictions about real-world problems.

Neural networks are especially useful for prediction problems where:

- no mathematical formula is known that relates inputs to outputs.

- prediction is more important than explanation.
- there is a lot of training data.

Common applications of neural networks include credit risk assessment, direct marketing, and sales prediction.

The Neural Network node provides a variety of feedforward networks that are commonly called *backpropagation* or *backprop* networks. This terminology causes much confusion. Strictly speaking, *backpropagation* refers to the method for computing the error gradient for a feedforward network, a straightforward application of the chain rule of elementary calculus. By extension, *backprop* refers to various training methods that use backpropagation to compute the gradient. By further extension, a *backprop* network is a feedforward network trained by any of various gradient-descent techniques. *Standard backprop* is a euphemism for the *generalized delta rule*, the training technique that was popularized by Rumelhart, Hinton, and Williams in 1986 and which remains the most widely used supervised training method for feedforward neural nets. Standard backprop is also one of the most difficult to use, tedious, and unreliable training methods. Unlike the other training methods in the Neural Network node, standard backprop comes in two varieties.

- Batch backprop, like conventional optimization techniques, reads the entire data set, updates the weights, reads the entire data set, updates the weights, and so on.
- Incremental backprop reads one case, updates the weights, reads one case, updates the weights, and so on.

Batch backprop is one of the slowest training methods. Although the Neural Network node provides an option for batch backprop, it is recommended that you never use it for serious work. Incremental backprop can be useful for very large, redundant data sets, if you are skilled at setting the learning rate and momentum appropriately.

Fortunately, there is no need to suffer through the slow convergence and the tedious tuning of standard backprop. Much of the neural network research literature is devoted to attempts to speed up backprop. Most of these methods are inconsequential; two that are effective are Quickprop and RPROP, both of which are available in the Neural Network node. In addition, the Neural Network node provides a variety of conventional methods for nonlinear optimization that have been developed by numerical analysts over the past several centuries and that are usually faster and more reliable than the algorithms from the neural network literature.

Up until the early 1990s, neural networks were often viewed as alternatives to statistical methods. Some researchers made outlandish claims that neural networks could be used to analyze data with no expertise required on the part of the analyst. These unjustifiable claims, combined with the unreliability of early algorithms such as standard backprop, led to a backlash in which many people, especially statisticians, dismissed neural networks as entirely worthless for data analysis. But in recent years, it has been widely recognized that many kinds of neural networks are statistical methods, and that when neural networks are trained via reliable methods such as conventional optimization techniques or Bayesian learning, the results are just as valid as those obtained by many nonlinear or nonparametric statistical methods.

Neural networks, like other statistical methods, cannot magically create information out of nothing — the rule "garbage in, garbage out" still applies. The predictive ability of a neural network depends in part on the quality of the training data. It is also important for the analyst to have some knowledge of the subject matter, especially for selecting inputs and choosing an appropriate error function. Experienced neural network users typically try several architectures to determine the best network for a specific data set. The design process and the training process are both iterative.

Overview of Feedforward Neural Networks

Units and Connections

A neural network consists of units (neurons) and connections between those units. There are three kinds of units.

- Input Units obtain the values of input variables and optionally standardize those values.
- Hidden Units perform internal computations, providing the nonlinearity that makes neural networks powerful.
- Output Units compute predicted values and compare those predicted values with the values of the target variables.

Units pass information to other units through connections. Connections are directional and indicate the flow of computation within the network. Connections cannot form loops, since the Neural Network node allows only feedforward networks.

The following restrictions apply to feedforward networks:

- Input units can be connected to hidden units or to output units.
- Hidden units can be connected to other hidden units or to output units.
- Output units cannot be connected to other units.
- Due to limitations in the SAS supervisor, the total number of connections in a network cannot exceed roughly 32,000.

Predicted Values and Error Functions

Each unit produces a single computed value. For input and hidden units, this computed value is passed along the connections to other hidden or output units. For output units, the computed value is what statisticians call a predicted value. The predicted value is compared with the target value to compute the error function, which the training methods attempt to minimize.

Weight, Bias, and Altitude

Most connections in a network have an associated numeric value called a weight or parameter estimate. The training methods attempt to minimize the error function by iteratively adjusting the values of the weights. Most units also have one or two associated numeric values called the bias and altitude, which are also estimated parameters adjusted by the training methods.

Combination Functions

Hidden and output units use two functions to produce their computed values. First, all the computed values from previous units feeding into the given unit are combined into a single value using a combination function. The combination function uses the weights, bias, and altitude. Two general kinds of combination function are commonly used.

- Linear Combination Functions — compute a linear combination of the weights and the values feeding into the unit and then add the bias value (the bias acts like an intercept).
- Radial Combination Functions — compute the squared Euclidean distance between the vector of weights and the vector of values feeding into the unit and then multiply by the squared bias value (the bias acts as a scale factor or inverse width).

There is also an additive combination function that uses no weights or bias. For more details, see “Pairing Architectures with Combination and Activation Functions” on page 874 .

Activation Functions

The value produced by the combination function is transformed by an *activation function*, which involves no weights or other estimated parameters. Several general kinds of activation functions are commonly used.

- Identity Function is also called a linear function. It does not change the value of the argument, and its range is potentially is unbounded.
- Sigmoid Functions are S-shaped functions such as the logistic and hyperbolic tangent functions that produce bounded values within a range of 0 to 1 or -1 to 1.
- Softmax Function is called a multiple logistic function by statisticians and is a generalization of the logistic function that affects several units together, forcing the sum of their values to be one.
- Value Functions are bounded bell-shaped functions such as the Gaussian function.
- Exponential and Reciprocal Functions are bounded below by zero but unbounded above.

For more details, see “Pairing Architectures with Combination and Activation Functions” on page 874 .

Network Layers

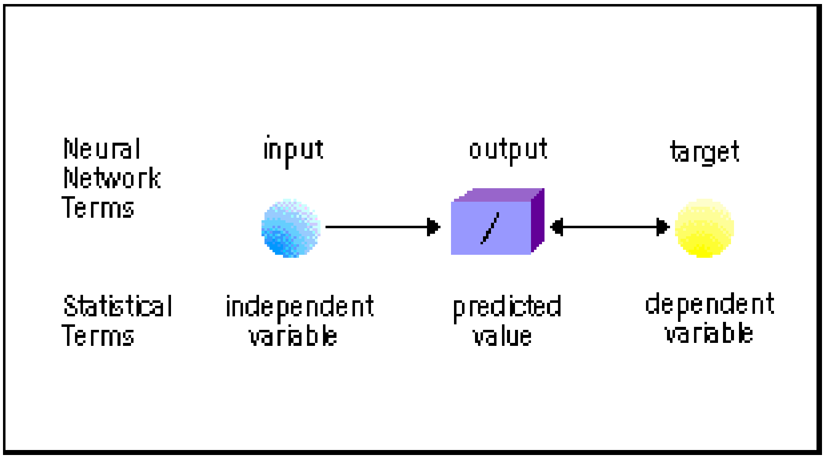
A network may contain many units, perhaps several hundred. The units are grouped into layers to make them easier to manage. SAS Enterprise Miner supports multiple input layers, a hidden layer, and multiple output layers. In the Neural Network node, when you connect two layers, every unit in the first layer is connected to every unit in the second layer.

All the units in a given layer share certain characteristics. For example, all the input units in a given layer have the same measurement level and the same method of standardization. All the units in a given hidden layer have the same combination function and the same activation function. All the units in a given output layer have the same combination function, activation function, and error function.

Simple Neural Networks

Overview

The simplest neural network has a single input unit (independent variable), a single target (dependent variable), and a single output unit (predicted values).

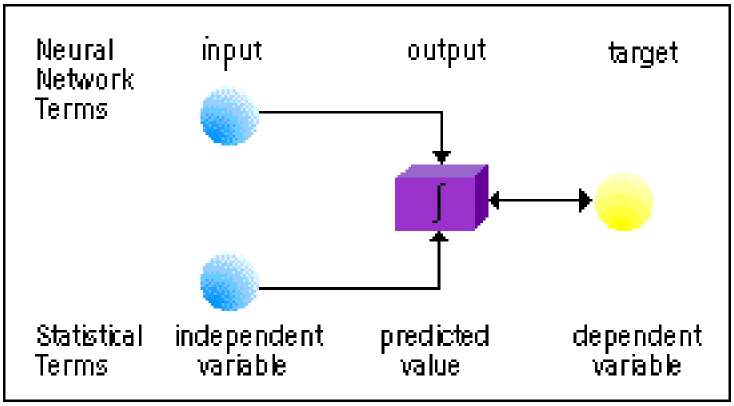


The slash inside the box represents a linear (or identity) output activation function. In statistical terms, this network is a simple linear regression model. If the output activation function were a logistic function, then this network would be a logistic regression model.

Perceptrons

One of the earliest neural network architectures was the *perceptron*, which is a type of linear discriminant model. A perceptron uses a linear combination of inputs for the combination function. Originally, perceptrons used a threshold (Heaviside) activation function, but training a network with threshold activation functions is computationally difficult. In current practice, the activation function is almost always a logistic function, which makes a perceptron equivalent in functional form to a logistic regression model.

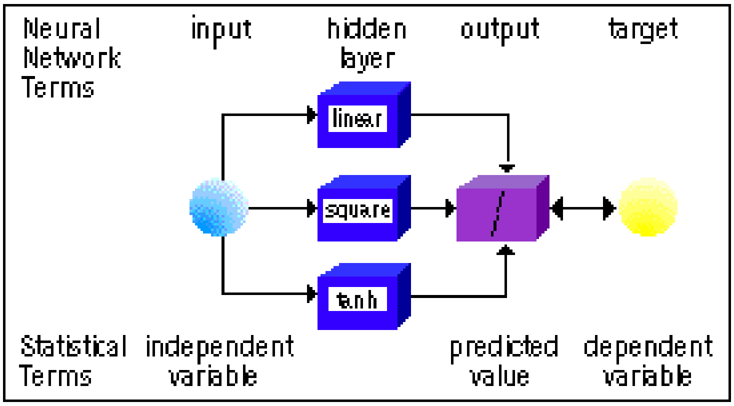
As an example, a perceptron might have two inputs and a single output.



In neural network terms, the diagram shows two inputs connected to a single output with a logistic activation function (represented by the sigmoid curve in the box). In statistical terms, this diagram shows a logistic regression model with two independent variables and one dependent variable.

Hidden Layers

Neural networks may apply additional transformations via a hidden layer. Typically, each input unit is connected to each unit in the hidden layer, and each hidden unit is connected to each output unit. The hidden units combine the input values and apply an activation function, which may be nonlinear. Then the values that were computed by the hidden units are combined at the output units, where an additional (possibly different) activation function is applied. If such a network uses linear combination functions and sigmoid activation functions, it is called a multilayer perceptron or MLP. It is also possible to use other combination functions and other activation functions to connect layers in many other ways. A network with three hidden units with different activation functions is shown in the following diagram.



Multilayer Perceptrons (MLPs)

The most popular form of neural network architecture is the *multilayer perceptron* (MLP), which is the default architecture in the Neural Network node. A multilayer perceptron

- has any number of inputs.
- has a hidden layer with any number of units.
- uses linear combination functions in the hidden and output layers.
- uses sigmoid activation functions in the hidden layers.
- has any number of outputs with any activation function.
- has connections between the input layer and the first hidden layer, between the hidden layers, and between the last hidden layer and the output layer.

The Neural Network node supports many variations of this general form. For example, you can add direct connections between the inputs and outputs, or you can cut the default connections and add new connections of your own.

Given enough data, enough hidden units, and enough training time, an MLP with one hidden layer can learn to approximate virtually any function to any degree of accuracy. (A statistical analogy is approximating a function with nth order polynomials.) For this reason, MLPs are known as universal approximators, and can be used when you have little prior knowledge of the relationship between inputs and targets.

Radial Basis Function (RBF) Networks

A Radial Basis Function Network

- has any number of inputs.
- typically has a hidden layer with any number of units.
- uses radial combination functions in the hidden layer, based on the squared Euclidean distance between the input vector and the weight vector.
- typically uses exponential or softmax activation functions in the hidden layer, in which case the network is a Gaussian RBF network.
- uses linear combination functions in the output layer.

- has any number of outputs with any activation function.
- has connections between the input layer and the hidden layer, and between the hidden layer and the output layer.

Local Processing Networks

MLPs are said to be *distributed-processing* networks because the effect of a hidden unit can be distributed over the entire input space. On the other hand, Gaussian RBF networks are said to be *local-processing* networks because the effect of a hidden unit is usually concentrated in a local area centered at the weight vector.

To use an RBF network in the Neural Network node, you can specify one of the built-in architectures on page 872 .

Width and Altitude

The hidden layer of an RBF network does not have anything that's exactly the same as the bias term in an MLP. Instead, RBFs have a width associated with each hidden unit or with the entire hidden layer. Instead of adding the width in the combination function like a bias, you divide the Euclidean distance by the width. Since dividing by a width of zero would produce undefined results, the Neural Network node has a different parameterization using the square root of the reciprocal of the width. In the names of the weights, the square root of the reciprocal of the width is called the "bias" for lack of an appropriate term, although it does not have the usual interpretation of a bias.

Some radial combination functions have another parameter called the altitude of the unit. The altitude is the maximum height of the Gaussian curve above the horizontal axis.

Ordinary RBF and Normalized RBF

There are two distinct types of Gaussian RBF architectures. The first type, the ordinary RBF (ORBF) network, uses the exponential activation function, so the activation of the unit is a Gaussian "bump" as a function of the inputs. The second type, the normalized RBF (NRBF) network, uses the softmax activation function, so the activations of all the hidden units are normalized to sum to one. While the distinction between these two types of Gaussian RBF architectures is sometimes mentioned in the NN literature, its importance has rarely been appreciated except by Tao (1993) and Werntges (1993).

The output activation function in RBF networks is customarily the identity. Using an identity output activation function is a computational convenience in training, but it is possible and often desirable to use other output activation functions just as you would in an MLP. The Neural Network node sets the default output activation function for RBF networks the same way it does for MLPs.

For further discussion of types of RBF networks and comparisons between RBF networks and MLPs, see the “List of Built-In Architectures” on page 872 .

Error Functions

A network is trained by minimizing an error function (also called an estimation criterion or Lyapunov function). Most error functions are based on the maximum likelihood principle, although computationally it is the negative log likelihood that is minimized. The likelihood is based on a family of error (noise) distributions for which the resulting estimator has various optimality properties. M estimators are formally similar to maximum likelihood estimators, but for certain kinds called redescending M estimators, no proper error distribution exists.

Some of the more commonly used error functions are

- *Normal distribution* — also called the least-squares or mean-squared-error criterion. Suitable for unbounded interval targets with constant conditional variance, no outliers, and a symmetric distribution. Can also be used for categorical targets with outliers.

- *Huber M-estimator* — Suitable for unbounded interval targets with outliers or with a moderate degree of inequality of the conditional variance but a symmetric distribution. Can also be used for categorical targets when you want to predict the mode rather than the posterior probability.
- *Redescending M estimators* — Suitable for unbounded interval targets with severe outliers. Can also be used for predicting one mode of a multimodal distribution. Includes biweight and wave estimators.
- *Gamma distribution* — Suitable for skewed, positive interval targets where the conditional standard deviation is proportional to the conditional mean.
- *Poisson distribution* — Suitable for skewed, nonnegative interval targets, especially counts of rare events, where the conditional variance is proportional to the conditional mean.
- *Bernoulli distribution* — Suitable for a target that takes only the values zero and one. Same as a binomial distribution with one trial.
- *Entropy* — Cross or relative entropy for independent interval targets with values between zero and one inclusive.
- *Multiple Bernoulli* — Suitable for categorical (nominal or ordinal) targets.
- *Multiple Entropy* — Cross or relative entropy for interval targets that sum to one and have values between zero and one inclusive. Also called Kullback-Leibler divergence.

For a categorical target variable, the default error function is multiple Bernoulli. For interval targets, the default error function is normal.

Initialization

The results of training a neural network with hidden units may depend on the initial values of the weights. For MLPs, there is no known rational method for computing initial weights, so random initial weights are used. The Neural Network node supplies pseudo-random initial weights.

By default, all output connection weights are initialized to zero, and output biases are initialized by applying the inverse of the activation function to the mean of the target variable.

For units with a linear combination function, the random initial weights are adjusted by dividing by the square root of the fan-in of the unit (the number of connections coming into the unit).

Preliminary Training

Local minima are a common problem with nonlinear networks. The simplest way to deal with local minima is to train the network for a few iterations from each of a large number (perhaps 10, 100, or 1000) of random initializations. The Neural Network node selects the best estimates obtained during these preliminary runs to be used for subsequent training.

Training Techniques

The Neural Network node provides a wide variety of training techniques, including both conventional optimization techniques and techniques from the neural network literature. The conventional optimization techniques are well-proven algorithms that are described in detail in the documentation for the NLP Procedure in the SAS/OR product.

The most popular conventional optimization techniques include the following:

- Default — The default method selects the best training technique for you based on the weights that are applied at execution.
- Trust-Region — The Trust-Region method is recommended for small and medium optimization problems with up to 40 parameters.
- Levenberg-Marquardt — is very fast and reliable for small least-squares networks, but requires a large amount of memory (quadratic in the number of estimated parameters).
- Quasi-Newton techniques — are good for medium-sized networks. They require quadratic memory, but only about half as much as Levenberg-Marquardt. Quasi-Newton techniques usually require more iterations than Levenberg-Marquardt but each iteration requires less floating-point computation.
- Conjugate gradient techniques — are good for large networks when there is not enough memory for the above techniques — memory requirements are only linear. They usually require more iterations than either of the above techniques, but each iteration requires less floating-point computation. Conjugate gradient techniques may be a good choice if you have a very fast disk drive.

The Neural Network node also provides three training techniques popular in the neural network literature, all of which have linear memory requirements.

- Standard batch backprop — is the most popular training method of all, but it is slow, unreliable, and requires the user to tune the learning rate manually, which can be a tedious process.
- Quickprop — usually requires more iterations than conjugate gradient methods, but each iteration is very fast. Quickprop is a Newton-like method but uses a diagonal approximation to the Hessian matrix. It is much faster and more reliable than standard batch backprop and requires little tuning.
- RPROP — usually requires more iterations than conjugate gradient methods, but each iteration is very fast. RPROP uses a separate learning rate for each weight, and adapts all the learning rates during training. RPROP is the most stable of all the "prop" techniques and rarely requires any tuning.

For all of the conventional optimization techniques, the objective function decreases on each iteration until a local minimum is reached, at which point the algorithm terminates. For the "prop" techniques, the objective function may go down and up repeatedly during training. For standard backprop, if you specify a learning rate that is too high, the weights will diverge and the objective function will increase without limit.

Scoring

After developing a trained network that meets your requirements, you can use the network to make predictions for various data sets. The Neural Network node allows you to score the training, validation, test, and score data sets in conjunction with training.

Preparing the Data

Preprocess the Data

Before you train a neural network, you may want to consider the following data mining tasks:

- Sample the Input Data — The Sampling node enables you to extract a sample of your input data. Sampling is recommended for extremely large data bases, because it can tremendously decrease training time. If the sample is sufficiently representative, then relationships found in the sample can be expected to generalize to the complete data set.

- Create Partitioned Data Sets — The Data Partition node enables you to split the sample into training, validation, and test data sets. The training data set is used to learn the network weights. The validation data set is used to choose among various network architectures. The validation data set is also used by the Model Comparison node for model assessment. The test set is used to obtain a final, unbiased estimate of generalization error.
- Use Only the Important Variables — When your data set has a large number of inputs, it is tempting to use most or all of the inputs. However, it is often better to use only a small number of important inputs, which can greatly reduce the time required to train the network, as well as improving the prediction results. Twenty to thirty inputs may be better than three hundred. If you know from your business expertise that an input is not useful in predicting the target, then exclude it from the regression analysis. The Explore window and the Multiplot node enable you to create exploratory plots that can help you identify important inputs (predictors). You can also use the Variable Selection node to remove unpromising inputs.

  When many inputs are available, the choice of network architecture is especially important. For example, MLPs tend to be better at ignoring irrelevant inputs than are some RBF networks. Having many inputs also reduces the number of hidden units you can use, since the number of weights connecting an input layer and a hidden layer is equal to the product of the number of units in each. For example, if you have five inputs, using 100 hidden units may be quite practical. But if you have 100 inputs and try to use 100 hidden units, you will have over 10,000 weights in the network and training will take a very long time.
- Transform Data and Filter Outliers — Since neural networks can learn nonlinear functions, they are not as sensitive to transformations of the input variables as are regression models. Nevertheless, the use of appropriate input transformations can improve generalization and speed up training. Outliers in the input variables are of special concern because they can have undue influence on predictions, just like high-leverage points in linear regression.

  On the other hand, transformations of the target variables are just as important for neural networks as for regression. For example, if you are using least-squares estimation, transforming the target variables to have conditional normal distributions with constant variance can improve generalization. More important, transformation of targets changes the relative importance of errors depending on the target value.

  For example, suppose you are trying to predict the price of some commodity. If the price of the commodity is 10 (in whatever currency unit) and the output of the net is 5 or 15, yielding a difference of 5, that is a huge error. If the price of the commodity is 1000 and the output of the net is 995 or 1005, yielding the same difference of 5, that is a tiny error. You do not want the network to treat those two differences as equally important. By taking logarithms, you are effectively measuring errors in terms of ratios rather than differences, since a difference between two logs corresponds to the ratio of the original values. This has approximately the same effect as looking at percentage differences, abs(target-output)/target or abs(target-output)/output, rather than simple differences.

  The Transform Variables node enables you to apply several transformations to a variable, such as log, exponential, square root, inverse, and square. The node also includes three power transformations (maximize normality, maximize correlation with the target, and equalize spread with target levels) plus an optimal binning for relationship to target transformation.

- Impute Missing Values — The Neural Network node omits cases from training if any of the inputs are missing or if all of the targets are missing. Hence you may want to replace missing data with imputed values. The Impute node enables you to replace interval missing values with the input's mean, median, or midrange. You can also use a robust M-estimator to impute missing values. Missing values of a categorical input can be replaced with the mode. The node also includes a tree-based imputation method for replacing missing values. In addition to imputing missing values, if an input has many missing values (more than five to twenty times the number of hidden units), it is often useful to generate a dummy variable that has the value 1 when the corresponding input variable is missing, and the value 0 otherwise. Then both the original input variable and the dummy variable can be used as inputs to the network.
- Use Other Modeling Nodes — For some problems, a neural network may not be the best analytical tool. If the input-output function is approximately linear and the data are noisy, linear regression may produce better generalization. If few of the input variables are relevant or you want an interpretable prediction rule, tree-based models may be preferable. Cross-model assessment can be performed with the Model Comparison node to help you choose the best model for scoring new data.

Specify the Input Data Sets

Input data sets are supplied to the Neural Network node from an Input Data node. When you are ready to train the neural network, you select the training, validation, and test data sets in the Imported Data property of the Neural Network node. You can also set the score data set, which is not required to contain the target variable.

The purposes of the training, validation, and test sets are described by Bishop (1995) in the following quote.

"Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the hold out method. Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set."

Specify the Inputs and the Targets

Variables are assigned roles and measurement levels in the Input Data node. For neural networks, variable roles are:

- **input** — for input variables
- **target** — for target variables
- **freq** — for a frequency variable
- **ID** — for variables that are not used in modeling but are retained in the scored data sets
- **rejected** — for variables that are not used in modeling and are not wanted in scored data sets

The Neural Network node requires one or more input variables and one or more target variables. To change the model role of a variable, use the Variables table in the Input Data node properties or use the Variables table in the Data Source properties.

The inputs and targets can be nominal, ordinal, or interval. However, you may not use both ordinal inputs and ordinal targets in the same network.

For a categorical (that is, nominal or ordinal) input with C categories, the Neural Network node automatically generates C — 1 dummy variables. For nominal inputs, deviation coding is used. For each case in the ith category for i< C, the ith dummy variable is set to 1, and all other dummy variables are set to 0. For i= C, all dummy variables are set to - 1. For example, if there is a nominal input called SEX with values Male, Female, Both, and Neither. The dummy variables would have the values shown in the following table, where it is assumed that the categories are ordered alphabetically.

	Dummy Variables		
SEX	**SEXBOTH**	**SEXFEMA**	**SEXMALE**
Male	0	0	1
Female	0	1	0
Both	1	0	0
Neither	-1	—1	-1

For ordinal inputs, bathtub coding is used. For each case in the i^{th} category, the j^{th} dummy variable is set to

$$\sqrt{1.5C/(C^2-1)}$$

when i > j, or to

$$-\sqrt{1.5C/(C^2-1)}$$

otherwise. If a population has uniformly distributed categories, then the total variance of all the dummy variables corresponding to a given input using bathtub coding is one. For example, if there is an ordinal variable called LIBIDO with values None, Low, Medium, and High in that order, the dummy variables would have the values shown in the following table:

	Dummy Variables		
LIBIDO	**LIBINONE**	**LIBILOW**	**LIBIMED**
None	-0.63246	-0.63246	-0.63246
Low	0.63246	-0.63246	-0.63246
Medium	0.63246	0.63246	-0.63246
High	0.63246	0.63246	0.63246

For an ordinal input, the weights from the dummy input units are constrained to be nonnegative. This constraint forces the relationship between the input and other units to which it is connected directly and only directly to be monotonic when other inputs are held constant. For example, when there are no hidden units, the Neural Network node does monotone regression for ordinal inputs. However, if there are hidden units, the relationship between ordinal inputs and predicted values may not be monotonic.

For a categorical target with C class, the Neural Network node automatically generates C dummy variables using GLM coding. For each case in the ith class, the ith dummy variable is set to 1, and all other dummy variables are set to 0. Hence the predicted value for a dummy variable is equal to the posterior probability of the corresponding class. For example, if there is a nominal target called SEX with values Male, Female, Both, and Neither, the dummy variables would have the values shown in the following table:

	Dummy Variables			
SEX	**SEXBOT**	**SEXFEM**	**SEXMAL**	**SEXNEI**
Male	0	0	1	0
Female	0	1	0	0
Both	1	0	0	0
Neither	0	0	0	1

Note: Avoid using categorical variable with many levels. For example, the variable ZIPCODE may have 29,000 categories, and if it is designated as a categorical variable, then 28,999 units would be created.

Note: For ordinal inputs, it is crucial to specify the correct order. In the Input Data node, you must not only specify whether the order of the categories is determined by internal values, formatted values, or first appearance in the training set, but also specify ascending or descending order to insure the correct direction of the relationship between the input and target. If the direction is set incorrectly, all or most of the weights for that input will be zero.

For information on how categories are defined and ordered, see Categorical Variables in the Predictive Modeling topic.

Defining the Network

Overview

In designing the Neural Network, there are many specifications that can be made. You can

- Accept the default network architecture settings
- Use the basic templates to configure the network

The sections that follow describe how to configure the Built-in Architectures network component.

Definitions for Built-In Architectures

The following definitions apply to the formulas for the built-in architectures:

a_j

The height (altitude) of the jth hidden unit.

b_j

The bias or inverse width of the jth hidden unit.

b

A common bias or inverse width shared by all hidden units.

f

The fan-in of each hidden unit. The fan-in of a unit is the number of other units feeding into that unit, exclusing biases and altitudes.

h_j

The activation of the jth hidden unit.

w_{ij}

The weight connecting the ith input to the jth hidden unit.

w_I

The common weight for the ith input shared by all hidden units.

x_I

The ith input.

Softmax Function for Built-In Architectures

Given net inputs gj computed by the combination function for each hidden unit, the softmax function is

$$soft\max(g_i) = \frac{\exp(g_i)}{\sum_k \exp(g_k)}$$

List of Built-In Architectures

The Neural Network node provides the following built-in architectures:

- GLIM — Generalized Linear Model, which has no hidden layers. By default, the inputs are not standardized.
- MLP — Multilayer Perceptron, which has one hidden layer by default, with a linear combination function and tanh activation function. The formula for the activation of a hidden unit is:

$$h_j = \tanh\left\{b_j + \sum_i w_{ij} x_i\right\}$$

- ORBFEQ — Ordinary Radial Basis Function Network with Equal Widths, which has one hidden layer with an EQRadial combination function and exponential activation function. The formula for the activation of a hidden unit is:

$$h_j = \exp\left\{ -b^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- ORBFUN — Ordinary Radial Basis Function Network with Unequal Widths, which has one hidden layer with an EHRadial combination function and exponential activation function. The formula for the activation of a hidden unit is:

$$h_j = \exp\left\{ -{b_j}^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- NRBFEQ — Normalized Radial Basis Function Network with Equal Widths and Heights, which has one hidden layer with an EQRadial combination function and softmax activation function. The formula for the activation of a hidden unit is:

$$h_j = \text{softmax}\left\{ -b^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- NRBFEH — Normalized Radial Basis Function Network with Equal Heights and Unequal Widths, which has one hidden layer with an EHRadial combination function and softmax activation function. The formula for the activation of a hidden unit is:

$$h_j = \text{softmax}\left\{ -{b_j}^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- NRBFEW — Normalized Radial Basis Function Network with Equal Widths and Unequal Heights, which has one hidden layer with an EWRadial combination function and softmax activation function.The formula for the activation of a hidden unit is:

$$h_j = \text{softmax}\left\{ f \log(a_j) - b^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- NRBFEV — Normalized Radial Basis Function Network with Equal Volumes, which has one hidden layer with an EVRadial combination function and softmax activation function.The formula for the activation of a hidden unit is:

$$h_j = \text{softmax}\left\{ f \log(b_j) - {b_j}^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

- NRBFUN — Normalized Radial Basis Function Network with Unequal Widths and Heights, which has one hidden layer with an XRadial combination function and softmax activation function.The formula for the activation of a hidden unit is:

$$h_j = \text{softmax}\left\{ f \log(a_j) - {b_j}^2 \sum_i (w_{ij} - x_i)^2 \right\}$$

Pairing Architectures with Combination and Activation Functions

Radial combination functions with altitudes are not useful with ORBF architectures, since the altitude would be redundant with the hidden-to-output weights. Altitudes are useful with NRBF architectures, since the normalization performed by the softmax function intervenes between the altitudes and hidden-to-output weights. Hence there is a wider variety of NRBF architectures than ORBF architectures.

The practical differences among these architectures can be understood in terms of the isoactivation contours of the hidden units. An isoactivation contour is the set of all points in the input space yielding a specified value for the activation function of a given hidden unit.

Since an MLP uses linear combination functions, the set of all points in the space having a given value of the activation function is a hyperplane. The hyperplanes corresponding to different activation levels are parallel to each other (the hyperplanes for different units are not parallel in general). These parallel hyperplanes are the isoactivation contours.

The ORBF architectures use radial combination functions and the exponential activation function. Radial combination functions are based on the Euclidean distance between the vector of inputs to the unit and the vector of corresponding weights. Thus, the isoactivation contours for ORBF networks are concentric hyperspheres. The output of an ORBF network consists of a number of superimposed bumps, hence the output is quite bumpy unless many hidden units are used. Thus an ORBF network with a small number of few hidden units is incapable of fitting many simple, smooth functions, and should rarely be used.

The NRBF architectures also use radial combination functions but the activation function is softmax, which forces the sum of the activations for the hidden layer to equal one. Hence the activation of each hidden unit depends on the activations of potentially all other hidden units in the same layer, although near-by units will have more effect than those far away. Because the activation of one hidden unit depends on other hidden units, the isoactivation contours of a NRBF network are considerably more complicated than those of ORBF networks or MLPs.

Consider the case of an NRBF network with only two hidden units. If the hidden units have equal widths, the isoactivation contours are parallel hyperplanes; in fact, this network is equivalent to an MLP with one logistic hidden unit. If the hidden units have unequal widths, the isoactivation contours are concentric hyperspheres; such a network is similar to an ORBF network with one Gaussian hidden unit.

If there are more than two hidden units in an NRBF network, the isoactivation contours have no such simple characterization. If the RBF widths are very small, the isoactivation contours are approximately piecewise linear for RBF units with equal widths, and approximately piecewise spherical for RBF units with unequal widths. The larger the widths, the smoother the isoactivation contours where the pieces join. As Shorten and Murray-Smith (1996) point out, the activation is not necessarily a monotone function of distance from the center when unequal widths are used.

In a NRBF network, each output unit computes a nonnegative weighted average of the hidden-to-output weights. Hence the output values must lie within the range of the hidden-to-output weights. Therefore, if the hidden-to-output weights are within a reasonable range (such as the range of the target values), you can be sure that the outputs will be within that same range for all possible inputs, even when the net is extrapolating. No comparably useful bound exists for the output of an ORBF network or MLP, since the output can potentially be as large as the sum of the absolute values of all the hidden-to-output weights.

If you extrapolate far enough in a Gaussian ORBF network with an identity output activation function, the activation of every hidden unit will approach zero, hence the extrapolated output of the network will equal the output bias. If you extrapolate far enough in an NRBF network, one hidden unit will come to dominate the output. Hence if you want the network to extrapolate different values in a different directions, an NRBF should be used instead of an ORBF.

The NRBFEQ architecture is a smoothed variant of the learning vector quantization (LVQ) and counterpropagation architectures. In LVQ and counterpropagation, the hidden units are often called codebook vectors. LVQ amounts to nearest-neighbor classification on the codebook vectors, while counterpropagation is nearest-neighbor regression on the codebook vectors. The NRBFEQ architecture uses not just the single nearest neighbor, but a weighted average of near neighbors. As the width of the NRBFEQ functions approaches zero, the weights approach one for the nearest neighbor and zero for all other codebook vectors. LVQ and counterpropagation use ad hoc algorithms of uncertain reliability, but standard numerical optimization techniques (not to mention backprop) can be applied with the NRBFEQ architecture.

In a NRBFEQ architecture, if each observation is taken as an RBF center, and if the weights are taken to be the target values, the outputs are simply weighted averages of the target values, and the network is identical to the well-known Nadaraya-Watson kernel regression estimator, which has been reinvented at least twice in the neural net literature. A similar NRBFEQ network used for classification is equivalent to kernel discriminant analysis. Kernels with variable widths are also used for regression in the statistical literature. Such kernel estimators correspond to the NRBFEV architecture, in which the kernel functions have equal volumes but different altitudes. In the neural net literature, variable-width kernels appear always to be of the NRBFEH variety, with equal altitudes but unequal volumes. The analogy with kernel regression would make the NRBFEV architecture the obvious choice, but which of the two architectures works better in practice is an open question.

Objective Functions and Error Functions

Introduction to Objective Functions and Error Functions

The Neural Network node trains a network by minimizing an objective function. The objective function is the sum of a total error function and a penalty function, divided by the total frequency. The total error function is a summation over all the cases in the training data and all the target variables of an individual error function applied to each target value and corresponding output. The penalty function is the product of the weight decay constant and the sum of all network weights other than output biases.

The objective function has several different forms depending on the types of error functions used. Maximum likelihood is the most important of all statistical estimation methods. For computational purposes, it is the negative log likelihood that is minimized, rather than the likelihood that is maximized. For noise distributions in the exponential family, the deviance function can be used instead of the likelihood function. The deviance is the difference between the likelihood for the actual network and the likelihood for a saturated model in which there is one weight for every case in the training set. The deviance cannot be negative and is zero for a perfect fit to noise-free data; these properties provide the deviance with computational advantages, so deviance training is often faster than likelihood training. However, the deviance function cannot be used for noise distributions that are not in the exponential family, and the Neural Network node does not estimate scale parameters when deviance training is used. Because scale parameters are not estimated when the deviance function is used, if there are multiple target variables, you should take care to standardize the target variables appropriately. A third form of objective function is provided by M-estimation, which is primarily used for robust estimation when the target values may contain outliers. As in likelihood estimation, M-estimation can be used to estimate scale parameters.

If you do not specify an objective function, the Neural Network node tries to choose one that is compatible with all of the specified error functions. The allowable combinations of error function and objective function are shown in the following table.

Table of Errors by Objective Functions

Error Function	Deviance	Likelihood	M-Estimation
Normal	Yes	Yes	No
Cauchy	No	Yes	No
Logistic	No	Yes	No
Huber	No	No	Yes
Biweight	No	No	Yes
Wave	No	No	No
Gamma	Yes	Yes	No
Poisson	Yes	Yes	No
Bernoilli	Yes	Yes	No
Binomial	Yes	Yes	No
Entropy	Yes	Yes	No
MBernoulli	Yes	Yes	No
Multinomial	Yes	Yes	No
MEntropy	Yes	Yes	No

Definitions for Error Function Formulas

The following definitions apply to the formulas below for individual error functions:

μ
: is the output (predicted value or posterior probability)

y
: is the target value

σ
: is the scale parameter

t
: is the number of trials for binomial error

$$z = \frac{y - \mu}{\sigma M_c}$$

is the argument to the ρ(.) function for M-estimators

M_c

is the M-estimation tuning constant

M_a

is the M-estimation scale constant

For all M-estimation methods, the individual error function is or function is $\sigma[\ \rho(z)M_c^2\ + M_a\,]$ where ρ(.) is defined separately for each M-estimation method.

Formulas for Individual Error Functions

Normal distribution

Likelihood: $$L(y, \mu, \sigma) = \frac{1}{2}\left[\left(\frac{y - \mu}{\sigma}\right)^2 + \ln(2\pi)\right] + \ln\ \sigma$$

Deviance: $L(y, \mu) = (y — \mu)^2$

Cauchy distribution

Likelihood: $$L(y, \mu, \sigma) = \ln\left\{\pi\sigma\left[\left(\frac{y - \mu}{\sigma}\right)^2 + 1\right]\right\}$$

Logistic distribution

Likelihood: $$L(y, \mu, \sigma) = -\ln\left\{\frac{\exp\left(\frac{y - \mu}{\sigma}\right)}{\sigma\left[1 + \exp\left(\frac{y - \mu}{\sigma}\right)\right]^2}\right\}$$

Huber M-estimator

$$\rho(z) = \begin{cases} .5z^2 & |z| < 1 \\ |z| - .5 & |z| \geq 1 \end{cases}$$

Biweight M-estimator

$$\rho(z) = \begin{cases} \dfrac{1-(1-z^2)^3}{6} & |z| < 1 \\ \dfrac{1}{6} & |z| \ge 1 \end{cases}$$

Wave M-estimator

$$\rho(z) = \begin{cases} \dfrac{1-\cos(\pi z)}{\pi^2} & |z| < 1 \\ \dfrac{2}{\pi^2} & |z| \ge 1 \end{cases}$$

Gamma distribution

Likelihood: $$L(y, \mu, \sigma) = -\sigma \ln\left(\frac{y}{\mu}\right) + \frac{y}{\mu} + \ln \Gamma(\sigma)$$

Deviance: $$L(y, \mu) = -2\left[\ln\left(\frac{y}{\mu}\right) - \frac{y}{\mu} + 1\right]$$

Poisson distribution

Likelihood: $$L(y, \mu) = \mu - y \ln \mu + \ln \Gamma(y+1)$$

Deviance: $$L(y, \mu) = 2\left[\ln\left(\frac{y}{\mu}\right) - (y - \mu)\right]$$

Bernoulli distribution

Likelihood: $$L(y, \mu) = \begin{cases} -\ln(1-\mu) & y = 0 \\ -\ln \mu & y = 1 \end{cases}$$

Deviance: $$L(y, \mu) = \begin{cases} -2\ln(1-\mu) & y = 0 \\ -2\ln \mu & y = 1 \end{cases}$$

Binomial distribution

Likelihood: $$L(y, \mu, t) = -y \ln \mu - (t - y) \ln(1 - \mu)$$

Deviance: $$L(y, \mu, t) = 2\left\{y \ln\left(\frac{y}{\mu t}\right) + (t - y) l \ln\left(\frac{t - y}{t - \mu t}\right)\right\}$$

Entropy error function

Likelihood: $L(y,\mu) = -y\ln\mu - (1-y)\ln(1-\mu)$

Deviance: $$L(y,\mu) = 2\left\{y\ln\left(\frac{y}{\mu}\right) + (1-y)l\ln\left(\frac{1-y}{1-\mu}\right)\right\}$$

Multiple Bernoulli

Likelihood: $$L(y,\mu) = \begin{cases} 0 & y = 0 \\ -\ln\mu & y = 1 \end{cases}$$

Deviance: $$L(y,\mu) = \begin{cases} 0 & y = 0 \\ -2\ln\mu & y = 1 \end{cases}$$

Multinomial distribution

Likelihood: $L(y,\mu) = -y\ln\mu$

Deviance: $$L(y,\mu,t) = 2y\ln\left(\frac{y}{\mu t}\right)$$

Multiple entropy

Likelihood: $L(y,\mu) = -y\ln\mu$

Deviance: $$L(y,\mu) = 2y\ln\left(\frac{y}{\mu}\right)$$

Formulas for Total Error Function

To compute the total error function L_+, let the individual error functions $L(y_{ik},\mu_{ik})$, $L(y_{ik},\mu_{ik},\sigma)$, or $L(y_{ik},\mu_{ik},t_i)$ be defined as above where

μ_{ik}

output (predicted value or posterior probability) for the kth unit of the ith case.

y_{ik}

target value for the kth unit of the ith case.

t_i

number of trials for the ith case when a binomial or multinomial error functions is used.

f_i

the frequency for the ith case.

$$z_{ik} = \frac{y_{ik} - \mu_{ik}}{\sigma M_c}$$

is the argument to the ρ(.) function for M-estimators.

Then the total error function is defined by one of the following formulas:

- For a binomial or multinomial error function,

$$L_+ = \sum_i f_i \sum_k L(y_{ik}, \mu_{ik}, t_i)$$

- For an error function with a scale parameter,

$$L_+ = \sum_i f_i \sum_k L(y_{ik}, \mu_{ik}, \sigma)$$

- For other error functions, $L_+ = \sum_i f_i \sum_k L(y_{ik}, \mu_{ik})$

Formula for Penalty Function

The penalty function is

$$P_+ = d \sum_l w_{(l)}^2$$

where d is the weight decay constant and w(l)is the lth weight excluding output biases. The objective function is

$$\overline{O} = \frac{L_+ + P_+}{\sum_i \sum_k f_i}$$

Choice of Architecture

Overview

In developing a neural network, there are many choices to be made: the number of inputs to use, which basic network architecture to use, whether to use a hidden layer, the number of units in the hidden layer, the activation and combination functions to use, and so on. If you have considerable prior information about the function to be learned, you may be able to make some of these choices based on theoretical considerations. More often, it takes trial and error to find a good architecture. This section will offer some general guidelines.

Since regression and trees are often much faster to train than neural nets, you should try those methods first to provide a baseline for comparing neural network results.

For exploring network architectures, there is no need to use huge data sets. For an interval target, the number of training cases need only be 5 to 25 times the number of estimated parameters in the network, depending on the noise level in the target variable. For classification, the number of training cases in the smallest class need only be 5 to 25 times the number of estimated parameters in the network. You can use a representative sample (a stratified sample if you are trying to predict rare events) of the data for exploration and save the weights of the best architecture you find. Then you can use those weights to initialize training on a larger data set to fine-tune the network.

Hidden Layer

Your neural network architecture may not need a hidden layer at all. Linear and generalized linear models are useful in a wide variety of applications. And even if the function you want to learn is mildly nonlinear, you may get better generalization with a simple linear model than with a complicated nonlinear model if there is too little data or too much noise to estimate the nonlinearities accurately.

In MLPs that have any of a wide variety of continuous nonlinear hidden-layer activation functions, one hidden layer with an arbitrarily large number of units suffices for the universal approximation property. But there is no theory to tell you how many hidden units are needed to approximate any given function.

Number of Hidden Units

The best number of hidden units depends on the number of training cases, the amount of noise, and the complexity of the function you are trying to learn. If you are using a neural network, you probably don't know exactly how much noise the target values have, or how complex the function is. Hence, it will generally be impossible to determine the best number of hidden units without training numerous networks with different numbers of hidden units and estimating the generalization error of each network.

The simplest procedure is to begin with a network with no hidden units, then add one hidden unit at a time. Estimate the generalization error of each network. Stop adding hidden units when the generalization error goes up.

The Neural Network node will provide you with a rough guess for the largest number of hidden units you can use without serious risk of overfitting, providing that you supply a rough guess for the noise level in the target values. You can specify the noise level as High, Moderate, Low, or Noiseless via the Neural Network node's properties panel. The noise levels can be interpreted according to the following table.

Table 58.1 *Interpretation of Noise Levels*

Noise Level	Misclassification Rate	Signal to Noise Ratio	R-squared
High	>40%	<1	<0.5
Moderate	20%	4	0.8
Low	10%	10	0.9
Noiseless	0%	Infinite	1.0

Note: The number of hidden units estimated from the noise level is not an optimal value. It is only a crude estimate of how many hidden units can be used without serious overfitting. To get the best generalization, you should experiment with different numbers of hidden units.

Neural Network Terms and Statistical Terms

Neural Network Term	Statistical Term
adaptation, learning, training, or self-organization	estimation or model selection
architecture	model
bias	intercept
case, example, pattern, sample	observation (in SAS System terminology)
competitive learning	cluster analysis
cross entropy or relative entropy	Kullback-Leibler divergence
epoch	iteration
error function or Lyapunov function	estimation criterion
error (target — output)	residual (observed — predicted)
features	variables
function mapping	regression
functional links	transformations
generalization	interpolation or extrapolation
general regression neural network	kernel regression
higher order network	polynomial regression or a model with interaction terms
higher-order neurons	interactions
inputs	independent variables
least mean squares	ordinary least squares
Lyapunov function	estimation criterion
noise	error
output	predicted value
perceptron (with no hidden layers)	Generalized Linear Model

Neural Network Term	Statistical Term
phi-machine	linear model
probabilistic neural network	kernel discriminant analysis
softmax function	multiple logistic function
supervised training or heteroassociation	discriminant analysis, regression
target or desired output	dependent variable
training set	sample
unsupervised training, encoding, or autoassociation	data reduction or cluster analysis
weight	parameter estimate

References

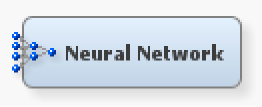
Bishop, C.M 1995. *Neural Networks for Pattern Recognition*. Oxford, England: Oxford University Press.

Tao, K.M 1993. “A Closer Look at the Radial Basis Function(RBF) Networks.” *Conference Record of the Twenty-Seventh Asilomar Conference on Signals, Systems, and Computers*, Los Alamitos, CA, Vol 1: 401–405.

Werntges, H.W 1993. “Partitions of Unity Improve Neural Function Approximation.” *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, Vol 2: 914–918.

Chapter 59
Neural Networking Node: Usage

Neural Network Node: Usage



Overview of the Neural Network Node

Before reading this section, you should be familiar with the Neural Network Node: Reference on page 859 section and the Predictive Modeling on page 149 section, which contains information that applies to all of the predictive modeling nodes.

For general information regarding the use of neural networks, as well as an extensive bibliography, consult the online Neural Network FAQ (Frequently Asked Questions). You can use a Web browser to read the FAQ at the following URL: `ftp://ftp.sas.com/pub/neural/FAQ.html`.

Neural Network Node Data Set Requirements

The training data that you input into the Neural Network node must contain at least one target variable and at least one input variable.

Neural Network Node Properties

Neural Network Node General Properties

The following general properties are associated with the Neural Network node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Neural node added to a diagram will have a Node ID of Neural. The second Neural node added to a diagram will have a Node ID of Neural2, and so on.

- **Imported Data** — the Imported Data property provides access to the Imported Data — Neural window. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Exported Data** — the Exported Data property provides access to the Exported Data — Neural window. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Neural Network Node Train Properties

The following train properties are available in the Neural Network node:

- **Variables** — Use the Variables property of the Neural Network node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, as well as select which variables you want included in reports.

- **Continue Training** — Use the Continue Training property of the Neural Network node to specify whether current estimates should be used as starting values for training. If the Continue Training property is set to **Yes**, the estimates from a previous run of the node will be used as initial values for subsequent training, and all other properties will be ignored. To use the Continue Training property, an estimates data set must have been created by the node prior to setting the Continue Training property to **Yes**.

- **Network** — Select the ... button to the right of the Network property to open a window that you can use to customize network options. The available user defined network options that you can choose in the User Defined Network Options window are

  - **Architecture** — Use the Architecture property of the Neural Network node to specify the network architecture that you want to use during network training.

The default is MLP (multilayer perceptron), which has no direct connections, and the number of hidden neurons is data dependent.

You can choose from the following network architectures:

  - **Generalized Linear Model**
  - **Multiplayer Perceptron** (default)
  - **Ordinary Radial — Equal Width**
  - **Ordinary Radial — Unequal Width**
  - **Normalized Radial — Equal Height**
  - **Normalized Radial — Equal Volumes**
  - **Normalized Radial — Equal Width**
  - **Normalized Radial — Equal Width and Height**
  - **Normalized Radial — Unequal Width and Height**
  - **User** — enables the user to define a network that has a single hidden layer. Selecting the User property enables the User Defined Network Options property setting.

- **Direct Connection** — Direct connections define linear layers, whereas hidden neurons define nonlinear layers. Use the Direct Connection property of the Neural Network node to indicate whether you want to create direct connections from inputs to outputs in your neural network. The default setting for the Direct Connection property is **No**. In the default setting, each input unit is connected to each hidden unit, and each hidden unit is connected to each output unit. Setting the Direct Connection property to **Yes** adds direct connections between each input unit and each output unit.
- **Number of Hidden Units** — Use the Number of Hidden Units property of the Neural Network node to specify the number n of hidden units that you want to use in the hidden layer. Permissible values are integers between 1 and 64. The default value is 3.
- **Randomization Distribution** — Use the Randomization Distribution property of the Neural Network node to specify the distribution to apply to the weights in your user-defined network. Permissible settings for the Randomization Distribution property of the Neural Network node are
  - **Normal** — The weights are distributed using a Normal bell curve. Normal is the default setting for the Randomization Distribution property.
  - **Cauchy** — The distributed weights favor the most frequent values in the Cauchy distribution. Cauchy is most often used when you want to predict the approximate conditional mode instead of the conditional mean.
  - **Uniform** — The weights are uniformly distributed across the network with the Uniform Distribution setting.
- **Randomization Center** — Use the Randomization Center property of the Neural Network node to specify the center of the distribution of the weights of your user-defined network. Permissible values are real numbers. The default value for the Randomization Center property is 0.0.
- **Randomization Scale** — Use the Randomization Scale property of the Neural Network node to specify the scale of the distribution of the weights in your user-defined network. Permissible values are real numbers. The default value for the Randomization Scale property is 0.1.

- **Input Standardization** — Use the Input Standardization property of the Neural Network node to specify the method that is used to standardize the input values in your user-defined network. Permissible values are **None**, **Standard Deviation**, **Range**, and **Midrange**. The default setting for the Input Standardization property is **Standard Deviation**.
- **Hidden Layer Combination Function** — The computed values from previous units in a neural network that feed into the given unit are combined into a single value using a combination function. The combination function uses the weights, bias, and altitude. Use the Hidden Layer Combination Function property of the Neural Network node to specify the hidden layer combination function that you want to use in your user-defined network. Permissible settings are as follows
  - **Default** — The Neural Network node uses the default PROC NEURAL setting for the Hidden Layer Combination Function, based on target and other Neural Network node property settings.
  - **Add** — Additive combination function. Additive combination functions are used in architectures such as generalized additive networks. The default activation function for the additive combination function is the Identity function. The additive combination function can use any number of hidden units.
  - **Linear** — Linear combination function. Linear combination functions are most often used for multilayer perceptrons (MLPs). The default activation function for the Linear combination function is the Hyperbolic Tangent. The Linear combination function can use any number of hidden units.
  - **EQSlopes** — EQSlopes is identical to the Linear combination function, except that the same connection weights are used for each unit in the layer, although different units have different biases. EQSlopes is mainly used for ordinal targets. The default activation function for the EQSlopes combination function is the Hyperbolic Tangent. The EQSlopes combination function can use any number of hidden units.
  - **EQRadial** — Radial basis function with equal heights and widths for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EHRadial** — Radial basis function with equal heights but not widths for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EWRadial** — Radial basis function with equal widths but not heights for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EVRadial** — Radial basis function with equal volumes for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **XRadial** — Radial basis function with unequal widths and heights for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions

that have two or more hidden units use the Softmax activation function by default.

- **Hidden-Layer Activation Function** — The value produced by the combination function is transformed by an activation function, which involves no weights or other estimated parameters. Use the Hidden-Layer Activation Function property of the Neural Network node to specify the hidden-layer activation function for your user-defined network. Permissible values are as follows:

  - **Default** — The Neural Network node uses the default PROC NEURAL setting for the Hidden Layer Activation Function, based on the combination function setting and the number of hidden units that exist in the layer.

  - **Arc Tangent** — Arc tangent hidden layer activation function for linear combination functions. The Arc Tangent function can be used as an alternative to the default Hyperbolic Tangent activation function. The arc tangent function for net input g is arctan(g).

  - **Elliot** — Elliot hidden layer activation function for linear combination functions. The Elliot function can be used as an alternative to the default Hyperbolic Tangent activation function.

  - **Hyperbolic Tangent** — Hyperbolic Tangent is the default hidden layer activation function for linear combination functions. The hyperbolic tangent function for net input g is htan(g).

  - **Logistic** — The Logistic hidden layer activation function for net input g is $(1 + \exp(-g))^{-1}$.

  - **Gauss** — The Gauss hidden activation function is useful when the data to contains local features (for example, bumps) in low dimensional subspaces.

  - **Sine** — Use the Sine hidden activation function with caution, because it yields an infinite Vapnik-Chervonenkis dimension, which can cause poor generalization. The range for the Sine activation function is [-1, 1] and the function for net input g is sin(g).

  - **Cosine** — Use the Cosine hidden activation function with caution, because it yields an infinite Vapnik-Chervonenkis dimension, which can cause poor generalization. The range for the Cosine activation function is [-1, 1] and the function for net input g is cos(g).

  - **Exponential** — The exponential function has the range (0, ∞) and is defined as exp(g).

  - **Square** — The square function has the range [0, ∞) and is defined as g^2.

  - **Reciprocal** — The reciprocal function has the range (0, ∞) and is defined as 1/g.

  - **Softmax** — The softmax function has the range (0, 1) and is defined as

    $$\frac{\exp(g)}{\sum (exponentials)}.$$

- **Hidden Bias** — Use the Hidden Bias property of the Neural Network node to specify whether to compensate the user-defined network for hidden bias. Permissible values are **Yes** and **No**. The default value is **Yes**.

- **Target Layer Combination Function** — The computed values from predecessor units that feed into the given unit are combined into a single value using a combination function. The combination function uses the weights, bias, and altitude. Use the Target Layer Combination Function property of the Neural

Network node to specify the target layer combination function for your user-defined network. Permissible values are as follows:

  - **Default** — The Neural Network node uses the default PROC NEURAL setting for the Target Layer Combination Function. The default target layer combination function varies according to the activation function.
  - **Add** — Additive combination function with no weights or bias. Additive combination functions are used in architectures such as generalized additive networks. The default activation function for the additive combination function is the Identity function. The additive combination function can use any number of hidden units.
  - **Linear** — Linear combination functions compute a linear combination of the weights and the values that feed into the unit and then add the bias value (the bias acts like an intercept). The default activation function for the Linear combination function is the Hyperbolic Tangent. The Linear combination function can use any number of hidden units.
  - **EQSlopes** — EQSlopes is identical to the Linear combination function, except that the same connection weights are used for each unit in the layer, although different units have different biases. EQSlopes is mainly used for ordinal targets. The default activation function for the EQSlopes combination function is the Hyperbolic Tangent. The EQSlopes combination function can use any number of hidden units.
  - **EQRadial** — Radial basis function with equal heights and widths for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EHRadial** — Radial basis function with equal heights but not widths for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EWRadial** — Radial basis function with equal widths but not heights for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **EVRadial** — Radial basis function with equal volumes for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.
  - **XRadial** — Radial basis function with unequal widths and heights for all units in the layer. Radial combination functions that have one hidden unit use the Exponential activation function by default. Radial combination functions that have two or more hidden units use the Softmax activation function by default.

- **Target Layer Activation Function** — The value produced by the combination function is transformed by an activation function, which involves no weights or other estimated parameters. Use the Target Layer Activation Function property of the Neural Network node to specify the target layer activation function for your user-defined network. The default activation function depends on the measurement level of the target variable. Select a function with a range that corresponds to the actual distribution of target values. Permissible values are as follows:

- **Default** — The Neural Network node uses the default PROC NEURAL setting for the Target Layer Activation Function, based on other Neural Network node property settings. The default target layer activation function depends on the selected combination function.
- **Identity** — Identity is the default target layer activation function for interval target variables. Identity is suggested for unbound target variables with range (-∞ , ∞). The Identity function for the net input g is g.
- **Linear** — Linear target layer activation function.
- **Exponential** — The Exponential target layer activation function is suggested for target variables with the range (0, ∞). The exponential function for the net input g is exp(g).
- **Reciprocal** — the Reciprocal target layer activation function is suggested for target variables with the range (0, ∞). The reciprocal function for the net input g is 1/g.
- **Square** — the Square target layer activation function is suggested for target variables with the range [0, ∞). The square function for the net input is g2.
- **Logistic** — Logistic is the default target layer activation function for ordinal target variables with the range (0,1). The logistic function for the net input is (1 + exp(-g))$^{-1}$.
- **MLogistic** — Multiple logistic, or Softmax, is the only output activation function allowed for nominal targets with an error function of multiple Bernoulli, multiple entropy, or multinomial. The range for the MLogistic target variables is (0,1) and the function for the net input is

  $$\frac{\exp(g)}{\sum(exponentials)}.$$

- **Softmax** — Uses the Softmax activation function for nominal target variables with a range of (0, 1).
- **Gauss** — The Gauss target layer activation function is useful when the data to contains local features (for example, bumps) in low dimensional subspaces.
- **Sine** — Use the Sine target layer activation function with caution, because it yields an infinite Vapnik-Chervonenkis dimension, which can cause poor generalization. The range for the Sine activation function is [-1, 1] and the function for net input g is sin(g).
- **Cosine** — Use the Cosine target layer activation function with caution, because it yields an infinite Vapnik-Chervonenkis dimension, which can cause poor generalization. The range for the Cosine activation function is [-1, 1] and the function for net input g is cos(g).
- **Elliot** — Elliot target layer activation function for linear combination functions. The Elliot function can be used as an alternative to the Hyperbolic Tangent activation function.
- **Tanh** — Hyperbolic tangent target layer activation function for linear combination functions. The hyperbolic tangent function for net input g is htan(g).

  - **ArcTan** — Arc tangent target layer activation function for linear combination functions. The Arc Tangent function can be used as an alternative to the Hyperbolic Tangent activation function. The arc tangent function for net input g is arctan(g).

- **Target Layer Error Function** — Each unit in a neural network produces a single computed value. Input and hidden units pass the computed values to other hidden or output units. The predicted value for output units is compared with the target value to compute the error function. Use the Target Layer Error Function property of the Neural Network node to specify the target layer error function for your user-defined network. Permissible values are as follows:

  - **Default** — The Neural Network node uses the default PROC NEURAL setting for the Target Layer Error Function, based on the specified objective function settings.

  - **Normal** — Normal distribution, also called the least-squares or mean-squared-error criterion. Normal is suitable for unbounded interval targets with constant conditional variance, no outliers, and a symmetric distribution. Normal can also be used for categorical targets that have outliers.

  - **Cauchy** — The Cauchy distribution may be used with any kind of target variable. It is most often used when you want to predict the approximate conditional mode instead of the conditional mean.

  - **Logistic** — Logistic distribution may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain outliers.

  - **Huber** — The Huber M-estimator is suitable for unbounded interval targets that have outliers or that have a moderate degree of inequality of the conditional variance and a symmetric distribution. Huber can also be used for categorical targets when you want to predict the mode rather than the posterior probability.

  - **Biweight** — The Biweight M-estimator may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain severe outliers. Because of severe problems with local minima, you should obtain initial values by training with the Huber M-estimator before using the biweight M-estimator.

  - **Wave** — The Wave M-estimator may be used with any kind of target variable. It is most often used with interval targets where the noise distribution may contain severe outliers. Because of severe problems with local minima, you should obtain initial values by training with the Huber M-estimator before using the Wave M-estimator.

  - **Gamma** — The Gamma distribution may be used only with strictly positive interval target variables. It is most often used when the standard deviation of the noise is proportional to the mean of the target variable.

  - **Poisson** — The Poisson distribution is suitable for skewed, nonnegative interval targets, especially counts of rare events, where the conditional variance is proportional to the conditional mean.

  - **Bernoulli** — The Bernoulli distribution is suitable for a target that takes only the values zero and one. The Bernoulli distribution is a binomial distribution with one trial

  - **Entropy** — The Entropy error function is cross-entropy or relative entropy for independent interval targets with values between zero and one inclusive.

        - **MBernoulli** — The Multiple Bernoulli error function is suitable for categorical (nominal or ordinal) targets.

        - **Multinomial** — The Multinomial error function is may be used only with two or more interval target variables with non-negative values, where each case represents a number of trials that are equal to the sum of the target values. The activation function must force the outputs to sum to one, like Softmax.

        - **Mentropy** — The Multiple Entropy error function is cross-entropy or relative entropy for targets that sum to 1. It is the same criterion as Kullback-Leibler divergence. Multiple entropy should normally be used only with two or more interval target variables with values that lie between zero and one inclusive, and that sum to one for each case. However, multiple entropy may also be used with nominal or ordinal targets, primarily for software testing. The activation function must force the outputs to sum to one, like Softmax.

    - **Target Bias** — Use the Target Bias property of the Neural Network node to specify a target bias. Permissible values are **Yes** and **No**. The default value is **Yes**.

    - **Weight Decay** — Use the Weight Decay property to specify the weight decay parameter. The larger the weight decay value, the greater the restriction on total weight growth. The weight decay parameter does not affect the output weights when the activation function in the hidden layer is set to Softmax, because the adjustment to the error is too small. The default setting for the Weight Decay property is zero.

- **Optimization** — Select the ... button to the right of the Optimization property to open a window that you can use to customize Neural Network optimization settings.

  The following Neural Network optimization properties can be configured in the Optimization window:

    - **Training Technique** — Use the Training Technique property of the Neural Network node to specify the neural network training algorithm that you want to use. The available training techniques are

        - **Default** — Use the Default property to let SAS choose the default training technique. The default method that SAS chooses depends on the number of weights that are applied to the data at execution.

        - **Trust-Region** — Use the Trust-Region technique for small and medium optimization problems that have up to 40 parameters.

        - **Levenberg-Marquardt** — Use the Levenberg-Marquardt technique for smooth least squares objective functions that have up to 100 parameters.

        - **Quasi-Newton** — Use the Quasi-Newton technique for medium optimization problems that have up to 500 parameters.

        - **Conjugate Gradient** — Use the conjugate gradient technique for large data mining problems with over 500 parameters.

        - **DBLDog** — The Double Dogleg optimization technique. The Double Dogleg optimization method combines the ideas of Quasi-Newton and Trust-Region methods. The Double Dogleg algorithm computes in each iteration the step s(k) as the linear combination of the steepest descent or ascent search direction s1(k) and a Quasi-Newton search direction s2(k) that is forced not to leave a prespecified Trust-Region radius.

        - **Back Prop** — The standard batch backpropagation technique. Standard backprop is the most popular back propagation method, but it is slow, unreliable, and requires the user to tune the learning rate manually, which can be a tedious process.

  - **RProp** — The standard incremental backprop technique uses a separate learning rate for each weight and adapts all the learning rates during training. The RProp technique tends to take more iterations than conjugate gradient methods, but each iteration is very fast.
  - **QProp** — The QuickProp technique is a Newton-like method which uses a diagonal approximation to the Hessian matrix. Like RProp, QProp tends to take more iterations than conjugate gradient methods, but each iteration is very fast. For more detailed information about network training techniques, see the "Neural Network Node: Reference" on page 859 documentation.

- **Maximum Iterations** — Use the Maximum Iterations property of the Neural Network node Train Options to specify the maximum number of iterations that you want to allow during network training. Permissible values are integers from 1 to 1000. The default value is 50.
- **Maximum Time** — Use the Maximum Time property of the Neural Network node Train Options to specify the maximum amount of CPU time that you want to use during training. Permissible values are 5 minutes, 10 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, or 7 hours. The default setting for the Maximum Time property is 4 hours.
- **Nonlinear Options** — The Nonlinear Options section of the Optimization window contains the following settings:
  - **Use Defaults** — Specifies wether the default values are used.
  - **Absolute** — Use the Absolute convergence property of the Neural Network node to specify an absolute convergence criterion. The Absolute convergence is a function of the log-likelihood for the intercept-only model. The optimization is to maximize the log-likelihood. The default value is -1.34078E154.
  - **Absolute Function** — Use the Absolute Function property of the Neural Network node to specify an absolute function convergence criterion. Absolute Function is a function of the log-likelihood for the intercept-only model. The minimum value is 0 and the default value is 0.0.
  - **Absolute Function Times** — Use the Absolute Function Times property of the Neural Network node to specify the number of successive iterations for which the absolute function convergence criterion must be satisfied before the process can be terminated. The minimum value and the default value are both 1.
  - **Absolute Gradient** — Use the Absolute Gradient property of the Neural Network node to specify the absolute gradient convergence criterion that you want to use. All values must be larger than 0. The default value is 1.0E-5.
  - **Absolute Gradient Times** — Use the Absolute Gradient Times property of the Neural Network node to specify the number of successive iterations for which the absolute gradient convergence criterion must be satisfied before the process can be terminated. The default value is 1.
  - **Absolute Parameter** — Use the Absolute Parameter property of the Neural Network node to specify the absolute parameter convergence criterion. The default value is 1.0E-8. The value must be greater than zero.
  - **Absolute Parameter Times** — Use the Absolute Parameter Times property of the Neural Network node to specify the number of successive iterations for

  which the absolute parameter convergence criterion must be satisfied before the process can be terminated. The default value is 1.

  - **Relative Function** — Use the Relative Function property of the Neural Network node to specify the relative function convergence criterion. The minimum and the default value are both 0.0.
  - **Relative Function Times** — Use the Relative Function Times property of the Neural Network node to specify the number of successive iterations for which the relative function convergence criterion must be satisfied before the process can be terminated. The default value is 1.
  - **Relative Gradient** — use the Relative Gradient property of the Neural Network node to specify the relative gradient convergence criterion. The default value is 1.0E-6.
  - **Relative Gradient Times** — Use the Relative Gradient Times property of the Neural Network node to specify the number of successive iterations for which the relative gradient convergence criterion must be satisfied before the process can be terminated. Permissible values for the Relative Gradient Times property are integers greater than 0. The default value is 1.

- **Propagation Options** — The Propagation Options section of the Optimization window contains the following settings:
  - **Accelerate** — Use the Accelerate propagation option to specify the rate of increase of learning for the RPROP optimization technique.
  - **Decelerate** — Use the Decelerate propagation option to specify the rate of decrease of learning for the RPROP optimization technique.
  - **Learn** — Use the Learn propagation option to specify the learning rate for BACKPROP or the initial learning rate for QPROP and RPROP optimization techniques.
  - **Maximum Learning** — Use the Maximum Learning propagation option to specify the maximum learning rate for the RPROP optimization technique.
  - **Minimum Learning** — Use the Minimum Learning propagation option to specify the minimum learning rate for the RPROP optimization technique.
  - **Momentum** — Use the Momentum propagation option to specify the momentum rate.
  - **Maximum Momentum** — Use the Momentum propagation option to specify the maximum momentum rate for the QPROP optimization technique.
  - **Tilt** — Use the Tilt propagation option to specify the tilt rate for the QPROP optimization technique.

- **Preliminary Training Options** — The Preliminary Training Options section of the Optimization window contains the following settings:
  - **Enable** — Use the Enable property to specify whether to enable preliminary training in the Neural Network node. The default setting for the Enable property is **Yes**.
  - **Number of Runs** — Use the Number of Runs property of the Neural Network node to specify the number of preliminary runs that you want to perform. Permissible values are integers greater than or equal to 1. The default value is 5.
  - **Maximum Iterations** — Use the Maximum Iterations property to specify the maximum number of iterations that you want to allow during preliminary

network training. Permissible values are integers from 1 to 100. The default value is 10.

  - **Maximum Time** — Use the Maximum Time property to specify the maximum amount of CPU time that you want to use during preliminary training. Permissible values are 5 minutes, 10 minutes, 30 minutes, 1 hour, 2 hours, 4 hours, or 7 hours. The default setting for the Maximum Time property is 1 hour.

- **Initialization Seed** — Use the Initialization Seed property to specify the random seed value that you want to use during network weight optimization.

- **Model Selection Criterion** — Use the Model Selection Criterion property of the Neural Network node to specify the most appropriate model from your network. The available criteria are

  - **Profit/Loss** — (default setting) The Profit/Loss model selection criterion chooses the model that maximizes the profit or minimizes the loss for the cases in the validation data set. If a validation data set is not available, then the node uses the training data set.

  - **Misclassification** — The Misclassification model selection criterion chooses the model that has the smallest misclassification rate for the validation data set. If the validation data set does not exist, then the training data set is used.

  - **Average Error** — The Average Error model selection criterion chooses the model that has the smallest average error for the validation data set. If the validation data set does not exist, then the training data set is used.

- **Suppress Output** — Set the Suppress Output property of the Neural Network node to **Yes** if you want to suppress all printed output generated by the Neural Network node. The default setting for the Suppress Output property is **No**.

Neural Network Node Score Properties

The following score properties are available in the Neural Network node:

- **Hidden Units** — Set the Hidden Units property of the Neural Network node to **Yes** if you want to create hidden units variables in your scoring data. The default setting of **No** suppresses the creation of hidden units.

- **Residuals** — Set the Residuals property of the Neural Network node to **No** if you want to suppress the creation of variables for residuals when scoring data. The default setting for the Residuals property is **Yes**.

- **Standardization** — Set the Standardization property of the Neural Network node to **Yes** if you want to create standardization variables in your scoring data. The default setting of **No** suppresses the creation of standardization variables.

Neural Network Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.

- **Last Error** — displays the error message from the last run.

- **Last Status** — displays the last reported status of the node.

- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Neural Network Node Results

You can open the Results window of the Neural Network node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Neural Network node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Neural Network node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.
- **SAS Results**
  - **Log** — the SAS log of the Neural Network run.
  - **Output** — the SAS output of the Neural Network run. The SAS Output for the Neural Network node includes a variable summary, a model event summary, a decision matrix, a table of predicted and decision variables, optimization start parameter estimates, dual quasi-Newton optimization statistics, optimization results parameter estimates, optimization summary statistics, fit statistics for train, classification tables for train and validation data sets, decision tables for train and validation data sets, an event classification table, assessment score rankings for train and validation data sets, and assessment score distribution for train and validation data sets.
  - **Flow Code** — the SAS code used to produce the output that the Neural Network node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Assessment**
  - **Fit Statistics** — opens the “Neural Network Node Fit Statistics Table” on page 899 , which contains fit statistics from the model that are calculated for interval and non-interval targets.

- **Residual Statistics** — opens the residual statistics chart for an interval target.
- **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

  The Score Rankings Overlay plot displays both train and validate statistics on the same axis. By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any).

  - posterior probability of target event
  - number of events
  - cumulative and noncumulative lift values
  - cumulative and noncumulative % response
  - cumulative and noncumulative % captured response
  - gain
  - actual profit or loss
  - expected profit or loss.
- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

  For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

  The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are

  - Percentage of Events — for categorical targets
  - Number of Events — for categorical targets
  - Cumulative Percentage of Events — for categorical targets
  - Mean for Predicted — for interval targets
  - Max. for Predicted — for interval targets
  - Min. for Predicted — for interval targets

- **Model**
  - **Iteration Plot** — by default, a line plot of the average squared error versus the training iteration is displayed.
  - **Weights — Final** — a two-dimensional histogram of all weights generated for the selected iteration of the selected run.

  - **Weights — History** — a chart containing the neural history estimates.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Neural Network Node Fit Statistics Table

		Statistic Calculated for	
Fit Statistic Variable Name	**Fit Statistic Label**	**Non-Interval Targets**	**Interval Targets**
AIC	Akaike's Information Criterion	Yes	Yes
APROF	Average Profit of the Target	Yes	No
ASE	Average Squared Error	Yes	Yes
AVERR	Average Error Function	Yes	Yes
DFE	Degrees of Freedom for Error	Yes	Yes
DFM	Model Degrees of Freedom	Yes	Yes
DFT	Total Degrees of Freedom	Yes	Yes
DIV	Divisor for _ASE_	Yes	Yes
ERR	Error Function	Yes	Yes
FPE	Final Prediction Error	Yes	Yes
MAX	Maximum Absolute Error	Yes	Yes
MISC	Misclassification Rate	Yes	No
MSE	Mean Squared Error	Yes	Yes
NOBS	Sum of Frequencies	Yes	Yes

		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
_NW_	Number of Estimated Weights	Yes	Yes
PROF	Total Profit	Yes	Yes
RASE	Root Average Squared Error	Yes	Yes
RFPE	Root Final Prediction Error	Yes	Yes
RMSE	Root Mean Squared Error	Yes	Yes
SBC	Schwarz's Bayesian Criterion	Yes	Yes
SSE	Sum of Squared Errors	Yes	Yes
SUMW	Sum of Case Weights Times Frequency	Yes	Yes
WRONG	Number of Wrong Classifications	Yes	No

Chapter 60
Partial Least Squares Node

Partial Least Squares Node

Overview of the Partial Least Squares Node

The Partial Least Squares (PLS) node is located on the Model tab of the Enterprise Miner tools bar. Partial least squares is a tool that you can use to model continuous and binary targets. The SAS Enterprise Miner PLS node is based on SAS/STAT PROC PLS. The Enterprise Miner PLS node produces data step score code and standard predictive model assessment results.

Data mining problems that might traditionally be approached using multiple linear regression techniques become more difficult when there are many input variables or there is significant collinearity between variables. In these instances, regression models tend to overfit the training data and do not perform well when modeling other data. Often this is the case when just a few latent variables among the many input variables are responsible for most of the variation in response, or target variable values.

Partial least squares is a tool that is useful for extracting the latent input variables that account for the greatest variation in the predicted target. To many data miners, PLS means "projection to latent structures". While PLS is useful for identifying latent variables from a pool of many, the analytical results of the PLS tool are not useful for identifying variables of minor or no importance. Those tasks should be performed using other Enterprise Miner data mining tools, such as the Variable Selection node.

It is difficult to identify the weighting of the latent input predictor variables that PLS uses, because they are based on cross-product relations with the target variable, instead of the covariances between the input variables themselves as is more commonly seen in common factor analysis.

Partial Least Squares Node Algorithm

The PLS algorithm is a multivariate extension of a multiple linear regression that was developed by Herman Wold in the 1960s as an econometric technique. Since then, PLS has been widely used in industrial modeling and process control systems where processes can have hundreds of input variables and scores of outputs. PLS is used today in data mining projects for marketing, social sciences, and education.

The PLS algorithm reduces the set of variables (both input and target) to principal component matrices. The input variable components are used to predict the scores on the target variable components, then the target variable component scores are used to predict the value of the target variable.

Why does this the PLS linear method work when collinearity exists between input variables? Because the principal component scores for the target variable are linear combinations of the original input variables, no correlation exists between the component score variables that become inputs in the predictive model.

Partial Least Squares Node and Missing Data Set Values

If an observation contains missing values, the Partial Least Squares node excludes the observation from the analysis. If the input data set that you want to perform Partial Least Squares analysis on contains a significant amount of observations with missing values, you should use the Enterprise Miner Impute node to replace or impute missing variable values before submitting the data set to the Partial Least Squares node.

Partial Least Squares Node Data Set Results

The following rules are useful when preparing data for use with the Partial Least Squares node:

- The PLS node must have at least one interval target or one binary target.
- Numeric binary targets should be coded 1 (event) or 0 (nonevent).
- Characteristic binary targets should be coded into dummy targets with values 1 or 0.
- If predicted values in the binary target are greater than 1, then the PLS node sets the predicted values to 1.
- If predicted values in the binary target are less than 0, then the PLS node sets the predicted values to 0.
- The PLS node does not support frequency variables.
- The PLS node does not support decision score code.
- The PLS node does not support Profit/Loss fit statistics.

Partial Least Squares Node Cross-Validation

When you use raw or training data to fit partial least squares predictive models, you should verify, or cross-validate your results to ensure that you have not over-fitted the training data during model creation. Cross-validating means applying the current results to a set of hold-out observations that were not used to estimate the parameters of the original model. Cross validation computes predicted value and residual statistics for

- observations that were not used in the computations for fitting the current model, but that have observed values for the dependent variables (the cross-validation sample).
- observations that were not used in the computations for fitting the current model, but do not have data for the dependent variables (prediction sample).

Partial Least Squares Node Properties

Partial Least Squares Node General Properties

The following general properties are associated with the Partial Least Squares node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first PLS node added to a diagram will have a Node ID of PLS. The second PLS node added to a diagram will have a Node ID of PLS2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — PLS window. The Imported Data — PLS window contains a list of the ports that provide data sources to the PLS node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — PLS window. The Exported Data — PLS window contains a list of the output data ports that the PLS node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Partial Least Squares Node Train Properties

The following train properties are associated with the Partial Least Squares Node.

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the Partial Least Squares node. Select the ... button to open a window containing the variables table. You can set the variable status to either Use or Don't Use in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. For more information, view the Enterprise Miner Help section on the Variables table.

Partial Least Squares Node Train Properties: Modeling Techniques

- **Regression Model** — Use the Regression Model property to specify the method to be used for general factor extraction.
  - **PLS** — performs general factor extraction using the original Partial Least Squares (PLS) method of Wold (1966). PLS is the default setting for the Regression Model property.
  - **PCR** — performs general factor extraction using principal components regression. Traditional PCR methods use the first k components (first by having the highest eigenvalues) to predict the response variable.
  - **SIMPLS** — performs general factor extraction using the SIMPLS method of de Jong (1993). When you have a single target variable, SIMPLS is identical to PLS. When you have many target variables, SIMPLS is computationally much more efficient. PLS and SIMPLS results are usually very similar.
- **PLS Algorithm** — When the Regression Model property is set to PLS or SIMPLS, use the PLS Algorithm property to specify the algorithm that you want to use to derive the extracted partial least squares factors.
  - **NIPALS** — requests the nonlinear iterative partial least squares (NIPALS) algorithm. NIPALS is the default setting for the PLS Algorithm property.
  - **SVD** — bases the PLS factor extraction on the Singular Value Decomposition (SVD) of X'Y. SVD is the most accurate but least computationally efficient approach.
  - **Eigenvalue** — bases the PLS factor extraction on the Eigenvalue (EIG) decomposition of Y'XX'Y
  - **RLGW** — bases the PLS factor extraction on an iterative approach that is efficient when there are many predictors (Rannar et al. 1994).
- **Maximum Iteration** — When the PLS Algorithm property is set to NIPALS or RLGW, use the Maximum Iteration property to specify the maximum number of iterations that are allowed for the NIPALS or RLGW algorithms. The Maximum Iteration property accepts integer inputs, and the default value is 200.
- **Epsilon** — When the PLS Algorithm property is set to NIPALS or RLGW, use the Epsilon property to specify the convergence criterion for the NIPALS or RLGW algorithms. The Epsilon property accepts real number inputs, and the default value is 1E-12.

Partial Least Squares Node Train Properties: Number of Factors

- **Default** — specifies whether to use the default number of factors to be extracted by the selected PLS model and algorithm. The Default property is a binary property, and the default value is **Yes**. If you set Default to **No**, you must specify the number of factors that you want to extract using the Number of Factors property.

- **Number of Factors** — when the Default property is set to **No**, use the Number of Factors property specifies the number of factors that you want to extract in your PLS analysis. The Number of Factors property accepts integer inputs, and the default setting for the Number of Factors property is 15.

Partial Least Squares Node Train Properties: Cross Validation

- **Cross Validation Method** — Specifies the method used to designate a proportion of the data to be used for cross-validation checks on your PLS predictive models.
  - **One at a time** — performs one-observation-at-a-time cross validation.
  - **Split** — excludes every nth observation.
  - **Random** — excludes randomly chosen subsets of observations. You can use the Number of Iterations and the Default No. of Test Obs. properties in the Random CV Options group to specify the number of subsets to exclude and the number of observations to be included in each subset.
  - **Block** — excludes blocks of n observations.
  - **Test Set** — performs cross validation using either the validation data set or the test data set.
  - **None** — does not perform cross validation. None is the default setting for the Cross Validation Method property.
- **CV N Parameter** — If the Cross Validation Method property is set to SPLIT or BLOCK, use the CV N Parameter to specify an integer value for n, the indexing value for the SPLIT and BLOCK methods. The default value for the CV N Parameter is 7.

Partial Least Squares Node Train Properties: Random CV Options

- **Number of Iterations** — When the Cross Validation Method property is set to RANDOM, the Number of Iterations property specifies the number of random subsets of observations to exclude. The Number of Iterations property accepts integer inputs, and the default value is 10.
- **Default Number of Test Observations** — When the Cross Validation Method property is set to RANDOM, the number of observations to be included in each random subset that is chosen for exclusion must be specified. If the Default Number of Test Observations property is set to **Yes**, then the number of observations to be included in each random subset is automatically calculated as one-tenth of the total number of observations. If the Default No. of Test Observations property is set to No, then the number of observations to be included in each random subset must be specified in the No. of Test Obs. property.
- **Number of Test Observations** — When the Cross Validation Method property is set to RANDOM and the Default Number of Test Observations property is set to **No**, use the Number of Test Observations property to specify the number of observations to be included in the random subsets that are chosen for exclusion.
- **Default Random Seed** — Specifies whether or not Enterprise Miner should use a default seed value to perform random number generation. Enterprise Miner uses the processor's clock time as default seed values. The default random seed generator will generate different random numbers every time the node is run. To generate random values that will not change every time the node is run, use the Seed property to specify a repeatable random number seed.

- **Seed** — Specifies the seed value for random number generation. The seed value will generate the same random numbers during successive node runs so that results can be reproduced in subsequent runs using the same data.

Partial Least Squares Node Train Properties: Variable Selection

- **Variable Selection Criterion** — Specifies the variable selection criterion.
  - **Coefficient** — Uses parameter estimates to select variables.
  - **Variable Importance** — Uses variable importance scores to select variables.
  - **Both** — Uses both parameter estimates and variable importance scores to select variables.
  - **Either** — Uses either parameter estimates and variable importance scores to select variables.
- **Para. Est. Cutoff** — Cutoff value for variable selection using parameter estimate. The default is 0.1.
- **VIP Cutoff** — Cutoff value for variable selection using variable importance for projection (VIP). The default is 0.8.
- **Export Selected Variables** — Specifies weather or not the node should export the variable selection result to a successor node. The default setting for the Export Selected Variables property is No.
- **Hide Rejected Variables** — Specifies whether to hide the rejected variables in successor node. The default setting for the Hide Rejected Variables property is **Yes**.

Partial Least Squares Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Partial Least Squares Node Results

You can open the Results window of the Partial Least Squares node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**

  - **Settings** — displays a window with a read-only table of the configuration information in the Partial Least Squares node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Partial Least Squares node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information on the data set submitted to the Partial Least Squares node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Partial Least Squares node run.
  - **Output** — the SAS output of the Partial Least Squares node run. Typical output includes information such as a Variable Summary by Role, Level and Count, summary of predicted and decision variables, summary of variation attributed to partial least square factors, extracted factors listed by variable, dependent variable weights, parameter estimates for centered and scaled data, standard parameter estimates, score output and report output if created, as well as training and validation fit statistics and classification tables for the target variable.
  - **Flow Code** — the SAS code used to produce the output that the Partial Least Squares node passes on to the next node in the process flow diagram.
  - **Train Graphs** — The Partial Least Squares node does not produce Train graphs.
  - **Report Graphs** — The Partial Least Squares node does not produce Report graphs.
- **Scoring**
  - **SAS Code** — the SAS scoring code that was generated by the Partial Least Squares node during its run.
  - **PMML Code** — the Partial Least Squares node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — The Partial Least Squares Fit Statistics table displays the statistics in the "Partial Least Squares Node Fit Statistics Table" on page 909 .
  - **Classification Chart** — The Classification chart displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represent the percentage of total observations.
  - **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

    The Score Rankings Overlay plot displays both train and validate statistics on the same axis. By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any).

    - posterior probability of target event
    - number of events
    - cumulative and noncumulative lift values
    - cumulative and noncumulative % response
    - cumulative and noncumulative % captured response
    - gain
    - actual profit or loss
    - expected profit or loss

  - **Score Rankings Matrix** — The score ranking matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. Plots can also be created for report variables. The chart choices are

    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response.

- **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used.

  For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

  The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are

  - Percentage of Events — for categorical targets
  - Number of Events — for categorical targets
  - Cumulative Percentage of Events — for categorical targets
  - Mean for Predicted — for interval targets
  - Max. for Predicted — for interval targets
  - Min. for Predicted — for interval targets

- **Variable Selection**

  - **Variable Selection** — The Variable Selection table displays the list of submitted variables and indicates whether each variable is selected as an input or rejected for use with the successor data mining node. The Variable Selection tool uses

Standardized Parameter Estimate and Variable Importance for Projection (VIP) scores as variable selection criteria. Along with Parameter Estimate and VIP scores for each variable, Rejected columns in the table identify the variables that are rejected for insufficient Parameter Estimate scores or low VIP scores.

- **Variable Importance For Projection** — The Variable Importance for Projection (VIP) plot is a bar plot showing the VIP scores of individual variables that were submitted to the node. A VIP score of 1 or better tends to indicate variables with stronger predictive powers, variables with VIP scores of less than 1 tend to be weaker predictive powers and should be removed from the analysis.
- **Absolute Standardized Parameter Estimates** — The Absolute Standardized Parameter Estimates plot is a bar plot showing the parameter estimate scores for each variable. The plot uses an absolute value y-axis with positive variable scores displayed as blue bars, and negative variable scores displayed as red bars. The y-axis displays a threshold line that reflects the value that was specified in the Score property for the Parameter Estimate cutoff value.

- **Model**
  - **Parameter Estimates** — The Parameter Estimates plot is a bar plot that displays variable parameter estimate scores ranked in descending magnitude. The y-axis for the Parameter Estimates plot uses actual values instead of absolute values.
  - **Absolute Parameter Estimates** — The Absolute Parameter Estimates plot is a bar plot that displays absolute variable parameter estimate scores ranked in descending magnitude. The y-axis for the Parameter Estimates plot uses absolute values instead of actual values.
  - **Percent Variation** — a plot that displays model percent variation.

Partial Least Squares Node Fit Statistics Table

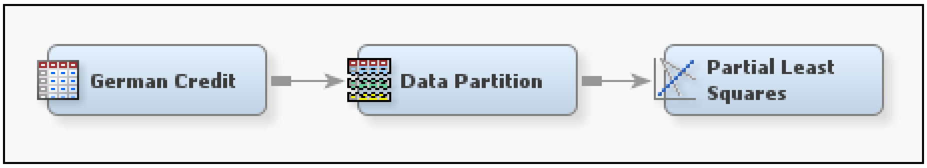
		Statistic Calculated for	
Fit Statistic Variable Name	**Fit Statistic Label**	**Non-Interval Targets**	**Interval Targets**
AIC	Akaike's Information Criterion	Yes	Yes
APROF	Average Profit of the Target	Yes	No
ASE	Average Squared Error	Yes	Yes
AVERR	Average Error Function	Yes	Yes
DFE	Degrees of Freedom for Error	Yes	Yes
DFM	Model Degrees of Freedom	Yes	Yes

		Statistic Calculated for	
Fit Statistic Variable Name	Fit Statistic Label	Non-Interval Targets	Interval Targets
DFT	Total Degrees of Freedom	Yes	Yes
DIV	Divisor for _ASE_	Yes	Yes
ERR	Error Function	Yes	Yes
FPE	Final Prediction Error	Yes	Yes
MAX	Maximum Absolute Error	Yes	Yes
MISC	Misclassification Rate	Yes	No
MSE	Mean Squared Error	Yes	Yes
NOBS	Sum of Frequencies	Yes	Yes
_NW_	Number of Estimated Weights	Yes	Yes
PROF	Total Profit	Yes	Yes
RASE	Root Average Squared Error	Yes	Yes
RFPE	Root Final Prediction Error	Yes	Yes
RMSE	Root Mean Squared Error	Yes	Yes
SBC	Schwarz's Bayesian Criterion	Yes	Yes
SSE	Sum of Squared Errors	Yes	Yes
SUMW	Sum of Case Weights Times Frequency	Yes	Yes
WRONG	Number of Wrong Classifications	Yes	No

Partial Least Squares Node Example

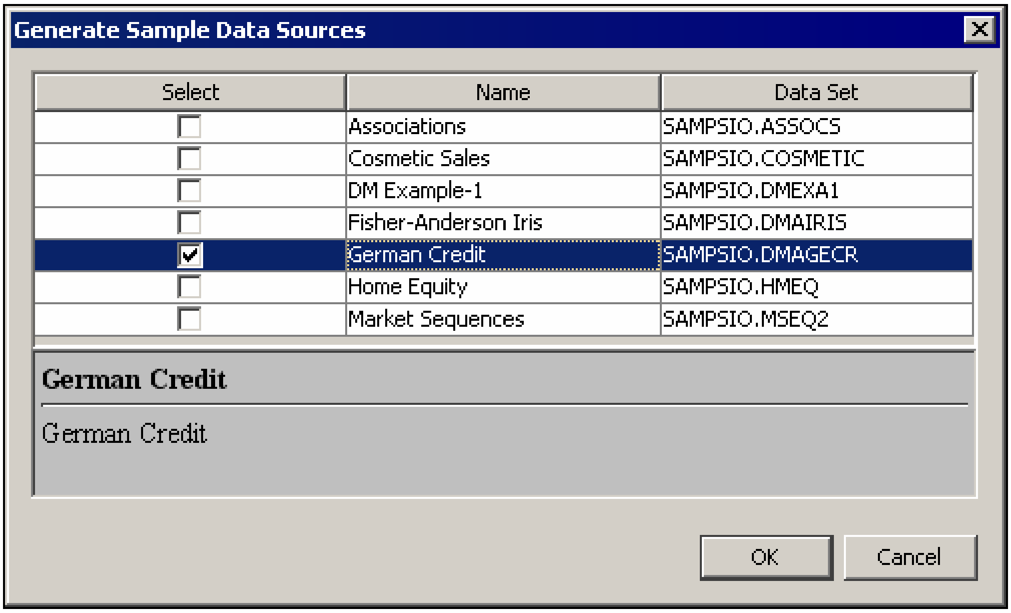
This example performs a partial least squares regression on an example SAS data set composed of training metadata about applicants for credit. The training data has a binary target variable named GOOD_BAD. The GOOD_BAD status indicates whether the individual in the training data defaulted on his loan. The example uses partial least squares regression to create a model that uses a combination of latent vectors to predict the good risks among a pool of credit applicants.

This example will create the process flow diagram shown below.



First, create a new diagram on page 230 to hold this example.

Next, you will need to create the German Credit data set. To create the German Credit data set, select **Help** ⇨ **Generate Sample Data Sources...** from the main menu. Select only the German Credit data set, as shown in the image below.



Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
□	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
☑	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

German Credit

German Credit

Use the **Variables** property to ensure that the following conditions are met:

- The role of the variable GOOD_BAD is **Target** and its **Level** is set to **Binary**.
- The measurement level for the input variable PURPOSE is set to **Nominal**.

Add the German Credit data set to your project diagram workspace. Then, add a Data Partition node, located in the Sample tab, to the diagram workspace.

Specify the following settings in the Data Set Percentages section of the Partition Node properties panel:

- Set the percentage for the **Training** property to 70.0
- Set the percentage for the **Validation** property to 30.0

- Set the percentage for the **Test** property to 0.0

Add a Partial Least Squares node, located in the Model tab, to the diagram.

Specify the following setting in the Variable Selection section of the Partial Least Squares properties panel:

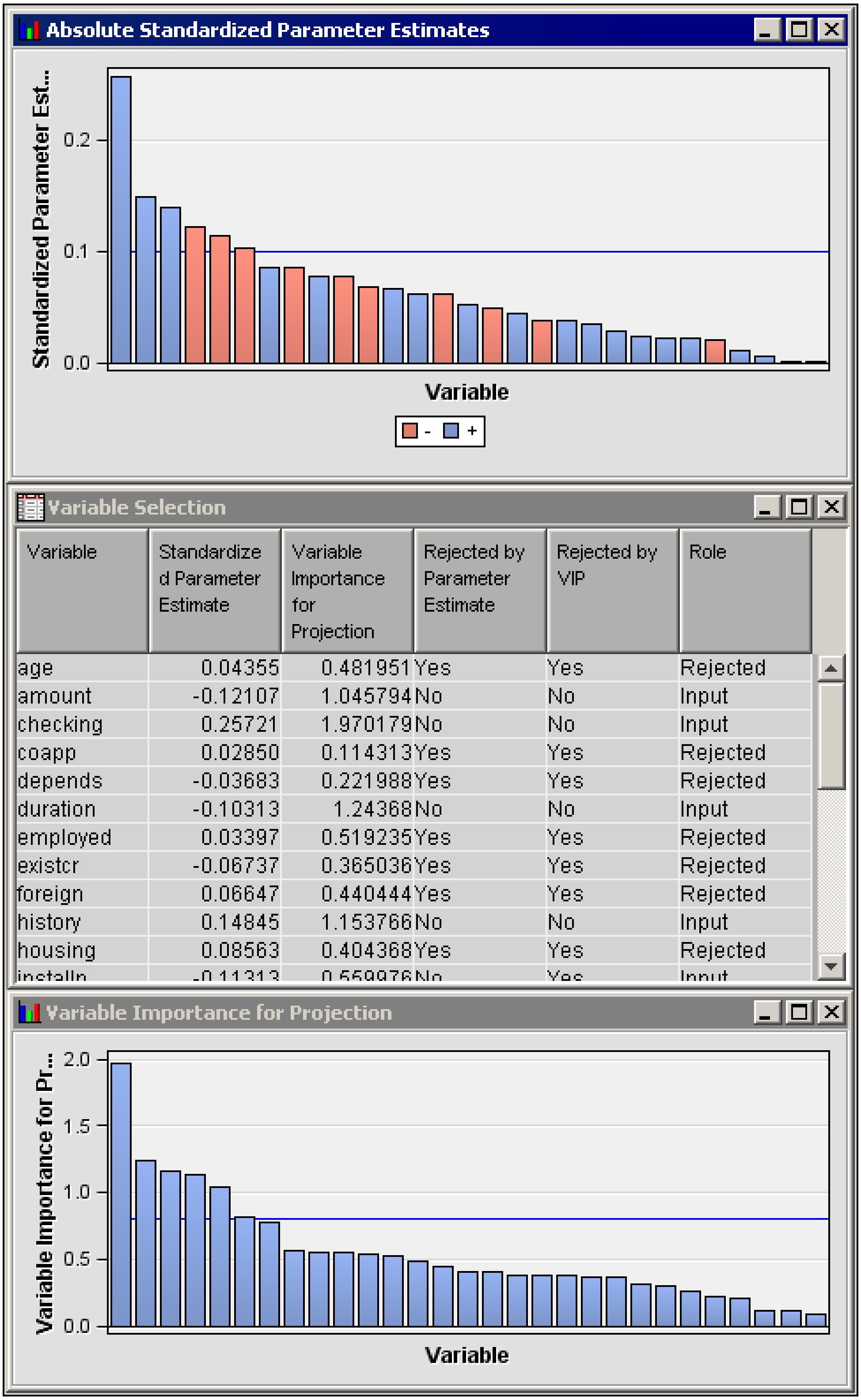
- Set the **Export Selected Variables** property to **Yes**.

The Partial Least Squares node is configured by default to use the NIPALS algorithm to create a partial least squares model. The model generates model weight and loading tables for each variable, according to the number of exported factors. The model also calculates coefficient values for standardized parameter estimates and variable importance scores. The standardized parameter estimates and variable importance scores can be used to select the variables that best explain variation in the target variable GOOD_BAD. The model exports the best predictor variables to a subsequent modeling node, such as a Decision Tree, for further analysis.

In this example, the Variable Selection Criterion setting specifies **Both**. The Both setting means that the model selects variables based on performance criteria from either of two measures. Variables will be selected that have Variable Importance for Projection scores that are greater than the threshold value that is specified in the Partial Least Square node's VIP Cutoff property. Variables will also be selected if their Standardized Parameter Estimate score exceeds the threshold specified in the Para. Est. Cutoff property.

Right-click the Partial Least Squares node and select **Run**. After the process flow diagram completes successfully, select **Results** in the Run Status window.

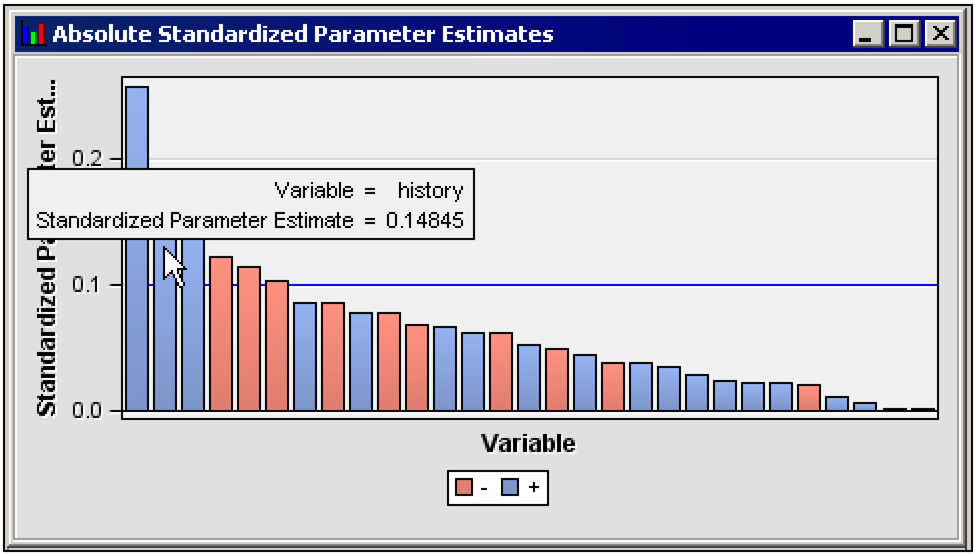
The Partial Least Squares Results window displays bar charts for Absolute Standardized Parameter Estimates and Variable Importance for Projection. The Variable Selection table provides Standardized Parameter Estimate and Variable Importance for Projection scores for each variable. Summary columns indicate whether the variable was rejected, based on both performance measure thresholds. If a variable was not rejected by both of the variable selection criteria, the variable is assigned a Role of Input and exported to the successor node.


Absolute Standardized Parameter Estimates
Standardized Parameter Est...
0.2
0.1
0.0
Variable
🟥 - 🟦 +

Variable	Standardized Parameter Estimate	Variable Importance for Projection	Rejected by Parameter Estimate	Rejected by VIP	Role
age	0.04355	0.481951	Yes	Yes	Rejected
amount	-0.12107	1.045794	No	No	Input
checking	0.25721	1.970179	No	No	Input
coapp	0.02850	0.114313	Yes	Yes	Rejected
depends	-0.03683	0.221988	Yes	Yes	Rejected
duration	-0.10313	1.24368	No	No	Input
employed	0.03397	0.519235	Yes	Yes	Rejected
existcr	-0.06737	0.365036	Yes	Yes	Rejected
foreign	0.06647	0.440444	Yes	Yes	Rejected
history	0.14845	1.153766	No	No	Input
housing	0.08563	0.404368	Yes	Yes	Rejected
installp	-0.11313	0.559976	No	Yes	Input

Variable Importance for Projection
Variable Importance for Pr...
2.0
1.5
1.0
0.5
0.0
Variable

In the Variable Selection table of the Partial Least Squares Results window, click on the Role column heading to sort it alphabetically. View the grouping of variables with a Role of Input, then drag your mouse pointer down the rows to highlight all of them. As each variable is highlighted, its corresponding measurement graphic in the Absolute Standardized Parameter Estimates and Variable Importance for Projection is also highlighted. You can select and deselect variables in the table to find their corresponding plot graphics, or you can hover your mouse pointer over a plot graphic to view its pop-up information.

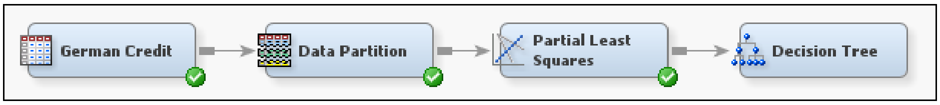


In the sorted Variable Selection table, the following variables from the input data set SAMPSIO.DMAGECR are selected for use as input variables:

- amount
- checking
- duration
- history
- installp
- purpose3
- savings

Each of the selected input variables exceeded the minimum threshold value for at least one of the two specified Variable Selection performance measures.

You can verify that the Partial Least Squares node exports the correct variable by adding and connecting a successor node to the diagram, such as the Decision Tree node shown below.



It is not necessary to run the newly added node. After you connect the successor node, right-click the node and select Edit Variables. When the Edit Variables window of the successor node opens, it will display the variables that the Partial Least Squares node exported. Here we can see the variables that the Decision Tree imports are indeed the same variables that the Partial Least Squares node chose to export as sources of variation in predicted target variable values. Subsequent modeling nodes will further investigate relationships between the input and target variables.

Chapter 61
Regression Node

Regression Node

Overview of the Regression Node

The Regression node belongs to the Model category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA).

You use the Regression node to fit both linear and logistic regression models to a predecessor data set in a SAS Enterprise Miner process flow. Linear regression attempts to predict the value of an interval target as a linear function of one or more independent inputs. Logistic regression attempts to predict the probability that a binary or ordinal target will acquire the event of interest as a function of one or more independent inputs.

You can also use the Neural Network node to build regression models. In this case, you should configure the network to have direct connections between the input units and the output unit(s), without including any hidden units. Direct connections define linear layers, whereas hidden neurons define nonlinear layers. The Neural Network node supports more link functions (such as, identity, logit, log, square root, and reciprocal), and more error functions (such as, normal, poison, and gamma) than does the Regression node, as well as robust estimation capabilities (such as, Cauchy, Logistic, and Huber).

The Regression node uses an identity link function and a normal distribution error function for linear regression. The Regression node uses either a logit, complementary log-log, or probit link function and a binomial distribution error function for a logistic regression analysis. A disadvantage in using the Neural Network node for a regression analysis is that it does not provide p-values for testing the significance of the parameter estimates.

Before running the Regression node, you must use an Input Data node to designate the input (right-hand or independent) variables for the input data set, and target (left-hand or response) variable (or variables).

The Regression node supports binary, interval, nominal, and ordinal target variables. An example of a binary target variable is "purchase" or "no-purchase", which is often used for modeling customer profiles. An example of an interval target variable is value of purchase, which is useful for modeling the best customers for particular products, catalogs, or sales campaigns. An example of an ordinal target is sales volume, which might contain a few discrete values such as "low", "medium", and "high".

Your input variables can be continuous (interval) or discrete (binary, nominal, or ordinal). If you defined more than one target variable, then the Target Selector window opens when you open the node, prompting you to select a single target variable for modeling.

The Regression node supports forward, backward, and stepwise selection methods. Data sets that have a role of score are automatically scored when you train the model.

See the Predictive Modeling section for information that applies to all of the predictive modeling nodes.

Regression Node Data Set Requirements

The input data should have the following data structure:

- One observation per customer.
- An interval, ordinal, or binary target (response) variable. The Regression node does not support nominal targets with more than two levels. You can model more than one target variable by incorporating a Group Processing node into the process flow, but the Regression node does not support multivariate analyses.
- Input variables, which can contain Id (identification) variables, demographic data, history of previous purchases, and so forth.

The data set can also have cross-sectional data, which are data collected across multiple customers, products, geographic regions, and so on, but typically not across multiple time periods.

Customer	Purchase	Amount	Salary	Age	Martial	Own-Rent
John Doe	1	500	50,000	31	Single	1
Jean Hall	1	2000	56,000	28	Married	2
Mark Price	0	.	68,000	33	Married	2
Cathy Harp	0	0	22,000	43	Single	1

Before you build a regression model with the Regression node, you may need to perform one or more of the following data mining tasks:

- Sample the Input Data Source — Use the Sampling node to create a sample from your input data source. Sampling is recommended for extremely large databases because it can tremendously decrease model training time. If the sample is sufficiently representative, then relationships found in the sample can be expected to generalize the complete data set.
- Create Partitioned Data Sets — Use the Data Partition node to split the sample into training, validation, and test data sets. The training data set is used to calculate the regression equation. The validation data set can be used to fine-tune stepwise regression models (prevent the models from over-fitting the training data). The validation data set is also used by default for model assessment. The test data set can be used to obtain an unbiased estimate of the generalization error of a model.
- Use Only the Important Variables — When your data set has a large number of inputs, it is tempting to use most or all of the inputs. However, it is often better to use only a small number of important inputs, which can greatly reduce the time that is required to train the regression model as well as improve the prediction results. If you know from your business expertise that an input is not useful in predicting the target, then exclude it from the regression analysis. The Multiplot node enables you to create exploratory plots that can help you identify important inputs (predictors). You can also use the Variable Selection node to reject inputs that are unrelated to the target.
- Transform Data and Filter Outliers — The regression parameter estimates tend to be more stable and produce better predicted values when an appropriate transformation (for example, taking the log of an input to stabilize its variance) and/or filtering method is applied to noisy inputs. The Transform Variables node enables you to apply several transformations to a variable, such as log, exponential, square root, inverse, and square.
- Use Other Modeling Nodes — For some problems, a traditional regression model may not be the best analytical tool. Greater predictive accuracy may be achieved using either the Tree node or the Neural Network node. Cross-model assessment can be performed with the Model Control node to help you choose the best model for scoring new data.

Regression Node Properties

Regression Node General Properties

The following general properties are associated with the Regression node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Regression node that is added to a diagram will have a Node ID of Reg. The second Regression node added to a diagram will have a Node ID of Reg2, and so on.
- **Imported Data** — accesses the Imported Data — Regression window. The Imported Data — Regression window contains a list of the ports that provide data sources to the Regression node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.

- **Exported Data** — accesses the Exported Data — Regression window. The Exported Data — Regression window contains a list of the output data ports that the Regression node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Regression Node Train Properties

- **Variables** — Use the Variables property of the Regression node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable Use and Report values to indicate whether to use the variable or include it in the report.

Regression Node Train Properties: Equation

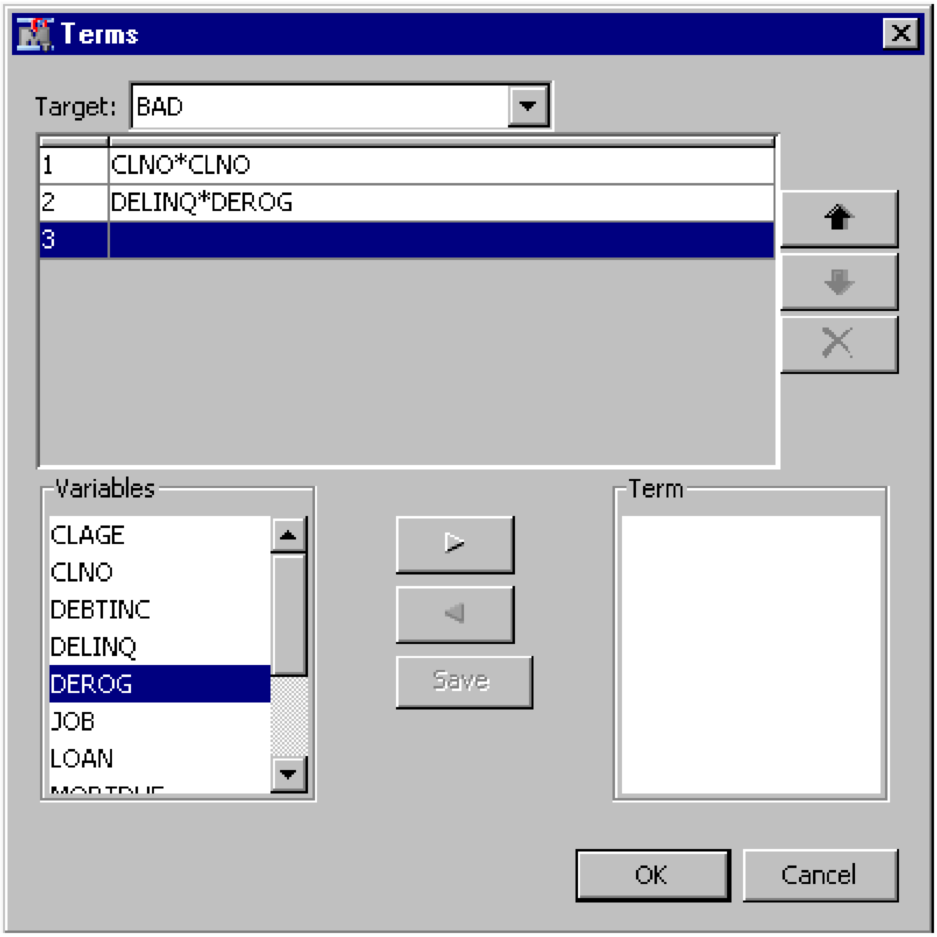
Users can support two-factor interactions, polynomial terms, and terms specified in an interaction data set in regression analysis by using the Equation properties.

The following properties are available:

- **Main Effects** — Set the Main Effects property of the Regression node to **No** if you want to suppress the input and rejected variables with status of Use in the regression analysis. The default setting for this property is **Yes**.
- **Two-Factor Interactions** — Set the Two-Factor Interactions property of the Regression node to **Yes** if you want to include all two-factor interactions for class variables that have a status of Use. The default setting for this property is **No**.
- **Polynomial Terms** — Set the Polynomial Terms property of the Regression node to **Yes** if you want to include polynomial terms for interval variables with status of Use in the regression analysis. When this property is set to **Yes**, you must specify an integer value for the Polynomial Degree property. The default setting for Polynomial Terms is **No**.
- **Polynomial Degree** — When the Polynomial Terms property of the Regression node is set to **Yes**, use the Polynomial Degree property to specify the highest degree of polynomial terms (for interval variables with status set to **Use**) to be included in the regression analysis. The Polynomial Degree property can be set to 2 or 3.

- **User Terms** — Set the User Terms property of the Regression node to **Yes** if you want to create a Terms data set using the Term Editor. The default state for this Boolean property is **No**.
- **Term Editor** — First set the User Terms property to **Yes** and then click the ... button in the Term Editor property to open the Terms Window. Use the Term Editor to order and specify variable interaction terms in a data set for regression analysis. The Term Editor supports two factor interactions for class variables and polynomial terms for interval variables. Only variables with a status of **Use** are displayed in the Term Editor.

  To create an interaction term, select the applicable target variable from the Target drop-down list, and then use the arrows to transfer the interaction term variable components from the Variables to the Terms list. Click the Save button to create the interaction term in the upper table. Populate the Terms list with the same variable twice to create degreed polynomial terms such as (CLNO).



Target: BAD

1	CLNO*CLNO
2	DELINQ*DEROG
3	

If you have more than one interaction term, you can order the terms by using the arrows to the right of the interaction terms list. Use the delete X button beneath the arrows to remove interaction terms from the list.

If all Boolean Regression Node Equation properties are set to No, then a simple intercept model is used.

For more information, see “Equation Examples” on page 927 .

Regression Node Train Properties: Class Targets

- **Regression Type** — Use the Regression Type property of the Regression node to specify the type of regression that you want to run.
  - **Logistic Regression** — the default regression type for binary or ordinal targets. Logistic regression attempts to predict the probability that a binary or ordinal target will acquire the event of interest as a function of one or more independent inputs. For binary or ordinal targets, the default type is Logistic Regression.
  - **Linear Regression** — the default regression type for interval targets. Linear regression attempts to predict the value of a continuous target as a linear function of one or more independent inputs. For interval targets, the default type is Linear Regression.
- **Link Function** — Use the Link Function property of the Regression node to specify the link function that you want to use in your regression analysis. Link functions link the response mean to the linear predictor. In a linear regression, the identity link function g(M) = Xβ is used.

  In a Logistic regression, you can select one of the link functions:
  - **Cloglog** — Specifies the complementary log-log function which is the inverse of the cumulative extreme-value function.

    $$g\left(M\right) = log\left(-log\left(1 - M\right)\right)$$
  - **Logit** — (default) Specifies the inverse of the cumulative logistic distribution function.

    $$g\left(M\right) = log\left(\frac{M}{1-M}\right)$$
  - **Probit** — Specifies the inverse of the cumulative standard normal distribution.

    $$g(M) = 1/\Theta(M)$$

Regression Node Train Properties: Model Options

- **Suppress Intercept** — Set the Suppress Intercept property of the Regression node to **Yes** to suppress intercepts when you are coding class variables. The Suppress Intercept property is ignored for ordinal targets. The default setting for the Suppress Intercept property is **No**.
- **Input Coding** — Use the Input Coding property of the Regression node to specify the method that you want to use to code class variables. For more information, see "Coding Categorical Variables (Input Coding) with the Regression Node" on page 931 .
  - **GLM** — The non-full-rank General Linear Models (GLM) coding uses parameters to estimate the difference between each level and a reference level. The reference level is the last level when the levels are sorted in ascending numeric or alphabetic order. The last parameter is constrained to be 0 with GLM coding. This coding is also known as dummy coding. The dummy indicator variables for an effect are coded as 1 except for the last level, which is coded as 0. The parameter estimate for a given effect level is equal to the estimate of the effect minus the estimate of the last effect level.
  - **Deviation** — The Deviation from mean coding uses parameters to estimate the difference between each level and the average across each level. The parameters

for all levels are constrained to sum to 0. This coding is also known as effects coding. Deviation is the default coding method.

Regression Node Train Properties: Model Selection

- **Selection Model** — Use the Selection Model property of the Regression node to specify the model selection method that you want to use during training. For more information, see "Regression Node Model Selection Methods" on page 928 .

  You can choose from the following effect selection methods:

  - **Backward** — begins with all candidate effects in the model and removes effects until the Stay Significance Level or the Stop Criterion is met.
  - **Forward** — begins with no candidate effects in the model and adds effects until the Entry Significance Level or the Stop Criterion is met.
  - **Stepwise** — begins as in the forward model but may remove effects already in the model. Continues until Stay Significance Level or Stepwise Stopping Criteria are met.
  - **None** — (default setting) all inputs are used to fit the model.

- **Selection Criterion** — If you choose the forward, backward, or stepwise selection method for the Selection Model property, you should also set the Selection Criteria property to specify the criterion for choosing the final model. The available criteria are

  - **Default** — Uses Profit/Loss with training data, Profit/Loss with validation data, and None with raw or test data.
  - **None** — Chooses standard variable selection based on the entry and/or stay p-values.
  - **Akaike's Information Criterion (AIC)** — The model with the smallest criterion value is chosen.
  - **Schwarz's Bayesian Criterion (SBC)** — The model with the smallest criterion value is chosen.
  - **Validation Error** — the error rate for the validation data set. The error is the sum of square errors for least-square regression and negative log-likelihood for logistic regression. The model with the smallest error rate is chosen.
  - **Validation Misclassification** — the misclassification rate for the validation data set. The model with the smallest misclassification rate is chosen.
  - **Cross-Validation Error** — the error rate for cross validation. The error is the sum of square errors for least-square regression and negative log-likelihood for logistic regression. The model with the smallest error rate is chosen.
  - **Cross-Validation Misclassification** — the misclassification rate for cross validation. The model with the smallest misclassification rate is chosen.
  - **Profit/Loss** — Profit/Loss with training data.
  - **Validation Profit/Loss** — Profit/Loss with validation data.
  - **Cross Validation Profit/Loss** — Cross Validation Profit/Loss.

  Note: To use a Profit/Loss criterion, you must define a profit or loss matrix in the target profile for the target. For more detailed information, see "Regression Node Model Selection Criteria" on page 929 .

- **Use Selection Defaults** — Set the Use Selection Default property of the Regression node to **No** to use non-default values for your model selection criteria, such as stopping number of variables, stepwise stopping criteria, and entry or stay significance levels. You must also set the corresponding individual selection criteria properties to your user-defined values. The default setting for the Selection Default property is **Yes**.

- **Selection Options** — Click the ... button to open the Selection Options window.

  Use the Selection Options window to specify the following optional user-defined model selection criteria:

  - **Sequential Order** — When set to **Yes**, the Sequential Order property of the Regression node adds or removes variables during effect selection, using the sequential effect order specified in the model statement. The default setting for the Sequential Order property is No. For more information, see "Effect Hierarchy" on page 933 .
  - **Entry Significance Level** — If the Selection Default property is set to **No**, you can use the Entry Significance Level property of the Regression node to specify the Entry Significance Level setting that you want to use to add variables in forward or stepwise regressions. The default value for the Entry Significance Level is 0.05. Values must be between 0 and 1.
  - **Stay Significance Level** — If the Selection Default property is set to **No**, you can use the Stay Significance Level property of the Regression node to specify the significance level you want to use when removing variables during backward or stepwise regression. The default value for the Stay Significance Level is 0.05. Values must be between 0 and 1.
  - **Start Variable Number** — If the Selection Default property is set to **No**, you can use the Start Variable Number property of the Regression node to specify the number of effects that you want to use in the first model of the model selection process. The start value selects the first n effects from the beginning of the effects list as the first model. The model selection methods use the following default values:

Method	Default Start Value
Forward	0
Stepwise	0
Backward	total number of candidate effects

  - **Stop Variable Number** — If the Selection Default property is set to **No**, you can use the Stop Variable property of the Regression node to specify the stopping threshold integer n for the number of effects during effect selection. Thresholds vary by the selection method used.

    Note: Effect selection may terminate before Stop values apply due to other criteria. The number of effects appear in the model using a Backward or Forward selection process. The following table shows the definitions of and the default values of the Stop value for the Backward and Forward methods. The Stop option is not used in the Stepwise selection method.

Method	Definition	Default Stop Value
Forward	The maximum number of effects to appear in the final model.	total number of input variables
Backward	The minimum number of effects to appear in the final model.	0

- **Force Candidate Effects** — Use the Force Candidate Effects property of the Regression node to enter the number of variables that you want to force into all candidate models. Use the Force Candidate Effects property if you decide in advance that one or more candidate effects is important in predicting the target. The default setting for the Force Candidate Effects property is 0. For more information on Hierarchy Effects, see “Effect Hierarchy” on page 933 .
- **Hierarchy Effects** — Use the Hierarchy Effects property of the Regression node to specify the variable types that hierarchy rules apply to during the effect selection process. The choices are
  - **All** — the hierarchy rule applies to all independent variables
  - **Class** — The hierarchy rule only applies to class variables. Class is the default setting.
- **Moving Effect Rule** — Use the Moving Effect Rule property of the Regression node to determine whether hierarchy is maintained and whether single or multiple effects enter or leave the model in one step. For more information on the Moving Effect Rule, see “Effect Hierarchy” on page 933 .

  You can choose from the following values:
  - **None** — hierarchy is not maintained. Single effects enter and leave the model. None is the default setting.
  - **Single** — hierarchy is maintained and single effects enter and leave the model.
  - **Multiple** — hierarchy is maintained and more than one effect can enter and leave the model, subject to hierarchy.
- **Maximum Number of Steps** — If the Selection Default property is set to **No**, you can use the Maximum Number of Steps property of the Regression node to specify, n, the maximum number of steps that you want to allow during the stepwise model effect selection process. The default setting for Maximum Number of Steps is 0.

Regression Node Train Properties: Optimization Options

- **Technique** — Use the Technique property of the Regression node to specify the optimization technique that you want to use while fitting your model. For more detailed information, see “Regression Node Optimization Techniques” on page 930 .
  - **Default** — the default method varies and depends on the number of weights applied at execution.
  - **Congra** — conjugate gradient optimization technique, for large data mining problems with over 500 parameters.

  - **Dbldog** — Double Dogleg optimization technique.
  - **Newrap** — Newton-Raphson with Line Search optimization technique.
  - **Nrridg** — Newton-Raphson with Ridging optimization technique.
  - **Quanew** — Quasi-Newton optimization technique, for medium optimization problems up to 500 parameters.
  - **Trureg** — Trust-Region optimization technique, for small and medium optimization problems up to 40 parameters.
- **Default Optimization** — Use the Default Optimization property of the Regression node to indicate whether you want to use model default optimization. When the Default Optimization property is set to Yes, then any settings made in the Max Iterations, Max Function Calls, and Max CPU Time properties are ignored. The default setting of the Default Optimization property is **Yes**.
- **Max Iterations** — Use the Max Iterations property of the Regression node to specify the maximum number of iterations to be used in the optimization technique. See “Regression Node Optimization Techniques” on page 930 to view the maximum number of iterations allowed for each method. The Default Optimization property must be set to **No** in order to use the Max Iterations property. The default setting for the Max Iterations property varies according to the selected optimization technique:

Optimization Technique	Default Max Iterations
Default	0
Congra	400
Dbldog	200
Newrap	50
Nrridg	50
Quanew	200
Trureg	50

- **Max Function Calls** — Use the Max Function calls property of the Regression node to specify the maximum number of function calls allowed in the optimization technique. See “Regression Node Optimization Techniques” on page 930 to view the maximum number of function calls allowed for each method. The Default Optimization property must be set to **No** in order to use the Max Function Calls property. The default setting for the Max Function Calls property is 0.

Optimization Technique	Default Max Function Calls
Default	0
Congra	1000
Dbldog	500

Optimization Technique	Default Max Function Calls
Newrap	125
Nrridg	125
Quanew	500
Trureg	125

- **Maximum Time** — Use the Maximum Time property to specify the maximum amount of CPU time that you want to dedicate to the neural optimization process. Permissible values are **5 minutes**, **10 minutes**, **30 minutes**, **1 hour**, **2 hours**, **4 hours**, **7 hours**, **1 day**, or **7 days**. The default setting for the Maximum Time property is **1 hour**.

  Note: The **Maximum Time** property governs only the neural optimization process time for the Regression node. It does not govern the maximum overall execution time for the Regression node.

Regression Node Train Properties: Convergence Criteria

- **Uses Defaults** — Use the Default convergence property of the Regression node to indicate whether you want to use the default convergence criterion values that are associated with your selection model. If you want to specify your own convergence criterion values, set the Default convergence property to **No** and use the Convergence Criteria Options property to specify your own convergence criterion values. The default setting for the Uses Defaults convergence property is **Yes**.
- **Options** — When the Uses Defaults property is set to No, select the ... button to open the Options window.

  Use the Options window to specify the following optional user-defined convergence criteria:

  - **Absolute** — Use the Absolute convergence property of the Regression node to specify an absolute convergence criterion. The Absolute convergence is a function of the log-likelihood for the intercept-only model. The optimization is to maximize the log-likelihood. The default value is -1.34078E154.
  - **Absolute Function** — Use the Absolute Function property of the Regression node to specify an absolute function convergence criterion. Absolute Function is a function of the log-likelihood for the intercept-only model. The default value is 0.
  - **Absolute Function Times** — Use the Absolute Function Times property of the Regression node to specify the number of successive iterations for which the absolute function convergence criterion must be satisfied before the process can be terminated. The minimum value and the default value are both 1.
  - **Absolute Gradient** — Use the Absolute Gradient property of the Regression node to specify the absolute gradient convergence criterion that you want to use. All values must be larger than 0. The default value is 1.0E-5.
  - **Absolute Gradient Times** — Use the Absolute Gradient Times property of the Regression node to specify the number of successive iterations for which the

absolute gradient convergence criterion must be satisfied before the process can be terminated. The default value is 1.

- **Absolute Parameter** — Use the Absolute Parameter property of the Regression node to specify the absolute parameter convergence criterion. The default value is 1.0E-8. The value must be greater than 0.
- **Absolute Parameter Times** — Use the Absolute Parameter Times property of the Regression node to specify the number of successive iterations for which the absolute parameter convergence criterion must be satisfied before the process can be terminated. The default value is 1.
- **Relative Function** — Use the Relative Function property of the Regression node to specify the relative function convergence criterion. The default value is 0.0.
- **Relative Function Times** — Use the Relative Function Times property of the Regression node to specify the number of successive iterations for which the relative function convergence criterion must be satisfied before the process can be terminated. The default value is 1.
- **Relative Gradient** — use the Relative Gradient property of the Regression node to specify the relative gradient convergence criterion. The default value is 1.0E-6.
- **Relative Gradient Times** — Use the Relative Gradient Times property of the Regression node to specify the number of successive iterations for which the relative gradient convergence criterion must be satisfied before the process can be terminated. The default value is 1.

Regression Node Train Properties: Output Options

- **Confidence Limits** — Set the Confidence Limits property of the Regression node to **Yes** if you want to generate confidence limits for parameter estimates. The default setting for the Confidence Limits property is **No**.
- **Save Covariance** — Set the Save Covariance property to **Yes** if you want to save the covariance matrix of parameter estimates in the OUTEST data set. This data set contains the parameter estimates and the fit statistics. Note that if a model selection method is used, then the covariance matrix is saved for each iteration.
- **Covariance** — Set the Covariance property of the Regression node to **Yes** if you want to print the covariance matrix of parameter estimates. The default setting for this property is **No**.
- **Correlation** — Set the Correlation property of the Regression node to **Yes** if you want to print the correlation matrix of parameter estimates. The default setting for this property is **No**.
- **Statistics** — Set the Statistics property of the Regression node to **Yes** if you want to print simple descriptive statistics of input variables. The default setting for this property is **No**.
- **Suppress Output** — Set the Suppress Output property of the Regression node to **Yes** if you want to suppress all of the printed output. The default setting for this property is **No**.
- **Details** — Set the Details property of the Regression node to **Yes** if you want to print details at each model selection step. The default setting for this property is **No**.
- **Design Matrix** — Set the Design Matrix property of the Regression node to **Yes** if you want to print the coded input for class variables. The default setting for this property is **No**.

Regression Node Score Properties

The following score property is available with the Regression Node:

- **Excluded Variables** — Use the Excluded variables to specify what action should be taken with variables that are excluded from the final model. The Excluded Variables property only applies when using a variable selection method.
  - **None** — the role of excluded variables remains unchanged.
  - **Hide** — the excluded variables are dropped from the metadata that is exported by the node.
  - **Reject** — the role of excluded variables is set to Rejected.

Regression Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Equation Examples

For the first example, suppose that A and B are class variables, and that X1 and X2 are interval variables. If you specify the properties below, then the model fitted will be A*B, X1, A*X1.

- Main Effects — No
- Two-factor Interactions — Yes
- Polynomial — No
- User Terms — Yes
- The Terms data set contains the terms X1, A*X1

For the second example, suppose that A and B are class variables, and that X1 and X2 are interval variables. If you specify the properties below, then the model fitted will be A*B, X1*X1, X2*X2.

- Main Effects — No
- Two-factor Interactions — Yes
- Polynomial — Yes
- Polynomial Degree — 2
- User Terms — No

Regression Node Model Selection Methods

The Regression node can select a model from a set of candidate terms using one of several methods and criteria. The terms in a regression model are also called effects. For instance, a model might contain (in addition to an intercept) the effects A, B, A*B, and A*A. Effects A and B are known as main effects. A*B is called an interaction effect. A*A is a polynomial effect. In the Regression node, you can specify a set of candidate effects, such as the four above, and the node can select a model from the candidates. At each step in the selection method, an effect is added or deleted from the model so that a sequence of models is generated — one at each step. The final model is chosen from the candidate models by determining which model best optimizes the criterion you specified.

Use the Selection Model property to specify the effect selection method that you want to use during training. For more complete definitions of the selection methods, see pages 1397-1398 of the "*SAS/STAT User's Guide, Volume 2, GLM-VARCOMP, Version 6 edition*".

You can choose from the following effect selection methods:

- **Backward** — begins, by default, with all candidate effects in the model and then systematically removes effects that are not significantly associated with the target until no other effect in the model meets the Stay Sig. Level or until the Stop Variable Number that you specified is met. This method is not recommended when the target is binary or ordinal and there are many candidate effects or many levels for some classification input variables.
- **Forward** — begins, by default, with no candidate effects in the model and then systematically adds effects that are significantly associated with the target until none of the remaining effects meet the Entry Sig. Level or until the Stop Variable Number criterion is met.
- **Stepwise** — As in the Forward method, Stepwise selection begins, by default, with no candidate effects in the model and then systematically adds effects that are significantly associated with the target. However, after an effect is added to the model, Stepwise may remove any effect already in the model that is not significantly associated with the target.

  This stepwise process continues until one of the following occurs:

  - No other effect in the model meets the Stay Significance Level.
  - The Max Steps criterion is met. If you choose the Stepwise selection method, then you can specify a Max Steps to put a limit on the number of steps before the effect selection process stops. The default value is set to the number of effects in the model. If you add interactions via the Interaction Builder, the Max Steps is automatically updated to include these terms.
  - An effect added in one step is the only effect deleted in the next step.
- **None** — (default) all candidate effects are included in the final model. If you choose None as the selection method, then the properties related to the selection method have no effect.

Stepwise selection of inputs requires more computing time than does Forward or Backward selection, but it has an advantage in terms of the number of potential subset models checked before the model for each subset size is decided.

Regression Node Model Selection Criteria

If you choose the Forward, Backward, or Stepwise effect selection method, then you can specify a selection criterion to be used to select the final model. When you specify one of the selection criteria below, the node first performs the effect selection process which generates a set of candidate models corresponding to each step in the process. Effect selection is done based on the Entry and/or Stay significance Levels. Once the effect selection process terminates, the candidate model that optimizes the selection criterion is chosen as the final model.

You can choose a model selection criterion from the following choices in the Selection Criteria property.

- **None** — (default if you did not define a profit or loss matrix with two or more decisions) the last model produced by the effects selection method is chosen as the final model.
- **AIC** — Akaike's Information Criterion = n*ln(SSE/n) + 2p , where n is the number of cases, SSE is error sum of squares, and p is the number of model parameters. The model with the smallest AIC value is chosen.
- **SBC** — Schwarz's Bayesian Criterion = n*ln(SSE/n) + p*ln(n). The model with the smallest SBC value is chosen.
- **Validation Error** — chooses the model that has the smallest error rate for the validation data set. For logistic regression models, the error is the negative log-likelihood. For linear regression, the error is the error sum of squares (SSE). This option is grayed out if a validation predecessor data set is not input to the Regression node.
- **Validation Misclassification** — chooses the model that has the smallest misclassification rate for the validation data set. This option is unavailable and appears dimmed if a validation predecessor data set is not input to the Regression node.
- **Cross-Validation Error** — this criterion chooses the model that has the smallest cross validation error rate for the training data set. For logistic regression models, the error is the negative log-likelihood. For linear regression, the error is the error sum of squares.
- **Cross Validation Misclassification** — this criterion chooses the model that has the smallest cross validation misclassification rate for the training data set.
- **Profit/Loss** — The node chooses the model that maximizes the profit or minimizes the loss for the cases in the training data set. If the decision matrix contains less than two decisions, then the profit or loss information is not considered in the model selection process. When there are less than two decisions, the None criterion is actually used for model selection. To use the Profit/Loss criterion, you must define a profit or loss matrix in the target profile for the target. The profit or loss values are adjusted for the prior probabilities that you specify in the prior vector of the target profile. To learn more about defining a target profile, read "Layout of a Target Profile" on page 208 .
- **Validation Profit/Loss** — The node chooses the model that maximizes the profit or minimizes the loss for the cases in the validation data set. If the decision matrix contains less than two decisions, then the profit or loss information is not considered in the model selection process. When there are less than two decisions, the None criterion is actually used for model selection. To use the Profit/Loss criterion, you must define a profit or loss matrix in the target profile for the target. The profit or

loss values are adjusted for the prior probabilities that you specify in the prior vector of the target profile. To learn more about defining a target profile, read "Layout of a Target Profile" on page 208 .

- **Cross Validation Profit/Loss** — this criterion chooses the model that maximizes the cross validation profit or minimizes the cross validation loss. You must also define a profit or loss matrix for the target to use this criterion.

Note: None of the cross-validation methods are valid when building a logistic regression model for an ordinal target. Cross-validation is a way to remove the optimistic bias resulting from using observations both to train the model and to evaluate it. It is commonly used when there is not enough data available to split into separate training and validation data sets. Cross-validation approximates a process in which an observation is left out of the training data set, the model is trained, and then the held-out observation is evaluated. Each observation in the training data set is held out and evaluated. The misclassification rate (or profit / loss) is then computed from the evaluations of all the held-out observations.

Regression Node Optimization Techniques

The following table provides a list of the general nonlinear optimization methods that the Regression node uses and the default maximum number of iterations and function calls for each method.

Optimization Method	Max Iterations	Max Function Calls
Conjugate Gradient	400	1000
Double Dogleg	200	500
Newton-Raphson with Line Search	50	125
Newton-Raphson with Ridging	50	125
Quasi-Newton	200	500
Trust-Region	50	125

You should set the optimization method based on the size of the data mining problem.

- Small to Medium Problems — Trust-Region, Newton-Raphson with Ridging, and Newton-Raphson with Line Search are appropriate for small and medium-sized optimization problems (number of model parameters up to 40) where the Hessian matrix is easy and cheap to compute. Sometimes Newton-Raphson with Ridging can be faster than Trust-Region, but Trust-Region is numerically more stable. If the Hessian matrix is not singular at the optimum, then the Newton-Raphson with Line Search can be a very competitive method.

- Medium Problems — The Quasi-Newton and Double Dogleg methods are appropriate for medium optimization problems (number of model parameters up to 400) where the objective function and the gradient are much faster to compute than the Hessian. Quasi-Newton and Double Dogleg require more iterations than the Trust-Region or the Newton-Raphson methods, but each iteration is much faster.

- Large Problems — The Conjugate Gradient method is appropriate for large data mining problems (number of model parameters greater than 400) where the objective function and the gradient are much faster to compute than the Hessian matrix and where they need too much memory to store the approximate Hessian matrix.

To learn more about these optimization methods, see *SAS/OR Technical Report: The NLP Procedure*.

The underlying default optimization method depends on the number of parameters in the model. If the number of parameters is less than or equal to 40, then the default method is set to Newton-Raphson with Ridging. If the number of parameters is greater than 40 and less than 400, then the default method is set to Quasi-Newton. If the number of parameters is greater than 400, then Conjugate Gradient is the default method.

To set a different optimization method, select the value field beside the Optimization Technique property and then select the desired method from the list.

The maximum number of iterations and function calls depends on the optimization method (see the table above). To change the default maximum number of iterations or function calls, set the Default Optimization property to No. Then, enter the number of iterations or function calls in the Max Iterations or Max Function Calls value field, respectively.

If the optimization is not finished by the time specified in Maximum Optimization CPU Time then:

- the iterations stop
- the node issues a warning message
- the node uses the estimate from the last complete iteration to continue its regression analysis

Note: For stepwise regression analysis, the node restarts the CPU timer for each new model that it fits.

The convergence criteria is identical to the criteria used by the Neural Network node. To change the convergence criteria, enter a different value in the desired entry field.

Setting the Min Resource Usage property to Yes specifies that only minimal resources will be used to fit a logistic regression model. Memory for the Hessian matrix is not needed. The optimization defaults to the conjugate gradient technique and standard errors of the regression parameters are not computed. This option can be useful for nominal targets, where it is common for the model to have a large number of parameters. It does not apply to the normal error regression models.

Note: You cannot use a model selection method in conjunction with the Min Resource Usage option.

Coding Categorical Variables (Input Coding) with the Regression Node

You use the Input Coding property setting to specify the coding method that you want to use with class variables.

The following examples show how the three levels of a nominal (or ordinal) input called JOB are coded for analysis by the two coding methods. The first example illustrates deviation coding:

Table 61.1 Deviation Coding

Level	Job Clerical	Job Lawyer
Clerical	1	0
Lawyer	0	1
Paralegal	–1	–1

The parameter estimate for Job Clerical measures the difference in effect between the Clerical level and the average of all (Clerical, Lawyer, and Paralegal) levels. The parameter estimate for Job Lawyer measures the difference in effect between the Lawyer level and the average of all levels. Because there are only two degrees of freedom for Job, only two parameters are estimated. Since the Deviation coding constrains the parameters for all levels to sum to zero, you can obtain the estimate of the difference in effect between the last level (Paralegal) and the average of all levels by computing the negative sum of the Clerical and Lawyer parameter estimates.

The next example illustrates GLM coding:

Level	Job Clerical	Job Lawyer	Job Paralegal
Clerical	1	0	0
Lawyer	0	1	0
Paralegal	0	0	1

In this example, the parameter estimate for Job Clerical measures the difference in effect between the Clerical level and the Paralegal category. The parameter estimate for Job Lawyer measures the difference in effect between the Lawyer level and the Paralegal category. The parameter estimate for Job Paralegal is set to zero.

For more information about input coding, see *SAS System for Linear Models, Third Edition*.

By default, the intercept is not suppressed. To suppress the intercept, select the Yes for the Suppress Intercept property. Note that you cannot suppress the intercept when building a logistic model for an ordinal target.

In summary, the following options are automatically set by the node:

- For all targets, the input coding is set to deviation by default.
- For binary targets, the type of regression is automatically set to logistic, and the link function is set to logit. You have the option to change the type of regression from logistic to linear for binary targets.
- For interval targets, the type of regression is automatically set to linear, and the link function is set to identity. You cannot change the type of regression for interval targets or the link function.

Effect Hierarchy

Model hierarchy refers to the requirement that for any effect in the model, all effects that it contains must also be in the model. For example, in order for the interaction A*B to be in the model, the main effects A and B must also be in the model. The Effect Hierarchy properties enable you to control how a set of effects enters or leaves the model during the effect selection process. For example, assume that you specify A, B and A*B as candidate effects. You can require the model to be hierarchical so that it includes the A*B interaction only if both main effects have already entered the model. Also, before a main effect is removed from the model, the interaction term that contains the main effect must be removed beforehand. You may also allow more than one effect to enter the model in a given step, as long as hierarchy is maintained.

Sequential

Select Yes for the Sequential property to force effects to enter the model sequentially. The Sequential option is especially useful when you are fitting a polynomial model. In this case, when using forward selection with Sequential, terms of increasing order (linear, quadratic, cubic, and so on listed in increasing order in the Model Ordering window) are added sequentially to the model until the selection process terminates based on the Entry Sig. Level. This prevents the node from adding a higher order term to the model without first including the lower order term. The Sequential option works independently of the Hierarchy Effects and Moving Effect Rule properties. By itself, Sequential does not require hierarchy.

Hierarchy Effects

The Hierarchy Effects determines if only class variables or both class and interval variables are subject to hierarchy. By default, the Hierarchy Effect property is set to Class (only class variables are considered in the hierarchies). To subject both interval and class variables to hierarchy, select the All variable type.

Moving Effect Rule

This option determines whether hierarchy is maintained and whether single or multiple effects enter or leave the model in one step. For the discussion that follows, suppose you specify the main effects A and B and the interaction of A*B as candidate effects.

You can choose from the following options for the Moving Effect Rule:

- **None** — (default) Hierarchy is not maintained. Any single effect can enter or leave the model at any given step of the selection process. The difference between None and Single is that hierarchy must be maintained with Single.

- **Single** — Only one term can enter or leave the model at one time subject to hierarchy. In the first step of the selection process, either A or B can enter the model. In the second step, the other main effect can enter the model. The interaction effect can only enter the model when both main effects have already been entered. Also, before A or B can be removed from the model, the A*B interaction must first be removed.

- **Multiple** — More than one effect can enter or leave the model at one time subject to hierarchy. For example, in the first step of the selection process, a single main effect can enter the model, or both main effects along with the interaction can enter the model. In the first step of a backward selection, the interaction alone or both the interaction along with both main effects can be deleted.

How the Regression Node Treats the Values of Categorical Variables

Categorical values (class variable level values) in SAS Enterprise Miner are normalized as follows:

- left justified
- uppercased
- truncated to a length of 32 (after left-justification)

This implies, for example, values such as "YES", "yes", "Yes", and " yes" are all considered the same. Likewise, "purchasing a new big bright red ball" and "purchasing a new big bright red cap" are also considered identical, because they are not unique within the first 32 characters.

Normalized values are stored in the DMDB catalog. The modeling tools work with normalized class values for both modeling as well as scoring.

New variable names constructed from category values will be based on normalized values.

The normalization process is aimed to provide consistent behavior in handling category values throughout the product.

Regression Node Results

Regression Node Results Window

After a successful node run, you can open the Results window of the Regression node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu in the Results — Regression window to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Regression node properties configuration when the node was last run.
  - **Run Status** — displays the status of the Regression node run. Information about the run start time, run duration, and completion status are displayed in this window.
  - **Variables** — displays a table of the variables in the training data set.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays notes of interest, such as data or configuration information.
- **SAS Results**
  - **Log** — the SAS log of the Regression run.
  - **Output** — the SAS output of the Regression run.
  - **Flow Code** — the SAS code used to produce the output that the Regression node passes on to the next node in the process flow diagram.

- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled on page 55 .
- **Assessment**
  - **Fit Statistics** — a table of the fit statistics from the model.
  - **Classification Chart** — The Classification chart displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represent the percentage of total observations.
  - **Decision Chart** — displays a bar chart of the proportion of correctly classified observations and the proportion of misclassified observations in the training and validation data sets.
  - **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles and observations in a decile are used to calculate the statistics that are plotted in deciles charts. The Score Rankings Overlay plot displays both train and validate statistics on the same axis.

    By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations.

    The vertical axis displays the following values:

    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response
    - Total Profit
    - Expected Profit
  - **Score Rankings Matrix** — The score rankings matrix plot overlays the selected statistics for standard, baseline and best models in a lattice that is defined by the training and validation data sets. Plots can also be created for report variables. The chart choices are
    - Cumulative Lift
    - Lift
    - Gain

        - % Response
        - Cumulative % Response
        - % Captured Response
        - Cumulative % Captured Response
    - **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used. For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. The Score Distribution chart of a useful model shows a higher percentage of events for higher model score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are
        - Percentage of Events — for categorical targets.
        - Number of Events — for categorical targets.
        - Cumulative Percentage of Events — for categorical targets.
        - Expected Profit — for categorical targets.
        - Report Variables — for categorical targets.
        - Mean for Predicted — for interval targets.
        - Max. for Predicted — for interval targets.
        - Min. for Predicted. — for interval targets.
    - **Residual Statistics** — displays a box-and-whisker plot for the residual variable VALUE measurements when the target is interval.
- **Model** — graphs and tables with information about the variables in the model. The available graphs and tables are
    - **Effects Plot** — displays a bar graph of the absolute values of the coefficients in the final model. The bars are color coded to indicate the algebraic signs of the coefficients.
    - **Estimates Selection Plot** — displays a graph of various statistics at each step in the model selection process. The horizontal axis displays the Model Selection Step Number and the variable names populate the vertical axis. For each variable, at each step number, color coded boxes appear. The intensity of the color in each box reflects the numeric value of the statistic. If you click on a box, the numeric value of the statistic is displayed. The available statistics are
        - Absolute Coefficient
        - Coefficient
        - T-value
        - Absolute T-value
    - **Iteration Plot** — displays graphs of various statistics at each step in the model selection process. The horizontal axis displays the Model Selection Step Number and the available statistics for the vertical axis are
        - Average Squared Error

  - Error Function
  - Sum of Squared Errors
  - Average Error Function
  - Maximum Absolute Error
  - Mean Squared Error
  - Root Mean Squared Error
  - Final Prediction Error

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Output from Linear Regression Runs

The standard output for linear regression lists the following information about the model:

- R-square — the coefficient of determination, which represents the proportion of the sum of squares of the target attributable to the information obtained from the independent input variables. For example, if the model R-square statistic is 0.40, then 40% of the variation in the target is explained by its relationship with the input variables.
- Adjusted R-square — the adjusted coefficient of determination, which is often used to compare two or more models fit to the same data. The Adjusted R-square is defined as 1 minus the mean square error divided by the mean square total. This statistic does not necessarily increase as more parameters are added to the model, has a maximum value of 1, and can be negative. Large differences between the R-square and the adjusted R-square values for a given model can indicate that you have used too many explanatory inputs in the model.
- AIC (Akaike's Information Criterion) — n*ln(SSE/n) +p*ln(n), where n is the number of cases, SSE is error sum of squares, and p is the number of model parameters. The AIC criterion penalizes for adding parameters to the model. Small values of AIC are preferred.
- SBC (Schwarz's Bayesian Criterion) — p + (s^2 — σ^2)*(n — p)/σ^2. The SBC criterion penalizes for adding parameters to the model. Small values of SBC are preferred.

For each parameter estimate, the standard output lists the following information:

- Degrees of Freedom
- Value of the Estimate
- Standard Error
- Type II Sum of Squares
- F-test
- p-value

Output from Logistic Regression Runs

The standard output for logistic regression provides the following information about the model:

- Response Profile — The Response Profile table orders the levels of the response variable by associating an Ordered Value with each level. For binary response variables, observations with Ordered Value 1 are considered event observations. Those with Ordered Value 2 are considered nonevent observations. The Response Profile table also displays the count of each level in the training data set.
- Input Class Level Information — For each class input variable, the Input Class Level Information table lists the values of the design matrix.
- Model Fitting Information and Testing Global Null Hypothesis BETA=0 — This table provides two criteria for comparing models and one test of the null hypothesis that all regression coefficients are zero. All statistics are based on the likelihood for fitting a model with intercepts only or for fitting a model with intercepts and input variables.
- Type III Analysis of Effects — This table provides overall tests for the model effects. Since effects involving class inputs consist of multiple model parameters, tests of these effects have multiple degrees of freedom. Tests of the individual constituent parameters appear in the following table.
- Analysis of Maximum Likelihood Estimates — This table displays the estimates of the individual model parameters and tests of their significance.
- Odds Ratio Estimates — This table displays odds ratio estimates for each main effect in the model that is not involved in an interaction. For an interval input, the odds ratio estimates the change in odds for a unit increase in the input. For a class input, a set of odds ratios is provided that compares each level of the input variable with the last level.

Regression Node Output Data Sets

The Regression node must run before you can view the Regression node output data sets. After a successful run, ensure that the Regression node is selected in the Diagram Workspace, then select the ... button to the right of the Exported Data property. This opens the Exported Data — Regression window, which lists the data sets that the Regression node outputs.

Two types of data sets are output from the Regression node:

- Scored Data Sets — are the scored training, validation, test, and score data set that contains original inputs and scores (prediction, residuals, classification results, and so on).

  Here is a list of the name prefixes of the scores in the scored data sets:

  - BL_ or BP_ — best possible profit or loss for any of the decisions
  - CL_ or CP_ — profit or loss that is computed from the target value
  - D_ — level of the decision that is chosen by the model
  - E_ — error function
  - EL_ or EP_ — expected profit or loss that is chosen by the model
  - F_ — normalized category that the case comes from
  - I_ — normalized category that the case is classified into

  - IC_ — investment cost
  - P_ — posterior probabilities for categorical targets or predicted values for interval targets
  - R_ — residuals
  - ROI_ — return on investment
  - U_ — un-normalized category that the case is classified into.

  When the model selection methods of forward, backward, or stepwise are applied, the Regression node automatically changes the model roles from input to rejected for variables that are not included in the regression model.

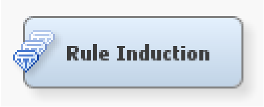
- Parameter Estimates Data Set — contains information about fit statistics (error function, misclassification rate, and so on).

  The following lists the variables that are included in the parameter estimates data set:

  - _AIC_ — Akaike's Information Criterion
  - _APROF_ or _ALOSS_ — average profit or loss for a decision processing
  - _ASE_ — average squared error for the training data set
  - _AVERR_ — average error function
  - _DFE_ — error degrees of freedom
  - _DFM_ — model degrees of freedom
  - _DFT_ — total degrees of freedom
  - _DIV_ — divisor for ASE
  - _ERR_ — error function
  - _FPE_ — final prediction error
  - _LINK_ — link function
  - _MAX_ — maximum absolute error
  - _MISC_ — misclassification rate of the training data set
  - _MSE_ — mean squared error
  - _NOBS_ — sum of frequencies for the training data set
  - _NW_ — number of estimates weights
  - _PROF_ or _LOSS_ — total profit for a decision processing
  - _RASE_ — root average squared error
  - _RFPE_ — root final prediction error
  - _RMSE_ — root mean squared error
  - _SBC_ — Schwarz's Bayesian Criterion
  - _SSE_ — sum of squared error
  - _SUMW_ — sum of case weights times frequency

Chapter 62
Rule Induction Node

Rule Induction Node



Overview of the Rule Induction Node

Use the Rule Induction node to improve the classification of rare events. When you classify rare events, the Rule Induction node can do the following:

- (optional) creates an initial ripping (Rule Induction Program) model. A tree model is run to find the largest node that meets the purity threshold that you set. The largest node that meets the purity threshold criterion is then removed from the data.
- creates a binary model for each level of a target variable. If you set the Binary Order property to Descending, the node starts with the most common event and progresses to the most rare event. Rows in which the rare event is correctly classified are removed from the training data set. You can also start from the most rare event and progress to the most common event by setting the Binary Order property to Ascending.
- creates a cleanup model of all of the unclassified rows. The cleanup model models all levels of the target.

See the Predictive Modeling section for information that applies to all of the predictive modeling nodes.

Data Set Requirements

The training data set that is submitted to the Rule Induction node must contain a binary, ordinal, or nominal level target variable.

Rule Induction Node Properties

Rule Induction Node General Properties

The following general properties are associated with the Rule Induction Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Rule Induction node that is added to a diagram has a Node ID of Rule. The second Rule Induction node that is added to a diagram has a Node ID of Rule2, and so on.
- **Imported Data** — the Imported Data property provides access to the Imported Data — Rule Induction window. The Imported Data — Rule Induction window contains a list of the ports that provide data sources to the Rule Induction node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — the Exported Data property provides access to the Exported Data — Rule Induction window. The Exported Data — Rule Induction window contains a list of the output data ports that the Rule Induction node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Rule Induction Node Train Properties

- **Variables** — Use the Variables property of the Rule Induction node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, as well as select which variables you want included in reports.
- **Initial Ripping** — Use the Initial Ripping property of the Rule Induction node to indicate whether you want to run the initial ripping algorithm. Set the Initial Ripping property to **No** if you want to suppress the initial ripping algorithm. The default

setting of **Yes** creates trees that find the largest pure node that is to be removed from the data.

- **Purity Threshold** — Use the Purity Threshold property of the Rule Induction node to specify the minimum training leaf purity percentage that is required for a leaf to be ripped. Observations in leaves that meet or exceed the threshold are removed from the training data. The default setting for the Purity Threshold property is 100. Acceptable values are integers between 50 and 100.
- **Max. Number of Rips** — Use the Max. Number of Rips property of the Rule Induction node to specify the maximum number of times that the ripping routine is run. The default setting for the Max. Number of Rips property is 16. Acceptable values are integers between 1 and 100.
- **Binary Models** — Use the Binary Models property of the Rule Induction node to indicate the algorithm that you want to use to create the binary models. A binary model is created for each level of the target variable. Rows in which the event level is correctly classified are removed from the training set. You can choose from **Tree**, **Neural**, and **Regression**. The default setting for the Binary Models property is **Tree**.
- **Binary Order** — specifies the order in which binary models are trained. Binary models are trained in the order of the training data frequencies of the target event. Select **Descending** to start modeling the most common event.
- **Cleanup Model** — indicates the model that is applied to all of the remaining observations that have not been correctly classified.

Rule Induction Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Rule Induction Node Results

Rule Induction Results Menu

You can open the Results window of the Rule Induction node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

- **Properties**
  - **Settings** — displays a window with a read-only table of the Rule Induction node properties configuration when the node was last run. Use the **Show Advanced**

  Properties check box at the bottom of the window to see all of the available properties.

  - **Run Status** — indicates the status of the **Rule Induction** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.

- **SAS Results**
  - **Log** — the SAS log of the Rule Induction run.
  - **Output** — the SAS output of the Rule Induction run.
  - **Flow Code** — the SAS code that was used to produce the output that the **Rule Induction** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **Rule Induction** node does not generate PMML code.
- **Assessment**
  - “Rule Induction Node Fit Statistics” on page 944 — a table of the fit statistics from the model.
  - “Classification Chart” on page 945 — a bar chart that shows the correct classification of the values of the class target variable.
  - Score Rankings Overlay: <TARGET> on page 945 — opens the Score Rankings Overlay chart.
  - Score Rankings Matrix: <TARGET> on page 946 — opens the Score Rankings Matrix chart.
  - Score Distribution: <TARGET> on page 947 — opens the Score Distribution chart.
- **Table** — opens a table of the data that is associated with the graph that you have open.
- **Graph Wizard** — opens the graph wizard, using the data in the table that you open.

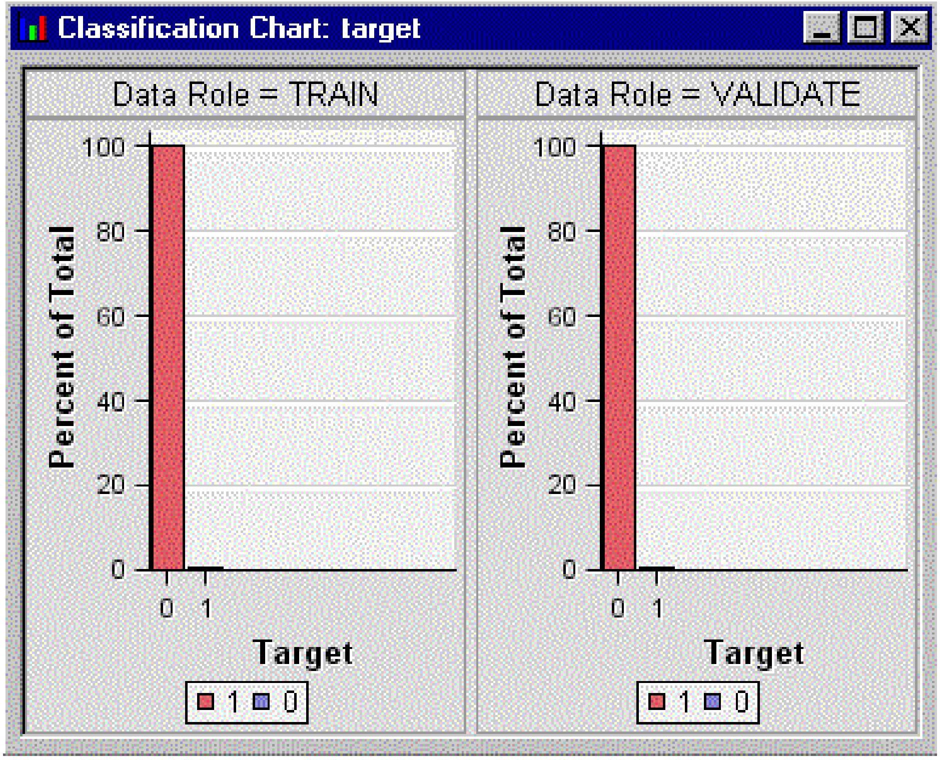
Rule Induction Node Fit Statistics

- The Rule Induction Fit Statistics table displays the statistics below.
  - _APROFIT_ — Average Profit of the Target
  - _AVERR_ — Average Error Function
  - _DFE_ — Degrees of Freedom for Error
  - _DISF_ — Frequency of Classified Cases
  - _DIV_ — Divisor for _ASE_
  - _ERR_ — Error Function
  - _MISC_ — Misclassification Rate
  - _NOBS_ — Sum of Frequencies

- _PROFIT_ — Total Profit
- _WRONG_ — Number of Wrong Classifications

Classification Chart

- The Classification chart displays a stacked bar chart of the classification results for a categorical target variable. The following display shows an example of the Classification Table chart:



The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into. The height of the stacked bars represents the percentage of total observations.

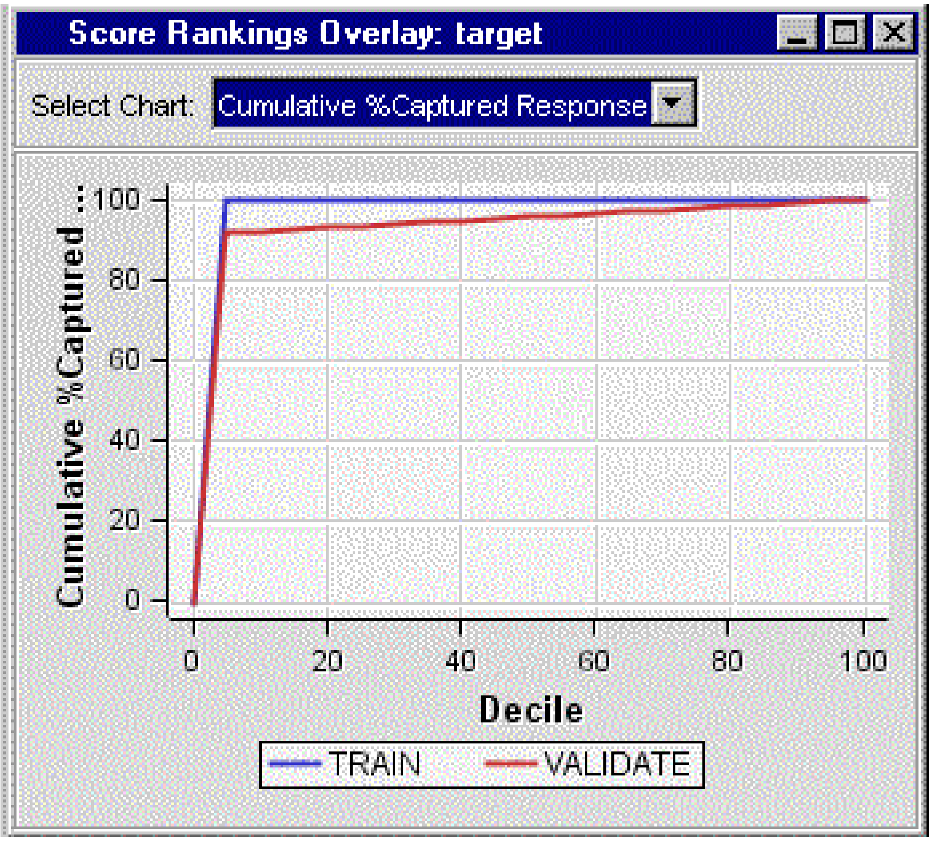
Score Rankings Overlay Chart

- In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles based on the Decile Bin property and observations in a decile are used to calculate the statistics that are plotted in deciles charts.

  By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations. The vertical axis displays the following values, and their mean, minimum, and maximum (if any).

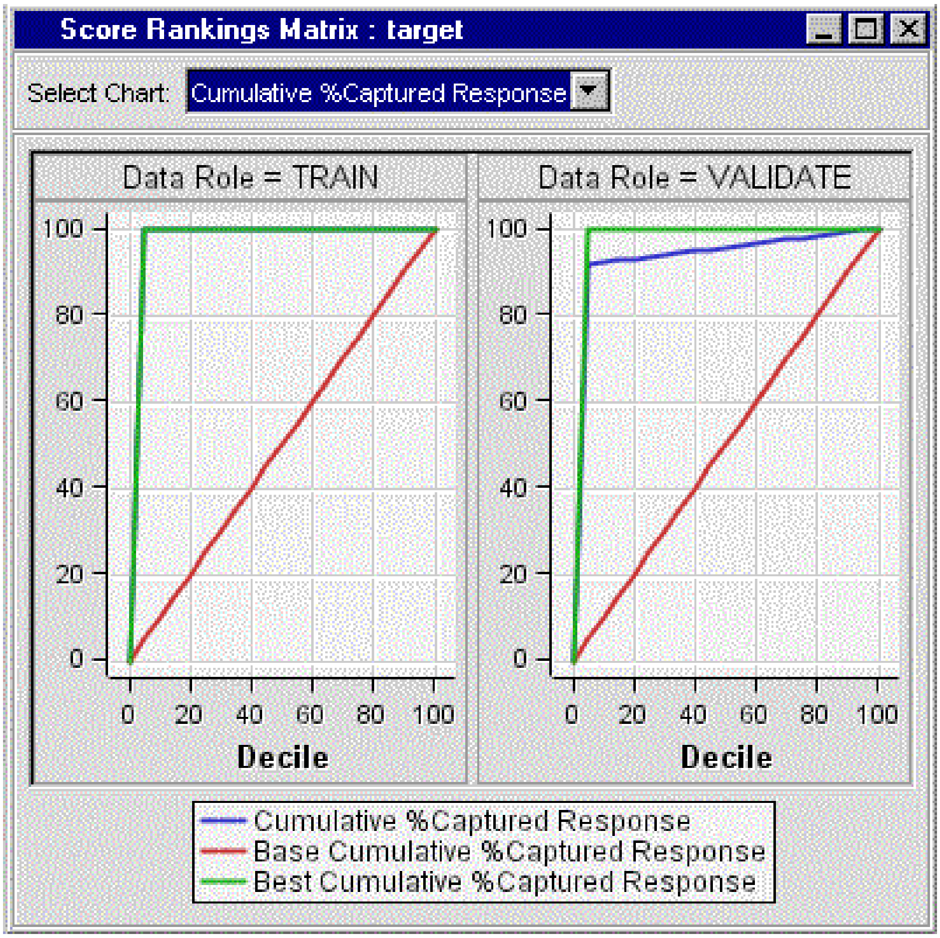
  - posterior probability of target event
  - number of events
  - cumulative and noncumulative lift values

- cumulative and noncumulative % response
- cumulative and noncumulative % captured response
- gain
- actual profit or loss
- expected profit or loss



Score Rankings Matrix Chart

- The score ranking matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the training and validation data sets. The example below plots model cumulative lift, base model cumulative lift, and the best cumulative lift across deciles. Plots can also be created for report variables.



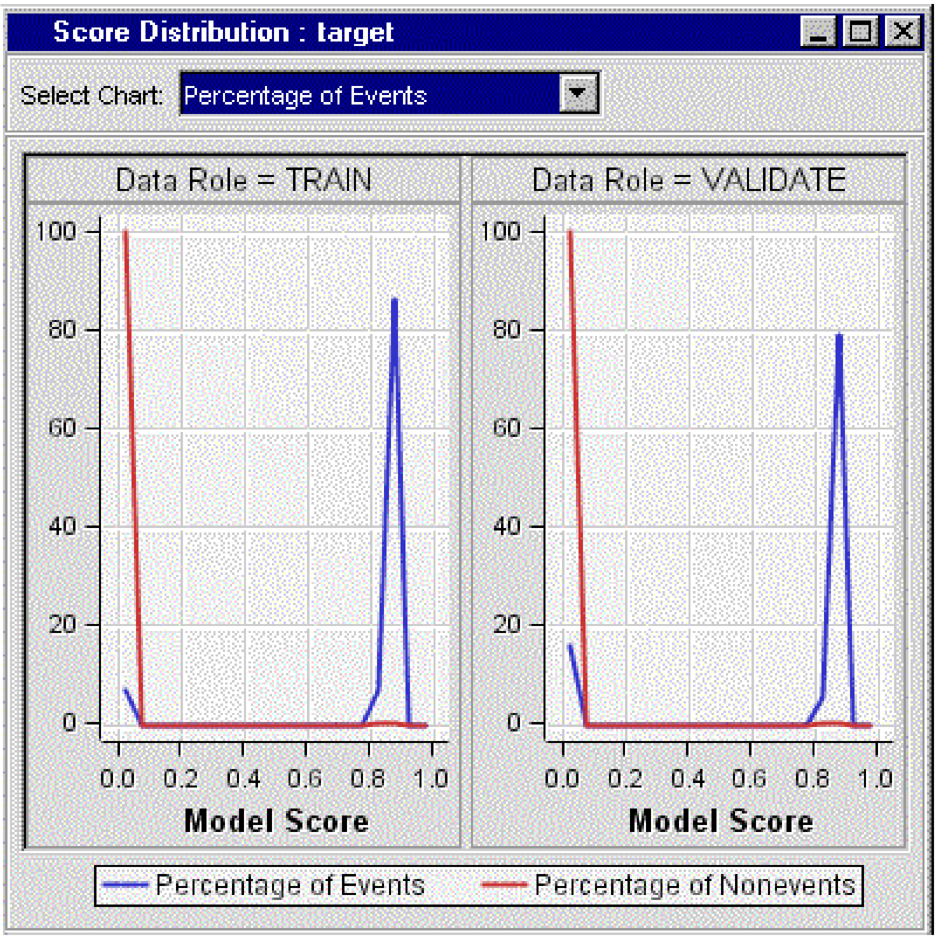
- Chart choices other than Cumulative Lift are available for the Score Rankings Matrix Chart. The chart choices are as follows:
  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response

Score Distribution Chart

- The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the value of the ScoreDistBin property.

  For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

  The Score Distribution chart of a useful model shows a higher percentage of events for higher model scores and a higher percentage of nonevents for lower model scores.



- Use the drop-down list to select the score distribution statistic that you would like to chart. The default chart choice is Percentage of Events. Multiple chart choices are available for the Score Distribution Chart. The chart choices are as follows:
  - Percentage of Events — for categorical targets
  - Number of Events — for categorical targets
  - Cumulative Percentage of Events — for categorical targets
  - Mean for Predicted — for interval targets
  - Max. for Predicted — for interval targets
  - Min. for Predicted — for interval targets

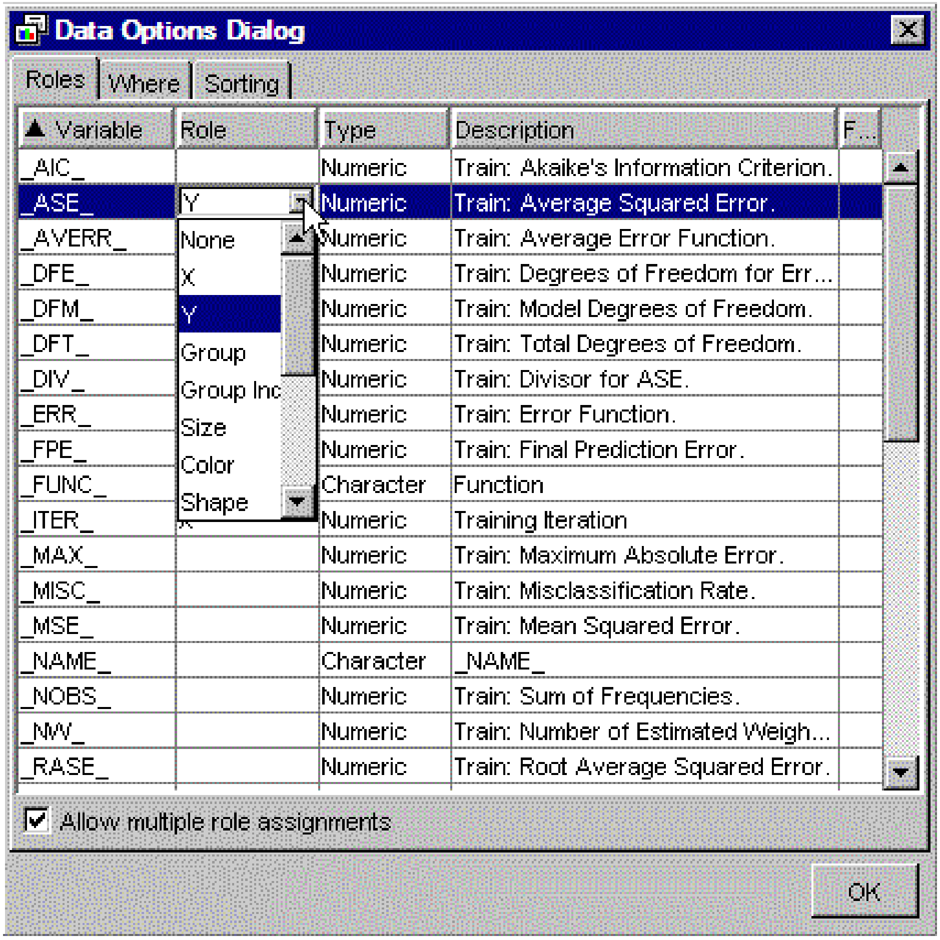
Modifying Results Graphs

You can use the Data Options Dialog window to modify existing results plots. For example, you can add or change the response variables in the graphs. To open the Data Options Dialog window, right-click inside a Results graph and select **Data Options** from the pop-up menu.

For more detailed information about using the Data Options Dialog window, see the section in the SAS Enterprise Miner User Interface Help on the Data Options Dialog window.

You can also modify attributes of an existing graph using the Graph Properties window. You can open the Data Options Dialog window, which contains three tabs for configuring data plots.

- The **Roles** tab of the Data Options Dialog window contains a table that you can use to configure response variables for a graph or plot. The Variables column lists a variety of SAS variables that you can choose as your response variables. Columns for Type, Description, and Format provide additional information about the available SAS variables.

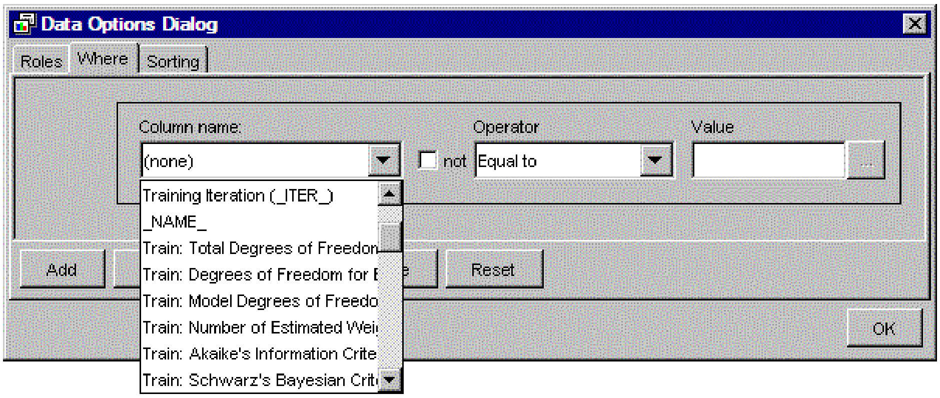


▲ Variable	Role	Type	Description	F...
AIC		Numeric	Train: Akaike's Information Criterion.	
ASE	Y	Numeric	Train: Average Squared Error.	
AVERR		Numeric	Train: Average Error Function.	
DFE		Numeric	Train: Degrees of Freedom for Err...	
DFM		Numeric	Train: Model Degrees of Freedom.	
DFT		Numeric	Train: Total Degrees of Freedom.	
DIV		Numeric	Train: Divisor for ASE.	
ERR		Numeric	Train: Error Function.	
FPE		Numeric	Train: Final Prediction Error.	
FUNC		Character	Function	
ITER		Numeric	Training Iteration	
MAX		Numeric	Train: Maximum Absolute Error.	
MISC		Numeric	Train: Misclassification Rate.	
MSE		Numeric	Train: Mean Squared Error.	
NAME		Character	_NAME_	
NOBS		Numeric	Train: Sum of Frequencies.	
_NW_		Numeric	Train: Number of Estimated Weigh...	
RASE		Numeric	Train: Root Average Squared Error.	

☑ Allow multiple role assignments

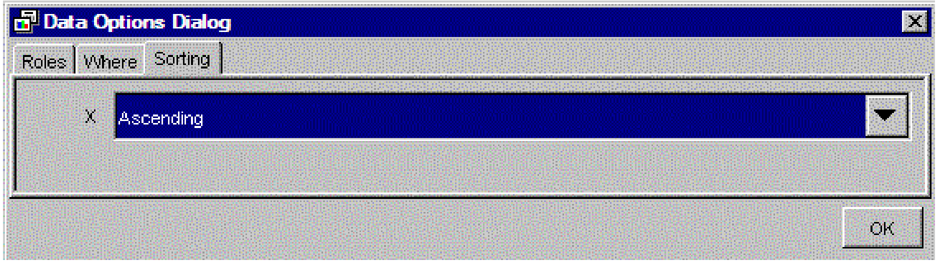
Click a cell in the Role column to open a list that contains a variety of graphical role options. Most users find the greatest utility in changing the **Y** response variable to different fit statistics. If the **Allow multiple role assignments** check box is selected, more than one Y response variable can be selected. Only the Role column can be changed or modified. You can click on column headings to toggle between ascending and descending sorts. In the example above, the Variable column is sorted in ascending order. Click **OK** to see your changes updated in the graph plot.

- The **Where** tab of the Data Options Dialog window contains an expression builder that you can use to build a WHERE-clause or a Boolean expression to subset the data that you want to plot.


Data Options Dialog
Roles
Where
Sorting
Column name:
(none)
not
Operator
Equal to
Value
...
Training Iteration (_ITER_)
NAME
Train: Total Degrees of Freedom
Train: Degrees of Freedom for E
Train: Model Degrees of Freedo
Train: Number of Estimated Wei
Train: Akaike's Information Crite
Train: Schwarz's Bayesian Crit
Add
Reset
OK

The Column name drop-down list contains a list of SAS output variables and in particular, SAS fit statistic variables, which are sorted by Train and Validation data roles. Use the Column name drop-down list to select the subsetting variable that you want to use. Use the Operator and the optional 'not' check box to specify your Boolean operator. You can either enter a Value, or click the **Value** browse button [...] to open a window that you can use to browse lists of unique values or variable names to complete the **Value** field. After you finish building your Boolean subsetting expression, click **OK** to apply your changes to the plotted data, or select another **Data Options Dialog** tab.

- The **Sorting** tab of the Data Options Dialog window can be used to sort the X-axis data points if desired.



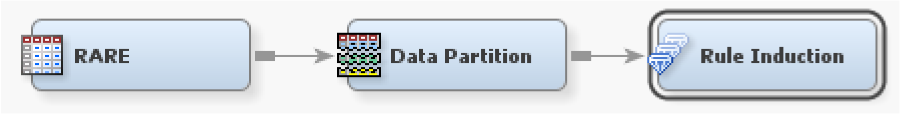
The drop-down list choices are **Ascending**, **Descending**, and **Data**. You can use the ascending and descending ordering controls, for example to look at descending frequency counts without having to sort the data table on the server. The Data ordering applies to ordering bars in bar charts where nominal or ordinal groupings are used.

Example: Using Initial Ripping with the Rule Induction Node

Overview

The following example demonstrates how the **Rule Induction** node uses initial ripping to classify rare events. In this example, a data set is generated with SAS code, and then analyzed using the default **Rule Induction** node properties.

Follow these steps to create the process flow diagram:



Create a Rare Event Data Set

1. From the SAS Enterprise Miner main menu, select **View** ⇨ **Program Editor**.
2. Type (or copy and paste) the following code in the Program Editor window.

```
data emds.rare ;
       drop i ;
       do i=1 to 200000 ;
          x1= ranuni(1) ;
          x2= ranuni(2) ;
          if x1*x2  < 0.05*0.001 then
            target= 1 ;
          else
            target= 0 ;
          output ;
       end ;
       run ;

       PROC FREQ data=emds.rare ;
       table target;
       run ;
```

3. Submit the SAS code. The code creates a data set that contains a target with a rare event. The example code uses random functions to generate the rare event data set. So it is not unusual if your resulting output and statistics for this example vary from the results that are shown here.
4. Select the **Output** tab at the bottom of the Program Editor window.
5. The following text is displayed in the Output window:

```
The FREQ Procedure

                                   Cumulative    Cumulative
target    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
     0      199885       99.94        199885        99.94
     1         115        0.06        200000       100.00
```

Define the Rare Event Data Source

1. From the SAS Enterprise Miner main menu, select **File** ⇨ **New** ⇨ **Data Source**, or right-click the Data Sources icon in the Project Navigator and select **Create Data Source** from the pop-up menu.
2. Select the **SAS Table** as the metadata source and click **Next**.
3. Type in the table **`EMDS.RARE`** as the data source.
4. View the table properties window and click **Next**.
5. In the Metadata Configuration window, select the **Basic** advisor. Click **Next**.
6. In the Column Metadata window, set the variable role of TARGET to **Target**. Set the Level of TARGET to **Binary**. Ensure that the variables X1 and X2 have a role of **Input**. Click **Next** until you reach the Data Source Attributes page.
7. In the Data Source Attributes window, select **Raw** as the data set Role. Click **Finish**.

Add Nodes to the Diagram Workspace

1. In the Project Panel, right-click **Diagrams** and select **Create Diagram**. In the Create New Diagram window, enter **`Rule Induction Example`**.

2. Drag the **RARE** data source onto the Diagram Workspace.

3. From the **Sample** tab, drag a **Data Partition** node onto the diagram, and connect the **RARE** data source node to the **Data Partition** node.

4. From the **Model** tab, drag a **Rule Induction** node onto the diagram, and connect the **Data Partition** node to the **Rule Induction** node.

Set the Partition Node Properties

1. Click the **Data Partition** node in your diagram to select it.

2. In the **Data Partition** node properties panel, modify the **Data Set Percentages** properties as follows:

   - Set the **Training** property percentage to `50`.
   - Set the **Validation** property percentage to `50`.
   - Set the **Test** property percentage to `0`.

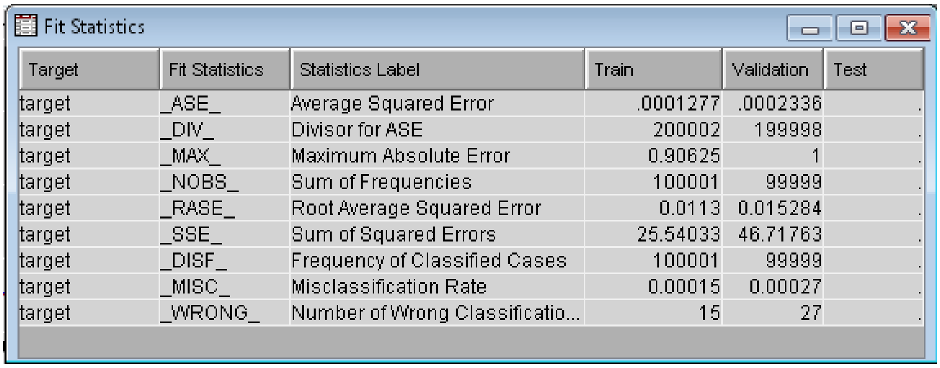
Run the Rule Induction Node and View Results

1. Right-click the **Rule Induction** node and click **Run**. In the Confirmation window, select **Yes**.

2. When the run is complete, click **Results** in the Run Status window.

3. In the Results window, select **View** ⇨ **Assessment** ⇨ **Fit Statistics**.

4. Examine the Fit Statistics table:



Target	Fit Statistics	Statistics Label	Train	Validation	Test
target	_ASE_	Average Squared Error	.0001277	.0002336	.
target	_DIV_	Divisor for ASE	200002	199998	.
target	_MAX_	Maximum Absolute Error	0.90625	1	.
target	_NOBS_	Sum of Frequencies	100001	99999	.
target	_RASE_	Root Average Squared Error	0.0113	0.015284	.
target	_SSE_	Sum of Squared Errors	25.54033	46.71763	.
target	_DISF_	Frequency of Classified Cases	100001	99999	.
target	_MISC_	Misclassification Rate	0.00015	0.00027	.
target	_WRONG_	Number of Wrong Classificatio...	15	27	.

Notice that the misclassification rates for the training and validation data sets are 0.00015 and 0.00027, respectively.

View the Output Window

1. Expand the Output window. In the window, you see that a total of five RIPs were performed. Here is the result of the first RIP:

```
RIP1 Leaf Table: Threshold= 100
Leaf 17 was ripped from the model.


                     Predicted:     Predicted:
Node          N       target=0       target=1


 17      99362        1.0000         0.0000
 15        244        1.0000         0.0000
 16        156        0.9679         0.0321
 13         57        1.0000         0.0000
  5         54        0.9630         0.0370
  4       41      0.4878        0.5122
  8       29      0.4828        0.5172
 12       29      0.7586        0.2414
 14       29      0.7241        0.2759
```

In this RIP, node 17 was removed from the model because it was the largest pure node.

2. After the 99362 observations in node 7 are removed from the model, there are 639 observations used in training for RIP2. Scroll down the Output window to view the results of RIP2.

```
RIP2 Leaf Table: Threshold= 100
Leaf 7 was ripped from the model.

                Predicted:    Predicted:
Node     N      target=0      target=1

  7     304      1.0000        0.0000
  9     196      1.0000        0.0000
  8      69      0.6812        0.3188
  2      41      0.4878        0.5122
  4      29      0.4828        0.5172
```

In this RIP, node 7 was removed from the model. In the fifth RIP, no node is removed from the model.

3. Scroll down the Output window. A frequency table of the values of the remaining target variables is displayed.

```
Target=target: Frequencies

          Train     Train     Valid     Valid
target    Count     Percent   Count     Percent

  0         60      50.8475     73      57.9365
  1         58      49.1525     53      42.0635
```

4. Scroll down the Output window. The classification for each value of the target is displayed.

   For each level of the target, the process for the binary models is as follows:

   - A dummy variable is created. A value of 1 means that the current observation has the target value that is being examined at that time. A value of 0 means that the target value is something other than the value that is currently being examined. For example, you have a target with levels a, b, and c. For the first binary model, a new target variable is created that has levels 1 (original target = a) and 0 (original target = b or c).
   - A binary model is now run on this new dummy variable that is coded as the target. If you look at the frequency table that is generated after the model run, a value of 1 in the From and Into columns identifies those observations that were correctly classified. In the example, a value of 1 in the From and Into columns corresponds to an observation that had an original target value=a and was classified as an a.

     In this example, the following table indicates that each observation is classified as either a 0 or a 1. The 54 correctly classified observations at target=0 are then removed from the training data set.

```
Binary Model target = 0
```

```
Classification Table

               Train     Train     Valid     Valid
From    Into   Count     Percent   Count     Percent

 0       0      49       41.5254     44      34.9206
 0       1       9        7.6271      9       7.1429
 1       0       6        5.0847     14      11.1111
 1       1      54       45.7627     59      46.8254
```

- When the observations are removed from this first binary model, the remaining observations are then dummied for the next value of the original target variable, and the process continues until you have gone through all values of the original target variable.

  In this example, the following table indicates that 58 observations were correctly classified as 1. When those 58 observations are removed from the training data, a unary target of only zeros remains. No cleanup process is needed for unary targets.

  ```
  Binary Model target = 1
  Classification Table

                 Train     Train     Valid     Valid
  From    Into   Count     Percent   Count     Percent

   0       1       6        9.375      14      20.8955
   1       1      58       90.625      53      79.1045
  ```

5. Scroll down the Output window and view the **Event Classification Table**:

   ```
   Event Classification Table

   Data Role=TRAIN Target=target

     False       True       False      True
   Negative    Negative    Positive   Positive

       9        99937         6          49


   Data Role=VALIDATE Target=target

     False       True       False      True
   Negative    Negative    Positive   Positive

      13        99928        14          44
   ```

   The Fit Statistics table displays a summary of this information.

Chapter 63
SVM Node (Experimental)

SVM Node (Experimental)

Overview of the SVM Node

A support vector machine (SVM) is a supervised machine-learning method that is used to perform classification and regression analysis. Vapnik (1995) developed the concept of SVM in terms of hard margin, and later he and his colleague proposed the SVM with slack variables, which is a soft margin classifier. The standard SVM model solves binary classification problems that produce non-probability output (only sign +1/-1) by constructing a set of hyperplanes that maximize the margin between two classes. Most problems in a finite dimensional space are not linearly separable. In this case, the original space needs to be mapped into a much higher dimensional space or an infinite dimensional space, which makes the separation easier. SVM uses a kernel function to define the larger dimensional space.

The experimental SAS Enterprise Miner **SVM** node uses PROC SVM and PROC SVMSCORE. The tool is located on the **Model** tab of the SAS Enterprise Miner toolbar. In SAS Enterprise Miner 12.3, the **SVM** node supports only binary classification problems, including polynomial, radial basis function, and sigmoid nonlinear kernels. The **SVM** node does not perform multi-class problems or support vector regression. The frequency variable is ignored.

SVM Node Properties

SVM Node General Properties

The following general properties are associated with the **SVM** node:

- **Node ID** – The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **SVM** node that is added to a diagram will have a Node ID of SVM. The second **SVM** node added to a diagram will have a Node ID of SVM2, and so on.
- **Imported Data** – accesses the Imported Data — SVM window. The Imported Data — SVM window contains a list of the ports that provide data sources to the **SVM** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click as follows:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — SVM window. The Exported Data — SVM window contains a list of the output data ports that the **SVM** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click as follows:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

SVM Node Train Properties

- **Variables** — Select the ... button to open the Variables — SVM table, which enables you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify **Use** and **Report** variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - **Apply** — changes metadata based on the values supplied in the drop-down menus, check box, and selector field.

  - **Reset** — changes metadata back to its state before use of the **Apply** button.
  - **Label** — adds a column for a label for each variable.
  - **Mining** — adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — adds statistics metadata for each variable.
  - **Explore** — opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Estimation Method** — specifies an SVM modeling method.
  - **Full Dense Quadratic Programming (FQP)** — Full dense quadratic programming refers to the method used to store a Hessian matrix, not the techniques used to solve the QP problem. A full dense matrix requires substantial amounts of memory, even for relatively small problems. Therefore, this technique is not recommended for medium- or large-sized data sets. For certain problems that are small enough to support a full dense Hessian, this method might be quicker than other methods.
  - **Decomposed Quadratic Programming (DQP)** —Decomposed quadratic programming provides solutions by decomposing large scale quadratic programming problems into a series of smaller quadratic programming problems. Decomposed quadratic programming divides the input data into two partitions. The data in a small partition is optimized as a sub-problem, and the remaining data in a much larger partition remains constant. As data points in the sub-problem are optimized, they are iteratively replaced with data from the larger partition by evaluating the optimality conditions. The large quadratic programming problem is resolved by successively resolving a series of small quadratic programming problems.
  - **Lagrangian SVM (LSVM)** — The Lagrangian SVM method is suitable for large scale SV classification models with a linear kernel. It is also suitable for medium-sized models with nonlinear kernels. The Lagrangian SVM method follows Mangasarian and Musicant (2000), where the modified quadratic programming problem has no linear equality constraint and no upper bounds. The only constraints are that estimated parameters must be nonnegative. For linear kernels, Lagrangian SVM does not require data (X, y) in the core. Lagrangian SVM is an iterative method that makes multiple passes through the data set.
  - **Least Squares SVM (LSSVM)** — The least squares SVM method was originally developed by Suykens and Vandewalle (1999). Least squares SVM solves linear and nonlinear C classification problems. Two versions of least squares SVM exist for classification. The faster version, which is suitable for smaller n × n kernel matrices, computes and stores the (dense) Cholesky factor. For larger n, the kernel matrix must not be stored in core when the iterative conjugate gradient algorithm is used. The slower version, which is suitable for small and medium sized n, uses a conjugate gradient solution.
- **Tuning Method** — specifies the method used to perform tuning.
  - **None** — Tuning is not performed. The constant value specified as a tuning parameter is used.
  - **Grid Search** — grid search.
  - **Optimal Search** — direct optimal or pattern search.

- **Optimization Options** — enables you to read and modify values for the following:
  - **Maximum QP Size** — specifies the maximum size in observations of the small decomposed quadratic programming partition when the Estimation Method property is set to DQP. The default setting for the Maximum QP Size property is 100.
  - **Maximum Iteration** — specifies the upper bound for the number of iterations in each optimization. The default is 200.
  - **Maximum Function Call** — specifies an upper bound for the number of function calls in each optimization. The default is 500.
  - **Gradient Convergence Criterion** — specifies a relative gradient convergence criterion for the optimization process. The default is 1e-8.
  - **Absolute Gradient Convergence Criterion** — specifies an absolute gradient convergence criterion for the optimization process. The default is 5e-4.
  - **Relative Convergence Criterion** — specifies a relative parameter convergence criterion for the optimization process. The default is 1e-8.
  - **Absolute Convergence Criterion** — specifies an absolute gradient convergence criterion for the optimization process. The default is 5e-4.
  - **Use Conjugate Gradient Method** — specifies whether the iterative conjugate gradient method is used. When there are more than 3000 observations in training data, the Default option uses the iterative conjugate gradient method automatically.
  - **Upper Bound of Conjugate Gradient Iteration** — This option is valid only for the methods LSSVM with any kernel, as well as LSVM method with any nonlinear kernel. Instead of solving the large linear system via Cholesky decomposition, the system is solved by the iterative conjugate gradient method. This option specifies an upper bound for the number of iterations. The default is 400.
  - **Conjugate Gradient Termination Tolerance** — This option is valid only when the iterative conjugate gradient method is being solved. The option specifies a termination tolerance for the iteration. By default, 1e-6 is used.
- **Scale Predictors** — specifies whether the predictor variables are scaled before entering the analysis. The Scale Predictors property defaults to Yes and becomes dimmed and unavailable when the Kernel property is set to Polynomial, because polynomial kernels always use scaled predictors.

SVM Node Train Properties: Regularization Parameter

- **Regularization Parameter** — Use the Regularization Parameter property to choose the method that you want to use to specify a value for the Regularization Parameter.
  - **Constant** — Set the Regularization Parameter property to Constant if you want to use the Constant Value property to manually specify a value for the Regularization Parameter *C*.
  - **Tuning** — Set the Regularization Parameter property to Tuning if you want to use a tuning method to choose the best Regularization Parameter *C* from a range of values that you specify.
- **Constant value** — When the Regularization Parameter property is set to Constant, use the Constant Value property to specify the value of the polynomial order parameter. The value of polynomial order parameter must be a real number greater than zero.

- **Tuning Range** — When the Regularization Parameter property is set to Tuning, select the ... button to the right of the Tuning Range property to open a window where you can specify parameters for tuning your regularization parameter value:
  - **Start Value** — When the Regularization Parameter property is set to Tuning, use the Start Value property to specify the beginning value of the tuning range.
  - **End Value** — When the Regularization Parameter is set to Tuning, use the End Value property to specify the end value of the tuning range.
  - **Increment Value** — When the Regularization Parameter is set to Tuning, use the Increment Value property to specify the increment value for each tuning step.

SVM Node Train Properties: Kernel

- **Kernel** — specifies the desired kernel function. The default kernel function is Linear.
  - **Linear** — K(u, v) = uTv.
  - **Polynomial** — K(u,v) = (uTv + 1)^{p} with polynomial order *p*. The 1 is added in order to avoid zero-value entries in the Hessian matrix for large values of *p* .
  - **Radial Basis Function** — K(u,v) = exp[-p (u - v)2], Gaussian radial basis function kernel. The radial basis function kernel requires you to specify a value for the kernel scale parameter *p*.
  - **Sigmoid** — K(u,v) = tanh(p*(uTv) + q) where *p* is the kernel scale parameter and *q* is the kernel location parameter.
- **Polynomial Kernel Parameter** — When the Kernel function is set to Polynomial, select the ... button to the right of the Polynomial Kernel Parameter property to open a window where you can specify the following parameters for your polynomial kernel function:
  - **Polynomial Order** — Use the Polynomial Order property to choose the method that you want to use to specify the order of the polynomial kernel parameter.
    - **Constant** — Set the Polynomial Order property to Constant if you want to use the Constant Value property to manually specify a value for the polynomial order parameter.
    - **Tuning** — Set the Polynomial Order property to Tuning if you want to use a tuning method to choose the best value of the polynomial order from a range of values that you specify.
  - **Constant Value** — When the Polynomial Order property is set to Constant, use the Constant Value property to specify the value of the polynomial order parameter. The value of polynomial order parameter must be an integer between 1 and 10.
  - **Tuning Start Value** — When the Polynomial Order property is set to Tuning, use the Tuning Start Value property to specify the beginning value of the tuning range.
  - **Tuning End Value** — When the Polynomial Order property is set to Tuning, use the Tuning End Value property to specify the end value of the tuning range.
  - **Tuning Increment Value** — When the Polynomial Order property is set to Tuning, use the Tuning Increment Value property to specify the increment value for each step in kernel parameter tuning.

- **RBF Kernel Parameter** — When the Kernel function is set to Radial Basis Function, select the ... button to the right of the RBF Kernel Parameter property to open a window where you can specify the following parameters for your RBF kernel function:
  - **Scale Parameter** — Use the Scale Parameter property to choose the method that you want to use to specify the value for the kernel scaling parameter *p*.
    - **Constant** — Set the Scale Parameter property to Constant if you want to use the Constant Value property to manually specify a value for the kernel scaling parameter *p*.
    - **Tuning** — Set the Scale Parameter property to Tuning if you want to use a tuning algorithm to select the best scaling parameter *p* from a range of values that you specify.
  - **Constant Value** — When the Scale Parameter property is set to Constant, use the Constant Value property to specify the value of the radial basis function scaling parameter *p*. The value for the Constant Value property must be a real number greater than zero.
  - **Tuning Start Value** — When the Scale Parameter property is set to Tuning, use the Tuning Start Value property to specify the beginning value of the tuning range.
  - **Tuning End Value** — When the Scale Parameter property is set to Tuning, use the Tuning End Value property to specify the end value of the tuning range.
  - **Tuning Increment Value** — When the Scale Parameter property is set to Tuning, use the Tuning Increment Value property to specify the increment value for each step in parameter tuning.
- **Sigmoid Kernel Parameter** — When the Kernel property is set to Sigmoid, select the ... button to the right of the Sigmoid Kernel Parameter property to open a window where you can specify the following parameters for your sigmoid kernel function:
  - **Sigmoid Kernel Scale Parameters**
    - **Kernel Scale Parameter** — Use the Kernel Scale Parameter property to choose the method that you want to use to specify the value for the sigmoid kernel scaling parameter *p*. The Kernel Scale Parameter property has two settings:

      Constant — Set the Kernel Scale Parameter property to Constant if you want to use the Constant Value property to manually specify a value for the hyperplane scaling parameter *p*.

      Tuning — Set the Kernel Scale Parameter property to Tuning if you want to use an iterative tuning algorithm to select the best scaling parameter *p* from a range of values that you specify.
    - **Constant Value** — When the Kernel Scale Parameter property is set to Constant, use the Constant Value property to specify the value of the scaling parameter *p*. The value for the Constant Value property must be a real number greater than zero.
    - **Tuning Start Value** — When the Kernel Scale Parameter property is set to Tuning, use the Tuning Start Value property to specify the beginning value of the tuning range.

    - **Tuning End Value** — When the Kernel Scale Parameter property is set to Tuning, use the Tuning End Value property to specify the end value of the tuning range.
    - **Tuning Increment Value** — When the Kernel Scale Parameter property is set to Tuning, use the Tuning Increment Value property to specify the increment value for each step in parameter tuning.
  - **Sigmoid Kernel Location Parameters**
    - **Kernel Location Parameter** — Use the Kernel Location Parameter property to choose the method that you want to use to specify the value for the hyperplane location parameter *q*.

      Constant — Set the Kernel Location Parameter property to Constant if you want to use the Constant Value property to manually specify a value for the kernel location parameter *q*.

      Tuning — Set the Kernel Location Parameter property to Tuning if you want to use an iterative algorithm to select the kernel location parameter q from a range of values that you specify.
    - **Constant Value** — When the Kernel Location Parameter property is set to Constant, use the Constant Value property to specify the value of the location parameter *q*. The value for the Constant Value property must be a real number greater than zero.
    - **Tuning Start Value** — When the Kernel Location Parameter property is set to Tuning, use the Tuning Start Value property to specify the beginning value of the tuning range.
    - **Tuning End Value** — When the Kernel Location Parameter property is set to Tuning, use the Tuning End Value property to specify the end value of the tuning range.
    - **Tuning Increment Value** — When the Kernel Location Parameter property is set to Tuning, use the Tuning Increment Value property to specify the increment value for each step in parameter tuning.

SVM Node Train Properties: Cross Validation

- **Cross Validation** — specifies whether cross validation is used.
- **Method** — specifies the cross validation method when the Cross Validation property is set to Yes.
  - **Random** — performs random cross validation. The number of cross validation runs is equal to the value specified in the **Fold** property. Let N be the number of observations in the input data set and F be the value specified in the **Fold** property. Each cross validation run uses N/F observations that are randomly selected from the input data set.
  - **Testset** — permits you to use the test data set as the validation data set.
- **Fold** — specifies the number of training runs and the number of left out data sets. Typical values are fold=4 or fold=10. For fold=1, leave-one-out estimation is used. The default value is 10.

SVM Node Train Properties: Sampling

- **Apply Sampling** — specifies whether sampling is used for SVM training. Stratified sampling is used for class targets.

- **Sample Size** — specifies the sampling size for SVM training. The default is sample size is 1000 observations.

SVM Node Train Properties: Print Options

- **Print Option** — use the Print Option property to configure the granularity of the printed output from the **SVM** node.
  - **Default** — suppresses printing of some SVM model details.
  - **All** — prints all SVM results information.
  - **No Print** — suppresses printing of all SVM results information.
- **Optimization History** — specifies whether to print the detailed optimization history.

SVM Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

SVM Node Details

The Binary Classification Problem

The binary classification problem involves finding a parametric linear or nonlinear function that describes a hyperplane that separates two sets of points in Rm. The hyperplane can be linear or nonlinear, depending on the kernel specification.

Separable Problem for Linear Kernel

The data set has N observations with predictors $x_i = (x_{i1}, . . . ,x_{im})$ and binary responses (target values) $y_i ∈ \{−1, 1\}$, which express cluster (group) membership. (Note: For mathematical convenience, the target values are defined here using -1 and +1, rather than with 0 and 1.)

If the two clusters are linearly separable, then there are the following three planes:

1. Separating Hyperplane: $H_0 : y = w^T x − b = 0$ separates the points of the two groups with target values +1 and -1.
2. Right Hyperplane: $H_1 : y = w^T x − b = +1$ contains at least one point x with target value +1 closest to H_0.

3. Left Hyperplane: $H_2 : y = w^T x - b = -1$ contains at least one point x with target value -1 closest to H_0. The points x on the left and right hyperplane are called support vectors. The orientation of the plane defined such that the distance between H_1 and H_2 is maximal. Also, there should be no points in between H_1 and H_2. The distance from one point on H_1 to H_0 is as follows:

$$\frac{w^T x - b}{w^T w} = \frac{1}{w^T w}$$

You maximize the distance between H_1 and H_2, minimize $w^T w$, with respect to the m + 1 unknown variables *w*, and the scalar quantity *b*, subject to the constraint that no point occupies the space between H_1 and H_2:

$$w^T x - b \geq 1 \quad \text{for} \quad y = +1$$
$$w^T x - b \leq -1 \quad \text{for} \quad y = -1$$

The constraints can be combined into one expression:

$$y(w^T x - b) \geq 1$$

Therefore, the primal separable problem contains m + 1 variables to estimate:

$$(LP1) \quad \min_{w,b} \frac{1}{2} w^T w$$

$$\text{s.t.} \quad y(w^T x - b) \geq 1$$

Note: The parameter estimates *w* and *b* are defined only by the points on H_1 and H_2. The other points can be moved rather freely without changing the optimization result.

Inseparable Problem for Linear Kernel

In practical applications, the data is seldom geometrically separable. With inseparable data, the problem (LP1) has no feasible solution. Therefore, the optimization problem must be modified in order to permit points to exist in between H_1 and H_2, and even on the wrong side of H_0. However, points that cross the boundaries should be penalized.

To compensate, introduce N slack variables $\xi i \geq 0$, i = 1, . . . ,*N*, and modify the constraints to do the following:

$$w_i^T x - b \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$
$$w_i^T x - b \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$

Add a penalty term to the objective function, and obtain the new problem. The new problem contains $N + m + 1$ unknowns ξ_i , *w*, *b* to perform the estimate:

$$(LP2) \quad \min_{w,b} \frac{1}{2} w^T w + C(\sum_{i=1}^{N} \xi_i)^m$$

$$\text{s.t.} \quad y_i(w_i^T x - b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \quad i = 1, \ldots, N$$

The estimate is for a given penalty constant C > 0 and for *m* = 1 or *m* = 2. You can use the SVM Regularization Parameter property to create penalty constants *C*. You can specify a single constant, or use the tuning setting for the Regularization Parameter property, where SVM training is performed for each *C* > 0 value, ordered from small to large. At the end, the best fitted result is scored. If you perform cross validation, the goodness-of-fit is evaluated on the test data set. If cross validation is not enabled, the fit is evaluated on the training data set. In this case, the best result normally corresponds to the largest *C* value. If the *C* value is too large, the solution will be overfitted.

Scoring the Model

For a linear kernel, obtain the linear weight vector *w* ∊ R^m and the linear intercept β. Then the predicted values y_j of the test data set X = (x_j) with the following result obtained.

$$\hat{y}_j = w^T x_j + \beta$$

This is very similar to linear regression. For a general nonlinear kernel, we need the optimal training parameters of the support vectors, $\alpha_i > 0$, the observations of the test data set X = (x_j), and a data set that consists of the support vectors of the training data set Z = (z_k), where z_k are all observations from the data set with parameter $\alpha_i > 0$. The following equation produces the predicted y_j values:

$$\hat{y}_j = \sum_{k=1}^{nsv} (K(x_j, z_k) y_k) \alpha_k + \beta$$

SVM Node Results

You can open the Results window of the **SVM** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

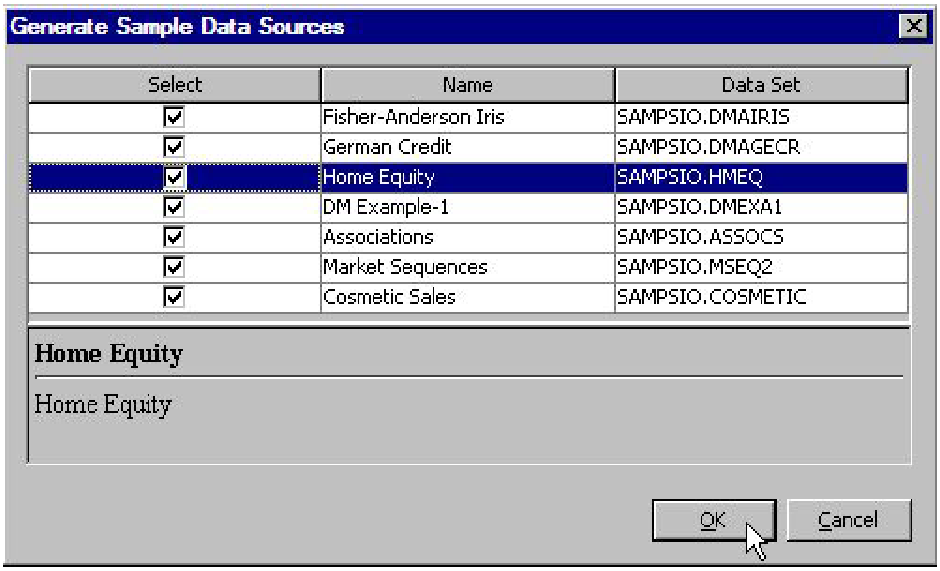
- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the **SVM** node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the **SVM** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.

  - **Variables** — a read-only table of variable meta information about the data set submitted to the **SVM** node . The table includes columns to see the variable's name, use, report, role, and level.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **SVM** node run.
  - **Output** — the SAS output of the **SVM** node run.
  - **Flow Code** — The SAS code used to produce the output that the **SVM** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **SVM** node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — a table of the fit statistics from the model.
  - **Classification Chart** — a bar chart that shows the correct classification of the values of the target variable.
  - **Score Rankings Overlay <TARGET>** — opens the Score Rankings Overlay chart.
  - **Score Rankings Matrix** — opens the Score Rankings Matrix chart.
  - **Score Distribution** — opens the Score Distribution chart.
- **Model**
  - **Histogram: Support Vectors** — opens a histogram that displays the distribution of support vectors.
  - **SVM Fit Statistics** — a table of the SVM fit statistics from the trained model.
  - **Tuning History** — a table of Tuning history. This table is generated when the Tuning Method property is set to Grid Search.
  - **Support Vectors** — displays a table of support vectors with LaGrange multipliers.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Select a Chart Type wizard to modify an existing Results plot or create a Results plot of your own.

SVM Node Examples

Example 1: Using the SVM Node

From the SAS Enterprise Miner menu, select **Help ⇨ Generate Sample Data Sources** to open the Generate Sample Data Sources window.



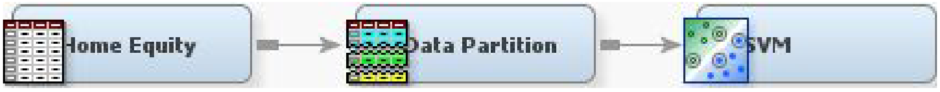
Generate Sample Data Sources

Select	Name	Data Set
☑	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
☑	German Credit	SAMPSIO.DMAGECR
☑	Home Equity	SAMPSIO.HMEQ
☑	DM Example-1	SAMPSIO.DMEXA1
☑	Associations	SAMPSIO.ASSOCS
☑	Market Sequences	SAMPSIO.MSEQ2
☑	Cosmetic Sales	SAMPSIO.COSMETIC

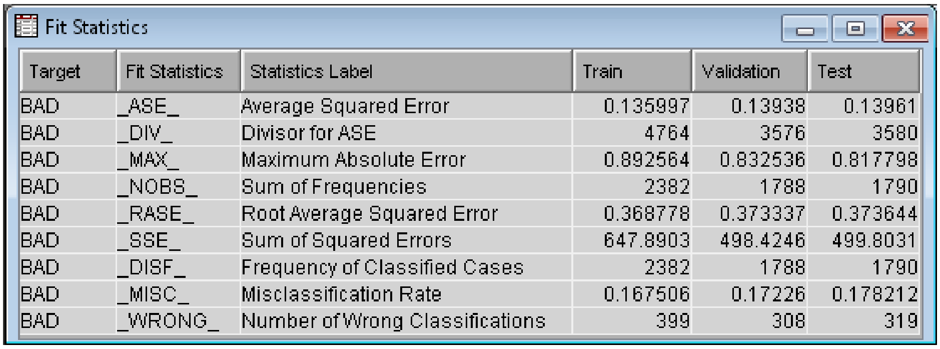
Home Equity

Home Equity

Select **OK** to generate the sample data sources. The new sample data sources automatically populate the Data Sources folder of the SAS Enterprise Miner Project Navigator. Create a new process flow diagram, and then drag the Home Equity data source from the Data Sources folder of the SAS Enterprise Miner Project Navigator onto the diagram workspace. Drag a **Data Partition** node from the **Sample** tab of the SAS Enterprise Miner node toolbar, and an **SVM** node from the **Model** tab of the SAS Enterprise Miner node toolbar, and connect these nodes to the Home Equity data source as shown below. Right-click the **SVM** node and select **Run**.



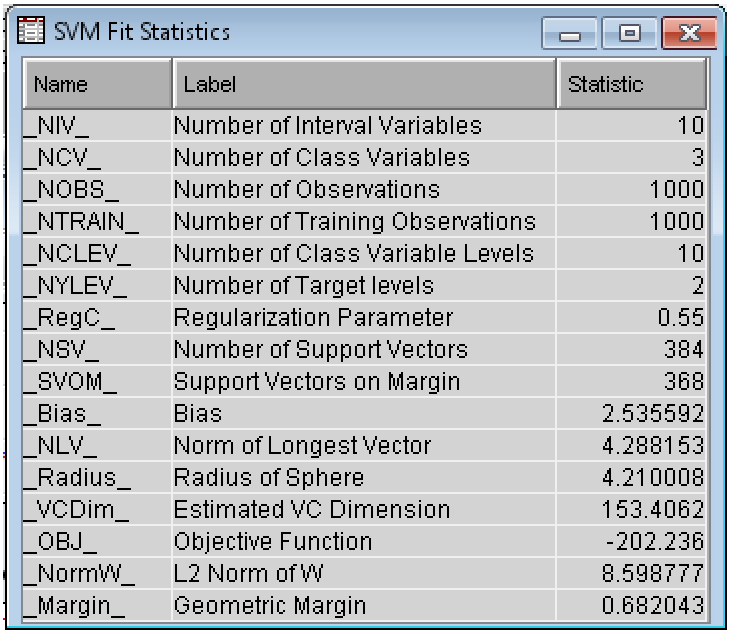
From the SVM Results window, examine the Fit Statistics table.



Fit Statistics

Target	Fit Statistics	Statistics Label	Train	Validation	Test
BAD	_ASE_	Average Squared Error	0.135997	0.13938	0.13961
BAD	_DIV_	Divisor for ASE	4764	3576	3580
BAD	_MAX_	Maximum Absolute Error	0.892564	0.832536	0.817798
BAD	_NOBS_	Sum of Frequencies	2382	1788	1790
BAD	_RASE_	Root Average Squared Error	0.368778	0.373337	0.373644
BAD	_SSE_	Sum of Squared Errors	647.8903	498.4246	499.8031
BAD	_DISF_	Frequency of Classified Cases	2382	1788	1790
BAD	_MISC_	Misclassification Rate	0.167506	0.17226	0.178212
BAD	_WRONG_	Number of Wrong Classifications	399	308	319

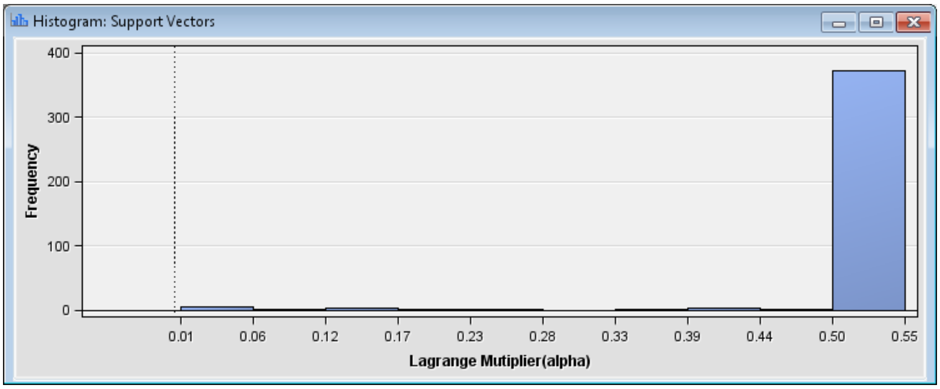
The table below contains selected fit statistics from the SVM training. Since sampling was used, the statistics in the table below are based on 1000 observations, which differs from the number of training observations that were used to generate the previous fit statistics table.

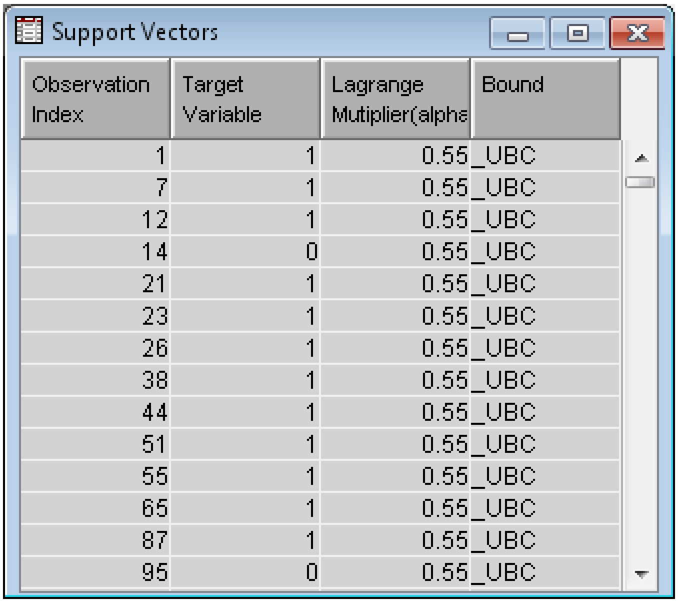


SVM Fit Statistics

Name	Label	Statistic
NIV	Number of Interval Variables	10
NCV	Number of Class Variables	3
NOBS	Number of Observations	1000
NTRAIN	Number of Training Observations	1000
NCLEV	Number of Class Variable Levels	10
NYLEV	Number of Target levels	2
RegC	Regularization Parameter	0.55
NSV	Number of Support Vectors	384
SVOM	Support Vectors on Margin	368
Bias	Bias	2.535592
NLV	Norm of Longest Vector	4.288153
Radius	Radius of Sphere	4.210008
VCDim	Estimated VC Dimension	153.4062
OBJ	Objective Function	-202.236
NormW	L2 Norm of W	8.598777
Margin	Geometric Margin	0.682043

The distribution of nonzero Lagrange Multipliers and Support Vectors are shown as follows:

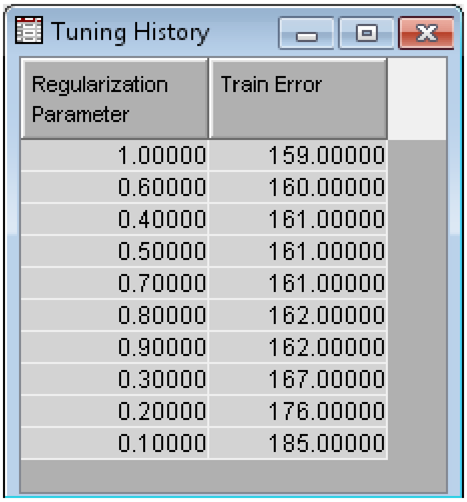




Support Vectors

Observation Index	Target Variable	Lagrange Mutiplier(alpha	Bound
1	1	0.55	_UBC
7	1	0.55	_UBC
12	1	0.55	_UBC
14	0	0.55	_UBC
21	1	0.55	_UBC
23	1	0.55	_UBC
26	1	0.55	_UBC
38	1	0.55	_UBC
44	1	0.55	_UBC
51	1	0.55	_UBC
55	1	0.55	_UBC
65	1	0.55	_UBC
87	1	0.55	_UBC
95	0	0.55	_UBC

When the **Grid Search** tuning method is used, its history is shown as a table.



Regularization Parameter	Train Error
1.00000	159.00000
0.60000	160.00000
0.40000	161.00000
0.50000	161.00000
0.70000	161.00000
0.80000	162.00000
0.90000	162.00000
0.30000	167.00000
0.20000	176.00000
0.10000	185.00000

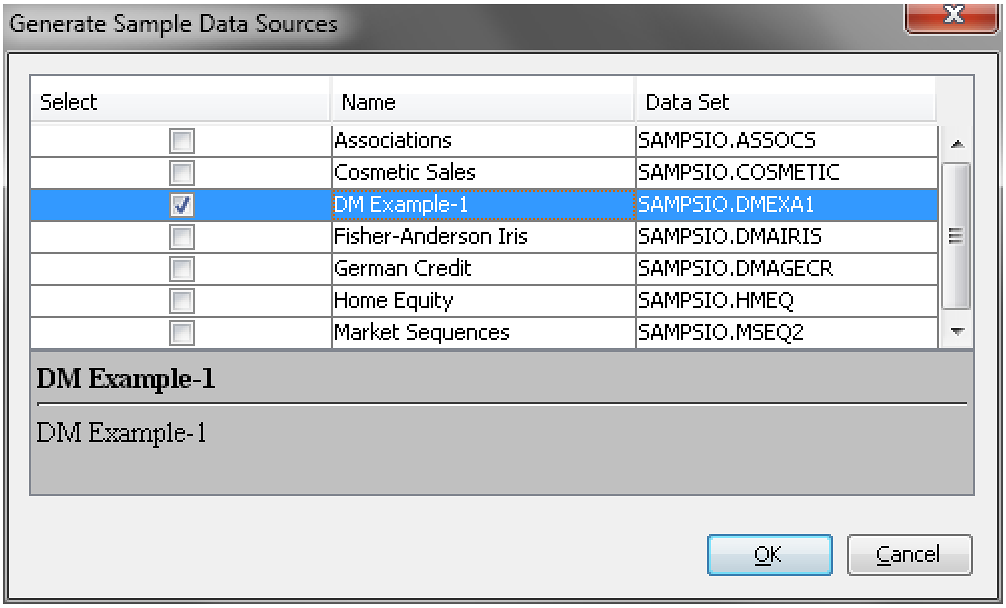
The SAS Output contains the following output from the SVM run:

```
The SVM Procedure
Parameter Tuning Process: Technique=  DQP

---Start Tuning Process: Technique=  DQP---

Terminate pattern search after 2 iterations (3 training runs) with maximum grid
distance=0.1125.

Best solution with C=0.55 selected.

             The SVM Procedure

      Regularization Parameter C=0.55
      Classification Table (Training Data)
     Misclassification=161  Accuracy=  83.90

                          Predicted
Observed           0           1         Missing

0              798.0      3.0000               0
1              158.0     41.0000               0
Missing            0           0               0


The SVM Procedure

Support Vector Classification            C_CLAS
Kernel Function                          Linear
Estimation Method                           DQP
Maximum QP Size                             100
Number of Observations (Train)             1000
Number of Effects                            20
Regularization Parameter C             0.550000
Classification Error (Training)      161.000000
Objective Function                  -202.236319
L1 Loss                            1.261213E-13
Inf. Norm of Gradient              1.412204E-13
Squared Euclidean Norm of w            8.598777
Geometric Margin                       0.682043
Number of Support Vectors                   384
Number of S.Vector on Margin                368
Norm of Longest Vector                 4.288153
Radius of Sphere Around SV             4.210008
Estimated VC Dim of Classifier       153.406184
Linear Kernel Constant (Fit)           2.535592
Linear Kernel Constant (PCE)           2.535592
Number of Kernel Calls                 19711817
```

Example 2: Scoring a Data Set in Foundation SAS

This example explains how to score the results of the SVM node in Foundation SAS. When used to score results from the SVM node, the Score node includes a PROC step that requires SAS Enterprise Miner. This example provides an alternative method to score the results from the SVM node.

This example uses the DM Example-1 data set. To add the DM Example-1 data set, select **Help** ⇨ **Generate Sample Data Sources**. In the Generate Sample Data Sources window, select only the DM Example-1 data set, as shown below. Click **OK**.



Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
□	Cosmetic Sales	SAMPSIO.COSMETIC
☑	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

DM Example-1

DM Example-1

Drag the **DM Example-1** data set onto the diagram workspace. Right click the **DM Example-1** node and select **Edit Variables**. Set the role and measurement level of the following variables to the indicated values:

- ACCTNUM — set the role to **ID**.
- STATECOD — set the role to **Rejected**.
- PURCHASE — set the role to **Target** and the measurement level to **Binary**.

From the **Model** tab, drag an SVM node onto the diagram workspace. Connect the **DM Example-1** node to the **SVM** node. Right click the **SVM** node and select **Run**. In the Confirmation window, click **Yes**.

After a successful run of the **SVM** node, select **Results** in the Run Status window. Select **View** ⇨ **Scoring** ⇨ **SAS Code** to open the SAS Code window. Select **File** ⇨ **Save As** to save the code. This example assumes that you save the code to `C:\myScoreCode.sas`.

When you run the score code, it overwrites the data set that it is scoring. Because you cannot edit the DMEXA1 data set in the SAMPSIO library, you must make a copy of this data set in an alternate library. The code that follows creates a copy of SAMPSIO.DMEXA1 that is named DUP_DMEXA1 and is stored in the library MYLIB. To score the results of the SVM node, run the following code:

```
%let em_score_output=mylib.dup_dmexa1;
data &em_score_output
    set sampsio.dmexa1;
run;
%include "C:\myScoreCode.sas";
```

This code must be run on a machine that has SAS Enterprise Miner installed, but does not need to be run in SAS Enterprise Miner.

References

Cristianni, N. & Shawe-Taylor, J. 2000. *Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, England: Cambridge University Press.

Mangasarian, O.L. & Musicant, D.R. 2000. *Lagrangian Support Vector Machines*. Madison, WI: Data Mining Institute, University of Wisconsin.

Suykens, J.A.K. & Vandewalle, J. "Least Squares Support Vector Machine Classifiers." 1999. *Neural Processing Letters* Volume 9, No. 3: pp. 293-300.

Vapnik, V.N. 1995. *The Nature of Statistical Learning*. New York, NY: Springer.

Chapter 64
TwoStage Node

TwoStage Node

Overview of the TwoStage Node

The TwoStage node belongs to the Model category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA). The TwoStage node enables you to model a class target and an interval target. The interval target variable is usually the value that is associated with a level of the class target. For example, the binary variable PURCHASE is a class target that has two levels: **Yes** and **No**, and the interval variable AMOUNT can be the value target that represents the amount of money that a customer spends on the purchase.

The TwoStage node supports two types of modeling: sequential and concurrent. For sequential modeling, a class model and a value model are fitted for the class target and the interval target, respectively, in the first and the second stages. By defining a transfer function and using the filter option, you can specify how the class prediction for the class target is applied and whether to use all or a subset of the training data in the second stage for interval prediction. The prediction of the interval target is computed from the value model and optionally adjusted by the posterior probabilities of the class target through the bias adjustment option. A posterior analysis that displays the value prediction for the interval target by the actual value and prediction of the class target is also performed. The score code of the TwoStage node is a composite of the class and value models. The value model is used to create assessment plots.

For concurrent modeling, the values of the value target for observations that contain a non-event value for the class target are set to missing prior to training. Then, the TwoStage node fits a neural network model for the class and value target variables simultaneously.

The default TwoStage node fits a sequential model that has a decision tree class model and a neural network value model. Posterior probabilities from the decision tree class model are used as input for the regression value model.

Before reading this document, you should be familiar with the Predictive Modeling, Regression Node, Decision Tree Node, and Neural Network Node documentation.

Variable Requirements for the TwoStage Node

The TwoStage node requires a class target variable and an interval target variable.

TwoStage Node Properties

TwoStage Node General Properties

The following general properties are associated with the TwoStage Node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first TwoStage node added to a diagram will have a Node ID of TwoStage. The second TwoStage node added to a diagram will have a Node ID of TwoStage2, and so on.
- **Imported Data** — accesses the Imported Data — TwoStage window. The Imported Data — TwoStage window contains a lists of the ports that provide data sources to the TwoStage node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — accesses the Exported Data — TwoStage window. The Exported Data — TwoStage window contains a list of the output data ports that the TwoStage node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

TwoStage Node Train Properties

The following train properties are associated with the TwoStage node:

- **Variables** — Use the Variables property of the TwoStage node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, as well as select which variables you want included in reports.

- **Model Type** — Use the Model Type property of the TwoStage node to specify the modeling type that you want to use for training.

  You can choose from the following model types:

  - **Sequential** — The class target is modeled first, and a categorical prediction variable is created. In the second stage of training, the value target is modeled. The prediction variable that you created in the first stage is included as an input in the second stage of training. You use the Class Model and Value Model properties to specify the model types for the first and second stages, respectively. Sequential is the default model type.

  - **Concurrent** — The class target and value target are modeled simultaneously by using the neural network model that you specify with the Concurrent Model property. Observations that contain a non-event value for the class target have the value target role set to Missing prior to training. When you choose Concurrent as your model type, you need to set the Concurrent Model and the Concurrent Hidden Units properties found in the "TwoStage Node Train Properties: Concurrent Training" on page 976 property group.

- **Class Model** — Use the Class Model property of the TwoStage node to specify the model that you want to use to fit a class model for the categorical target variable in the first stage.

  You can choose from the following models:

  - **Tree** — fits a tree model, which depends on the target measurement and the target profile that you defined in a predecessor node. Tree is the default class model.

  - **Regression** — uses default Regression node settings, which depend on the measurement level of the targets.

  - **Neural** — uses default Neural Network node settings.

- **Transfer** — Use the Transfer property of the TwoStage node to determine how the categorical target variable is incorporated into the value model of the second stage. The default selection is **Probability**, which uses the posterior probability of the class event as the input variable to fit the value model. You can change the selection of the transfer function to **Classification**, which uses the predicted classification as the input.

- **Filter** — Use the Filter property of the TwoStage node to specify how you want to exclude certain observations from the training data set in the value model. The default value is None.

  You can use the following filters:

  - **None** — No observations are excluded. All observations from the first stage are used to fit the value model. None is the default filter setting.

  - **Non-Events** — Observations that do not have the class event value for the class target variable are excluded from the training in the second stage.

  - **Misclassified** — Observations that are misclassified in the first stage (the class model) are excluded from the training in the second stage.
  - **Missing Values** — Observations with missing values of the interval target variable are excluded from the training in the second stage.
- **Value Model** — Use the Value Model property of the TwoStage node to specify the type of model that you want to use to fit a value model for the interval target variable in the second stage. The value model that you choose will use default model node settings unless you specifically change the model settings in the Properties panel of the TwoStage node. The model choices are
  - **Tree** — fits a tree model, which depends on the target measurement and the target profile that you defined in a predecessor node. No input variable selection is performed.
  - **Regression** — uses default Regression node settings, which depend on the measurement level of the targets. Regression is the default value model.
  - **Neural** — uses default Neural Network node settings, which depend on the measurement level of the targets.
- **Bias** — Use the Bias property of the TwoStage node to specify how the event posterior probability from the class model and the prediction from the value model are combined to compute the value prediction. You can choose from
  - **None** — No adjustments are made. None is the default setting.
  - **Multiply** — uses the posterior probability to adjust the value prediction of the interval target, as shown in the following equation: P_AMOUNT = P_AMOUNT * P_PURCHASE_Yes. Where, P_AMOUNT is the prediction of the interval target and P_PURCHASE_Yes is the posterior probability.
  - **Filter** — The predicted values of observations with non-event class prediction are set to 0.

TwoStage Node Train Properties: Concurrent Training

- **Concurrent Model** — If your Model Type property is set to Concurrent, use the Concurrent Model property of the TwoStage node to specify the neural network model that you want to use during simultaneous training.
  - **Generalized Linear Model**
  - **Multi-Layer Perceptron** (default)
  - **Ordinary Radial — Equal Width** — ordinary radial basis function with equal widths.
  - **Ordinary Raidal — Unequal Width** — ordinary radial basis function with unequal widths.
  - **Normalized Raidal — Equal Height** — normalized radial basis function with equal heights.
  - **Normalized Radial — Equal Volume** — normalized radial basis function with equal volumes.
  - **Normalized Radial — Equal Width** — normalized radial basis function with equal widths.
  - **Normalized Radial — Equal Width and Height** — normalized radial basis function with equal widths and heights.

  - **Normalized Radial — Unequal Width and Height** — normalized radial basis function with unequal widths and heights.

- **Concurrent Hidden Units** — When the Model Type property is set to Concurrent, use the Concurrent Hidden Units property of the TwoStage node to specify the number, n, of hidden units to use in each hidden layer. The permissible value range is positive integers from 2 to 64. The default value setting is 3.

- **Tree Class Model** — use the following properties to configure tree class models in the TwoStage node. The Tree Class Model properties are only available when the Class Model property is set to **Tree**. Click the ... button to the right of the Tree Class Model property to open a table of the following model properties:

  - **Criterion** — Use the Criterion properties of the TwoStage node to specify the method that you want to use to search for and evaluate candidate splitting rules. The available methods depend on whether the tree model is a class model or a value model.

    The available methods for a tree class model are as follows:

    - **ProbChisq** — the p-value of the Pearson chi-square statistic for the target versus the branches. This method is not available for ordinal targets.

    - **Entropy** — the reduction in the entropy measure of node impurity.

    - **Gini** — the reduction in the Gini measure of node impurity.

  - **Significance Level** — Use the Significance Level property of the TwoStage node to specify the threshold p-value for the worth of a candidate splitting rule. The measure of worth depends on the value of the Splitting Criterion property. For a criterion method based on p-values, the threshold is the maximum acceptable p-value. For other criteria, the threshold is the minimal acceptable increase in the measure of worth. The default setting for the Significance Level property is 0.02. Admissible values are real numbers between 0 and 1.

  - **Split Size** — Use the Split Size property of the TwoStage node to specify the smallest number of training observations that a node must have before it is eligible to be split. Split Size uses a default value of (2*Leaf Size), unless you specify an integer value that is greater than the calculated default value. If no value is specified for the Leaf Size property, then the default Split Size value is calculated as [2*Min(5000,Max(5,N/1000))].

  - **Leaf Size** — Use the Leaf Size property of the TwoStage node to specify the smallest number of training observations that can belong in a leaf. Permissible values are nonnegative integers. The default value is 1.

  - **Maximum Branch** — Use the Maximum Branch property of the TwoStage node to specify the highest number of subsets that a splitting rule can produce. For example, a value of 2 results in binary trees. Permissible values for the Maximum Branch property are integers from 2 to 100. The default value for the Maximum Branch property is 2.

  - **Maximum Depth** — Use the Maximum Depth property of the TwoStage node to specify the maximum number of generations for nodes. The original node, generation 0, is also called the root node. The children of the root node constitute the first generation. Nonnegative integer values are permitted. The default value for the Maximum Depth property is 6.

  - **Number of Rules** — Use the Number of Rules property of the TwoStage node to specify how many splitting rules are saved with each node. The tree only uses one rule, but the remaining rules are saved for comparison. For each node, a number of splitting rules are evaluated as the tree is constructed. A statistic that measures the strength of the rule is computed based on the splitting criterion that you select. Permitted values are nonnegative integers. The default value for the Number of Rules property is 5.

If the Selected Cirterion is...	Then the Computed Statistic is...
Chi-square or F-test	LOGWORTH
Entropy or Gini reduction	WORTH

- **Number of Surrogate Rules** — Use the Number of Surrogate Rules property of the TwoStage node to specify the maximum number of surrogate rules sought in each non-leaf node. A surrogate rule is a backup to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the first surrogate rule is invoked. If the first surrogate also relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all of the surrogates from applying to an observation, then the main rule assigns the observation to the branch it has designated as receiving missing values. Permissible values for the Number of Surrogate Rules property are nonnegative integers. The default value is 0.
- **Missing Values** — Use the Missing Values property of the TwoStage node to configure how observations that contain missing values are handled during a split search.
  - **Use in search** — use the observations in the split search. This is the default method.
  - **Assign to most correlated branch** — assigns the observations that contain missing values to the branch with the smallest residuals.
  - **Assign to largest branch** — assigns the observations that contain missing values to the largest branch.
- Method — Use the Method property of the TwoStage node to specify the selection method that you want to use while constructing the subtree. Choose from the following methods:

Method	Definition	Constructed Subtree
Assessment	Best assessment value (default setting)	Smallest subtree with the best assessment value
Largest	Largest tree in the sequence	Full tree
N	The most indicated number of leaves	Largest subtree with at most N leaves

- **Number of Leaves** — Use the Number of Leaves property of the TwoStage node to specify the number of leaves that you want to use to create the subtree when the SubTree option is set to **No**. Permissible values are nonnegative integers. The default value for the Number of Leaves property is 1.
- **Assessment Measure** — Use the Assessment Measure property of the TwoStage node to specify which assessment measure you want to use to evaluate a tree. You can choose from the following measures:

If you select...	The measure used is...
Decision	The decision method selects the tree that has the largest average profit and smallest average loss, if a profit or loss matrix is defined.
Average Square Error	The average square error method selects the tree that has the smallest average square error.
Misclassification	The misclassification method selects the tree that has the smallest misclassification rate.
Lift	The lift method evaluates the tree based on the prediction of the top n% of the ranked observations.

For detailed information, see the Decision Tree Node Assessment Options.

- **Assessment Fraction** — When the Assessment Measure is set to Average, use the Assessment Fraction property of the TwoStage node to specify the proportion (the top n %) of observations that you want to use to calculate the average assessment from. Permissible values range from 0 to 1.
- **Node Sample** — Use the Node Sample property of the TwoStage node to specify the within-node sample size that you want to use to find splits. If the number of training observations in a node is larger than the number, n, that you specify, then the split search for that node is based on a random sample of size n. The default value for the Node Sample property is 5000.
- **Exhaustive** — Use the Exhaustive property of the TwoStage node to specify the highest number of candidate splits that you want to find in an exhaustive search. If additional candidates have to be considered, a heuristic search is used instead. The Exhaustive property applies to multi-way splits and to binary splits on nominal targets with more than two values. The default value for the Exhaustive property is 5000. Acceptable values are integers between 1 and 2,000,000,000.
- **Bonferroni Adjustment** — Use the Bonferroni Adjustment property of the TwoStage node to indicate whether you want to apply a Bonferroni adjustment to the p-values. The default setting is for the Bonferroni Adjustment property is **Yes**.
- **Time of Kass Adjustment** — When the Bonferroni Adjustment property is set to **Yes**, use the Time of Kass Adjustment property of the TwoStage node to indicate whether you want the Bonferroni adjustment to take place before or after the split is chosen. The default setting for the Time of Kass Adjustment property is **After**.
- **Split Adjustment** — Use the Split Adjustment property of the TwoStage node to indicate whether you want to adjust the p-values for the number of ancestor splits. The default setting for the Split Adjustment property is **Yes**.
- **Inputs** — Use the Inputs property of the TwoStage node to indicate whether you want to adjust the p-values for the number of inputs. The default setting is for the Inputs property is **No**.
- **Number of Inputs** — When the Inputs property is set to **Yes**, use the Number of Inputs property of the TwoStage node to specify the number of inputs that you want to consider uncorrelated. Permissible values are nonnegative integers. The default value for the Number of Inputs property is 1.

  - **Variable Selection** — Use the Variable Selection property of the TwoStage node to indicate whether you want variable selection to be performed. The default setting for the Variable Selection property is **Yes**. This property is available for the tree class model only.

- **Tree Value Model** — use the following properties to configure tree value models in the TwoStage node. The Tree Value Model properties are only available when the Value Model property is set to **Tree**. Select the ... button to the right of the Tree Value Model property to open a table of the following model properties.

  - **Criterion** — Use the Criterion properties of the TwoStage node to specify the method that you want to use to search for and evaluate candidate splitting rules. The available methods for a tree value model are

    - **Variance** — reduction in squared error from node means.

    - **ProbF** — p-value of F-test that is associated with node variance.

  - **Significance Level** — Use the Significance Level property of the TwoStage node to specify the threshold p-value for the worth of a candidate splitting rule. The measure of worth depends on the value of the Splitting Criterion property. For a criterion method based on p-values, the threshold is the maximum acceptable p-value. For other criteria, the threshold is the minimal acceptable increase in the measure of worth. The default setting for the Significance Level property is 0.02. Admissible values are real numbers between 0 and 1.

  - **Missing Values** — Use the Missing Values property of the TwoStage node to configure how observations that contain missing values are handled during a split search.

    - **Use in search** — uses the observations in the split search. This is the default setting.

    - **Assign to most correlated branch** — assigns the observations that contain missing values to the branch with the smallest residuals.

    - **Assign to largest branch** — assigns the observations that contain missing values to the largest branch.

  - **Exhaustive** — Use the Exhaustive property of the TwoStage node to specify the highest number of candidate splits that you want to find in an exhaustive search. If additional candidates have to be considered, a heuristic search is used instead. The Exhaustive property applies to multi-way splits and binary splits on nominal targets with more than two values. The default value for the Exhaustive property is 5000. Acceptable values are integers between 1 and 2,000,000,000.

  - **Leaf Size** — Use the Leaf Size property of the TwoStage node to specify the smallest number of training observations that can belong in a leaf. Permissible values are nonnegative integers. The default value is 1.

  - **Maximum Branch** — Use the Maximum Branch property of the TwoStage node to specify the highest number of subsets that a splitting rule can produce. For example, a value of 2 results in binary trees. Permissible values for the Maximum Branch property are integers from 2 to 100. The default value for the Maximum Branch property is 2.

  - **Maximum Depth** — Use the Maximum Depth property of the TwoStage node to specify the maximum number of generations for nodes. The original node, generation 0, is also called the root node. The children of the root node constitute the first generation. Nonnegative integer values are permitted. The default value for the Maximum Depth property is 6.

- **Node Sample** — Use the Node Sample property of the TwoStage node to specify the within-node sample size that you want to use to find splits. If the number of training observations in a node is larger than the number, n, that you specify, then the split search for that node is based on a random sample of size n. The default value for the Node Sample property is 5000.
- **Number of Rules** — Use the Number of Rules property of the TwoStage node to specify how many splitting rules are saved with each node. The tree only uses one rule, but the remaining rules are saved for comparison. For each node, a number of splitting rules are evaluated as the tree is constructed. A statistic that measures the strength of the rule is computed based on the splitting criterion that you select. Permitted values are nonnegative integers. The default value for the Number of Rules property is 5.

If the Selected Cirterion is...	Then the Computed Statistic is...
Chi-square or F-test	LOGWORTH
Entropy or Gini reduction	WORTH

- **Number of Surrogate Rules** — Use the Number of Surrogate Rules property of the TwoStage node to specify the maximum number of surrogate rules sought in each non-leaf node. A surrogate rule is a backup to the main splitting rule. When the main splitting rule relies on an input whose value is missing, the first surrogate rule is invoked. If the first surrogate also relies on an input whose value is missing, the next surrogate is invoked. If missing values prevent the main rule and all of the surrogates from applying to an observation, then the main rule assigns the observation to the branch it has designated as receiving missing values. Permissible values for the Number of Surrogate Rules property are nonnegative integers. The default value is 0.
- **Split Size** — Use the Split Size property of the TwoStage node to specify the smallest number of training observations that a node must have before it is eligible to be split. Split Size uses a default value of (2*Leaf Size), unless you specify an integer value that is greater than the calculated default value. If no value is specified for the Leaf Size property, then the default Split Size value is calculated as [2*Min(5000,Max(5,N/1000))].
- **Assessment Measure** — Use the Assessment Measure property of the TwoStage node to specify which assessment measure you want to use to evaluate a tree. You can choose from the following measures:

If you select...	The measure used is...
Decision	The decision method selects the tree that has the largest average profit and smallest average loss, if a profit or loss matrix is defined.
Average Square Error	The average square error method selects the tree that has the smallest average square error.
Misclassification	The misclassification method selects the tree that has the smallest misclassification rate.

If you select...	The measure used is...
Lift	The lift method evaluates the tree based on the prediction of the top n% of the ranked observations.

For detailed information, see the Decision Tree Node Assessment Options.

- **Assessment Fraction** — When the Assessment Measure is set to **Average**, use the Assessment Fraction property of the TwoStage node to specify the proportion (the top n %) of observations that you want to use to calculate the average assessment from. Permissible values range from 0 to 1.
- **Method** — Use the Method property of the TwoStage node to specify the selection method that you want to use while constructing the subtree. Choose from the following methods:

Method	Definition	Constructed Subtree
Assessment	Best assessment value (default setting)	Smallest subtree with the best assessment value
Largest	Largest tree in the sequence	Full tree
N	The most indicated number of leaves	Largest subtree with at most N leaves

- **Number of Leaves** — Use the Number of Leaves property of the TwoStage node to specify the number of leaves that you want to use to create the subtree when the SubTree option is set to **No**. Permissible values are nonnegative integers. The default value for the Number of Leaves property is 1.
- **Bonferroni Adjustment** — Use the Bonferroni Adjustment property of the TwoStage node to indicate whether you want to apply a Bonferroni adjustment to the p-values. The default setting is for the Bonferroni Adjustment property is **Yes**.
- **Time of Kass Adjustment** — When the Bonferroni Adjustment property is set to **Yes**, use the Time of Kass Adjustment property of the TwoStage node to indicate whether you want the Bonferroni adjustment to take place before or after the split is chosen. The default setting for the Time of Kass Adjustment property is **After**.
- **Inputs** — Use the Inputs property of the TwoStage node to indicate whether you want to adjust the pp-values for the number of inputs. The default setting is for the Inputs property is **No**.
- **Number of Inputs** — When the Inputs property is set to Yes, use the Number of Inputs property of the TwoStage node to specify the number of inputs that you want to consider uncorrelated. Permissible values are nonnegative integers. The default value for the Number of Inputs property is 1.
- **Split Adjustment** — Use the Split Adjustment property of the TwoStage node to indicate whether you want to adjust the p-values for the number of ancestor splits. The default setting for the Split Adjustment property is **Yes**.

- **Regression Class Model** — You use the following properties to configure regression class models in the TwoStage node. The Regression Class Model properties are only available when the Class Model property is set to Regression. Select the ... button to the right of the Regression Class Model property to open a table of the following model properties.
  - **Force Candidate Effects** — If you decide in advance that you know one or more certain candidate effects is important in predicting the target, you can force the top n entries from the effects list into all candidate models. Use the Force Candidate Effects property of the TwoStage node to specify the integer nn. Permissible values for the Force Candidate Effects property are nonnegative integers. The default value is 0.
  - **Hierarchy Effects** — Use the Hierarchy Effects property of the TwoStage node to specify if only class variables or both class and interval variables are subject to hierarchy. **All** indicates that the hierarchy rule applies to all independent variables. **Class** indicates that the hierarchy rule only applies to class variables. The default setting for the Hierarchy Effects property is **Class**.
  - **Input Coding** — Use the Input Coding properties of the TwoStage node to specify the method for coding class input variables.
    - **GLM** — The non-full-rank General Linear Models (GLM) coding uses parameters to estimate the difference between each level and a reference level. The reference level is the last level when the levels are sorted in ascending numeric or alphabetic order.
    - **Deviation** — The deviation coding uses parameters to estimate the difference between each level and the average across each level. This is the default setting.
  - **Link Function** — Use the Link Function property of the TwoStage node to specify the link function for the regression class model. The Link Function property is not available for a regression value model. Link functions link the response mean to the linear predictor.

    The following link functions are available:
    - **Cloglog** — for logistic regression.
    - **Logit** — for logistic regression. This is the default setting.
    - **Probit** — for logistic regression.
  - **Maximum Number of Steps** — If you set the Selection Default property to **No**, you can use the Maximum Number of Steps property of the TwoStage node to specify the maximum number of steps you that want to allow during stepwise model selection. Permissible values for the Maximum Number of Steps property are nonnegative integers.
  - **Selection Model** — Use the Selection Model property of the TwoStage node to specify the selection method that you want to use.
    - **Backward** — begins with all candidate effects in the model and systematically removes effects that are not significantly associated with the target until no effect in the model meets the Stay Significance Level or the Stop Criterion.
    - **Forward** — begins with no candidate effects in the model and adds effects that are significantly associated with the target until none of the remaining effects meet the Entry Significance Level or the Stop Criterion is met.

  - **Stepwise** — begins with no candidate effects in the model (as in the Forward method) and then systematically adds effects that are significantly associated with the target. After an effect is added to the model, Stepwise evaluates effects in the model and may remove any effect already in the model that is not significantly associated with the target.

    The stepwise process continues until no other effect in the model meets the Stay Significance Level, the Stepwise Stop Criterion is met, or an effect that is added in one step is the only effect that is deleted in the next step. If you choose the Stepwise selection method, then you can use the Maximum Number of Steps property to limit the number of steps before the effect selection process stops.

  - **None** — no model selection method is specified. This is the default setting.

  You can specify a selection criterion if you select a model selection method other than None.

- **Moving Effect Rule** — Use the Moving Effect Rule property of the TwoStage node to determine whether hierarchy is maintained and whether single or multiple effects may enter or leave the model in one step.

  You can choose from the following values:

  - **None** — hierarchy is not maintained. Single effects can enter and leave the model. None is the default setting.
  - **Single** — hierarchy is maintained, and single effects can enter and leave the model.
  - **Multiple** — multiple effects can enter and leave the model, subject to hierarchy.

- **Selection Criterion** — Use the Selection Criterion properties of the TwoStage node to specify the selection criteria that you want to use to select the class models.

  You can choose from the following criteria:

  - **None** — The last model produced by the effects selection method is chosen as the final model. None is the default setting if you did not define a profit or loss matrix with two or more decisions.
  - **Akaike's Information Criterion (AIC)** — AIC = n*ln(SSE/n) + 2p , where n is the number of cases, SSE is the error sum of squares, and p is the number of model parameters. The model with the smallest AIC value is chosen.
  - **Schwarz's Bayesian Criterion (SBC)** — SBC = n*ln(SSE/n) + p*ln(n). The model with the smallest SBC value is chosen.
  - **Validation Error** — chooses the model that has the smallest error rate for the validation data set. For logistic regression models, the error is the negative log-likelihood. For linear regression, the error is the error sum of squares (SSE). This option appears dimmed and is unavailable if a validation predecessor data set is not input to the Regression node.
  - **Validation Misclassification** — chooses the model that has the smallest misclassification rate for the validation data set. This option is unavailable and appears dimmed if a validation predecessor data set is not input to the Regression node.
  - **Cross Validation Error** — chooses the model that has the smallest cross validation error rate. For logistic regression models, the error is the negative log-likelihood.

  - **Cross Validation Misclassification** — chooses the model that has the smallest cross validation misclassification rate for the training data set.
  - **Profit/Loss** — chooses the model that maximizes the profit or minimizes the loss for the cases in the training data set. If the decision matrix contains less than two decisions, then the profit or loss information is not considered in the model selection process. When there are less than two decisions, the None criterion is actually used for model selection. To use the Profit/Loss criterion, you must define a profit or loss matrix in the target profile for the target. The profit or loss values are adjusted for the prior probabilities that you specified in the prior vector of the target profile. To learn more about defining a target profile, read Layout of a Target Profile.
  - **Validation Profit/Loss** — The node chooses the model that maximizes the profit or minimizes the loss for the cases in the validation data set. If the decision matrix contains less than two decisions, then the profit or loss information is not considered in the model selection process. When there are less than two decisions, the None criterion is actually used for model selection. To use the Profit/Loss criterion, you must define a profit or loss matrix in the target profile for the target. The profit or loss values are adjusted for the prior probabilities that you specify in the prior vector of the target profile. To learn more about defining a target profile, read Layout of a Target Profile.
  - **Cross Validation Profit/Loss** — chooses the model that maximizes the cross-validation profit or minimizes the cross-validation loss. You must also define a profit or loss matrix for the target to use this criterion.

  Note: To use any of the Profit/Loss criteria, you must define a profit or loss matrix in the target profile for the target.

- **Use Selection Defaults** — To use non-default values for your model selection criteria, set the Selection Default properties of the TwoStage node to **No**, and then set the corresponding selection criteria property to the value that you want. The default setting for the Selection Default property is **Yes**.
- **Sequential Order** — When the Sequential Order property of the TwoStage node is set to **Yes**, effects are added or removed according to the order that is specified in the Model statement. When the property is set to **No**, the Model statement order is ignored. The default setting for the Sequential Order property is **No**.
- **Entry Significance Level** — If you set the Selection Default property to No, you can use the Entry Significance Level property of the TwoStage node to specify the entry significance level that you want to use to add variables in forward or stepwise regressions. Permissible values range from 00 to 1. The default value for the Entry Significance Level property is 0.05.
- **Stay Significance Level** — If you set the Selection Default property to **No**, you can use the Stay Significance Level property of the TwoStage node to specify the stay significance level that you want to use to remove variables in backward or stepwise regressions. Permissible values range from 0 to 1. The default value for the Stay Significance Level property is 0.05.
- **Start Variable Number** — If you set the Use Selection Defaults property to **No**, you can use the Start Variable Number property of the TwoStage node to specify the number of effects to use in the first model. The start value of n specifies that the first n effects from the beginning of the effects list are to be used in the first model. The forward and stepwise selection methods use default start values of 0. The backward selection method, by default, starts with the total number of candidate effects.

  - **Stop Variable Number** — If you set the Selection Default property to **No**, you can use the Stop Variable Number property of the TwoStage node to specify the number of effects that you want to appear in the model using a Backward or Forward selection process. The following table shows the definitions and default values of the Stop value for the **Backward** and **Forward** methods.

Method	Definition	Default Stop Value
Forward	The maximum number of effects to appear in the final model	Total number of input variables
Backward	The minimum number of effects to appear in the final model	0

    The Stop option is not used in the Stepwise selection method.

    Note: Other criteria may apply and terminate the effect selection before reaching the Stop criterion.

  - **Suppress Intercept** — Use the Suppress Intercept property of the TwoStage node to indicate whether you want to suppress the intercept of the regression models. This property is ignored for ordinal targets. The default setting for the Suppress Intercept property is **No**.

- **Regression Value Model** — You use the following properties to configure regression value models in the TwoStage node. The Regression Value Model properties are only available when the Value Model property is set to Regression. Select the ... button to the right of the Regression Value Model property to open a table of the following model properties.

  - **Input Coding** — Use the Input Coding properties of the TwoStage node to specify the method for coding value input variables.

    - **GLM** — The non-full-rank General Linear Models (GLM) coding uses parameters to estimate the difference between each level and a reference level. The reference level is the last level when the levels are sorted in ascending numeric or alphabetic order.

    - **Deviationn** — The deviation coding uses parameters to estimate the difference between each level and the average across each level. **Deviation** is the default setting.

  - **Suppress Intercept** — Use the Suppress Intercept property of the TwoStage node to indicate whether you want to suppress the intercept of the regression models. This property is ignored for ordinal targets. The default setting for the Suppress Intercept property is **No**.

  - **Selection Model** — Use the Selection Model property of the TwoStage node to specify the selection method that you want to use. Selection methods are available for both regression class models and regression value models.

    The available selection methods are as follows:

    - **Backward** — begins with all candidate effects in the model and systematically removes effects that are not significantly associated with the target until no effect in the model meets the Stay Significance Level or the Stop Criterion.

  - **Forward** — begins with no candidate effects in the model and adds effects that are significantly associated with the target until none of the remaining effects meet the Entry Significance Level or the Stop Criterion is met.
  - **Stepwise** — begins with no candidate effects in the model (as in the Forward method) and then systematically adds effects that are significant associated with the target. After an effect is added to the model, Stepwise evaluates effects in the model and may remove any effect already in the model that is not significantly associated with the target.

    The stepwise process continues until no other effect in the model meets the Stay Significance Level, the Stepwise Stop Criterion is met, or an effect that is added in one step is the only effect that is deleted in the next step. If you choose the Stepwise selection method, then you can use the Maximum Number of Steps property to limit the number of steps before the effect selection process stops.
  - **None** — no model selection method is specified. This is the default setting.

- **Selection Criterion** — Use the Selection Criterion properties of the TwoStage node to specify the selection criteria that you want to use to select the class and value models.

  You can choose from the following criteria:

  - **None** — the last model produced by the effects selection method is chosen as the final model. None is the default if you did not define a profit or loss matrix with two or more decisions.
  - **Akaike's Information Criterion (AIC)** — AIC = n*ln(SSE/n) + 2p , where n is the number of cases, SSE is the error sum of squares, and p is the number of model parameters. The model with the smallest AIC value is chosen.
  - **Schwarz's Bayesian Criterion (SBC)** — SBC = n*ln(SSE/n) + p*ln(n), where n is the number of cases, SSE is the error sum of squares, and p is the number of model parameters. The model with the smallest SBC value is chosen.
  - **Validation Error** — chooses the model that has the smallest error rate for the validation data set. For logistic regression models, the error is the negative log-likelihood. For linear regression, the error is the error sum of squares (SSE). This option appears dimmed and is unavailable if a validation predecessor data set is not input to the Regression node.
  - **Cross Validation Error** — chooses the model that has the smallest cross validation error rate. For logistic regression models the error is the negative log-likelihood.

- **Use Selection Defaults** — To use non-default values for your model selection criteria set the Use Selection Defaults property of the TwoStage node to **No**, and then set the corresponding selection criteria property to the value that you want. The default setting for the Use Selection Defaults property is **Yes**.

- **Entry Significance Level** — If you set the Use Selection Defaults property to **No**, you can use the Entry Significance Level property of the TwoStage node to specify the entry significance level that you want to use to add variables in forward or stepwise regressions. Permissible values range from 0 to 1. The default value for the Entry Significance Level property is 0.05.

- **Stay Significance Level** — If you set the Use Selection Defaults property to **No**, you can use the Stay Significance Level property of the TwoStage node to specify the stay significance level that you want to use to remove variables in backward or stepwise regressions. Permissible values range from 0 to 1. The default value for the Stay Significance Level property is 0.05.

- **Force Candidate Effects** — If you decide in advance that you know one or more certain candidate effects is important in predicting the target, you can force the top n entries from the effects list into all candidate models. Use the Force Candidate Effects property of the TwoStage node to specify the integer n. Permissible values for the Force Candidate Effects property are nonnegative integers. The default value is 0.
- **Maximum Number of Steps** — If you set the Use Selection Defaults property to **No**, you can use the Maximum Number of Steps property of the TwoStage node to specify the maximum number of steps you that want to allow during stepwise model selection. Permissible values for the Maximum Number of Steps property are nonnegative integers.
- **Start Variable Number** — If you set the Use Selection Defaults property to **No**, you can use the Start Variable Number property of the TwoStage node to specify the number of effects to use in the first model. The Start value select the first n effects from the beginning of the effects list as the first model. The Forward and Stepwise selection methods use default Start values of 0. The Backward selection method, by default, starts with the total number of candidate effects.
- **Stop Variable Number** — If you set the Use Selection Defaults property to **No**, you can use the Stop Variable Number property of the TwoStage node to specify the number of effects that you want to appear in the model using a Backward or Forward selection process. The following table shows the definitions and default values of the Stop value for the Backward and Forward methods.

Method	Definition	Default Stop Value
Forward	The maximum number of effects to appear in the final model	Total number of input variables
Backward	The minimum number of effects to appear in the final model	0

The Stop option is not used in the Stepwise selection method.

Note: Other criteria may apply and terminate the effect selection before reaching the Stop criterion.

- **Hierarchy Effects** — Use the Hierarchy Effects property of the TwoStage node to specify if only class variables or both class and interval variables are subject to hierarchy. **All** indicates that the hierarchy rule applies to all independent variables. **Class** indicates that the hierarchy rule only applies to class variables. The default setting for the Hierarchy Effects property is **Class**.
- **Moving Effect Rule** — Use the Moving Effect Rule property of the TwoStage node to determine whether hierarchy is maintained and whether single or multiple effects may enter or leave the model in one step.

  You can choose from the following values:

  - **None** — hierarchy is not maintained. Single effects can enter and leave the model. This is the default setting.
  - **Single** — hierarchy effects can enter and leave the model.

        - **Multiple** — multiple effects can enter and leave the model, subject to hierarchy.

    - **Sequential Order** — When the Sequential Order property of the TwoStage node is set to **Yes**, effects are added or removed according to the order that is specified in the model statement. When the property is set to **No**, the model statement order is ignored. The default setting for the Sequential Order property is **No**.

- **Neural Class Model** — You use the following properties to configure neural class models in the TwoStage node. The Neural Class Model properties are only available when the Class Model property is set to Neural. Select the ... button to the right of the Neural Class Model property to open a table of the following model properties.

    - **Architecture** — Use the Architecture property of the TwoStage node to specify the network architecture that you want to use to train your data. Your choices are

        - **Generalized Linear Model**

        - **Multi-Layer Perceptron** (default)

        - **Ordinary Radial — Equal Width** — ordinary radial basis function with equal widths.

        - **Ordinary Raidal — Unequal Width** — ordinary radial basis function with unequal widths.

        - **Normalized Raidal — Equal Height** — normalized radial basis function with equal heights.

        - **Normalized Radial — Equal Volume** — normalized radial basis function with equal volumes.

        - **Normalized Radial — Equal Width** — normalized radial basis function with equal widths.

        - **Normalized Radial — Equal Width and Height** — normalized radial basis function with equal widths and heights.

        - **Normalized Radial — Unequal Width and Height** — normalized radial basis function with unequal widths and heights.

    - **Direct Connection** — When the Direct Connection property of the TwoStage node is set to **Yes**, each input unit is connected to each hidden unit and each output unit. Normally, input units are not connected directly to the output units. The default setting for the Direct Connection property is **No**.

    - **Maximum Iterations** — Use the Maximum Iterations property of the TwoStage node to specify the maximum number of iterations that you want to perform during neural training. Permissible values are integers from 1 to 50. The default value of the Maximum Iterations property is 50.

    - **Number of Hidden Units** — Use the Number of Hidden Units property of the TwoStage node to specify the number of hidden units that you want to add when you add a hidden layer to your neural network. Permissible values are integers from 2 to 64. The default value of the Number of Hidden Units property is 3.

    - **Training Technique** — Use the Training Technique property of the TwoStage node to specify the neural network training algorithm that you want to use.

        - **Default** — the default method depends on the number of weights applied at execution.

        - **Trust-Region** — use this algorithm for small and medium optimization problems up to 40 parameters.

        - **Levenberg-Marquardt** — use this algorithm for smooth least squares objective functions, with up to 100 parameters.
        - **Quasi-Newton** — use this algorithm for medium optimization problems with up to 500 parameters.
        - **Conjugate Gradient** — use this algorithm for large data mining problems with over 500 parameters.
        - **Back Prop** — Standard batch backprop.
        - **RProp** — Uses a separate learning rate for each weight, and adapts all of the learning rates during training.
        - **QProp** — a Newton-like method, but it uses a diagonal approximation to the Hessian matrix. The default value depends on the number of weights applied during execution.
    - **Enable** — When the Preliminary Training property of the TwoStage node is set to **Yes**, preliminary optimization is performed. You must specify the number, n, of preliminary runs that you want to perform using the Preliminary Runs property. If desired, you can specify the maximum number of iterations using the Maximum Iterations property. The default setting of the Preliminary Training property is **No**.
    - **Number of Runs** — When the Preliminary Training property is set to **Yes**, use the Number of Runs property of the TwoStage node to specify the number of preliminary runs that you want to perform. Permissible values are integers greater than 1. The default value for the Number of Runs property is 5.
    - **Maximum Iterations** — When the Preliminary Training property is set to **Yes**, use the Maximum Iterations property of the TwoStage node to specify the maximum number of iterations that you want to perform during preliminary training. Permissible values are any integer from 1 to 50. The default value for the Maximum Iterations property is 10.
- **Neural Value Model** — You use the following properties to configure neural value models in the TwoStage node. The Neural Value Model properties are only available when the Value Model property is set to Neural. Select the ... button to the right of the Neural Value Model property to open a table of the following model properties.
    - **Architecture** — Use the Architecture property of the TwoStage node to specify the network architecture that you want to use to train your data. Your choices are
        - **Generalized Linear Model**
        - **Multi-Layer Perceptron** (default)
        - **Ordinary Radial — Equal Width** — ordinary radial basis function with equal widths.
        - **Ordinary Raidal — Unequal Width** — ordinary radial basis function with unequal widths.
        - **Normalized Raidal — Equal Height** — normalized radial basis function with equal heights.
        - **Normalized Radial — Equal Volume** — normalized radial basis function with equal volumes.
        - **Normalized Radial — Equal Width** — normalized radial basis function with equal widths.
        - **Normalized Radial — Equal Width and Height** — normalized radial basis function with equal widths and heights.

  - **Normalized Radial — Unequal Width and Height** — normalized radial basis function with unequal widths and heights.

- **Direct Connection** — When the Direct Connection property of the TwoStage node is set to **Yes**, each input unit is connected to each hidden unit and each output unit. Normally, input units are not connected directly to output units. The default setting for the Direct Connection property is **No**.

- **Maximum Iterations** — Use the Maximum Iterations property of the TwoStage node to specify the maximum number of iterations that you want to perform during neural training. Permissible values are integers from 1 to 50. The default value of the Maximum Iterations property is 50.

- **Number of Hidden Units** — Use the Number of Hidden Units property of the TwoStage node to specify the number of hidden units that you want to add when you add a hidden layer to your neural network. Permissible values are integers from 2 to 64. The default value of the Number of Hidden Units property is 3.

- **Training Technique** — Use the Training Technique property of the TwoStage node to specify the neural network training algorithm that you want to use.

  - **Default** — the default method depends on the number of weights applied at execution.

  - **Trust-Region** — use this algorithm for small and medium optimization problems up to 40 parameters.

  - **Levenberg-Marquardt** — use this algorithm for smooth least squares objective functions, with up to 100 parameters.

  - **Quasi-Newton** — use this algorithm for medium optimization problems with up to 500 parameters.

  - **Conjugate Gradient** — use this algorithm for large data mining problems with over 500 parameters.

  - **Back Prop** — Standard batch backprop.

  - **RProp** — Uses a separate learning rate for each weight, and adapts all of the learning rates during training.

  - **QProp** — a Newton-like method, but it uses a diagonal approximation to the Hessian matrix. The default value depends on the number of weights applied during execution.

- **Enable** — When the Preliminary Training property of the TwoStage node is set to **Yes**, preliminary optimization is performed. You must specify the number, n, of preliminary runs that you want to perform using the Number of Runs property. If desired, you can specify the maximum number of iterations using the Maximum Iterations property. The default setting of the Preliminary Training property is **No**.

- **Number of Runs** — When the Preliminary Training property is set to **Yes**, use the Number of Runs property of the TwoStage node to specify the number of preliminary runs that you want to perform. Permissible values are integers greater than 1. The default value for the Number of Runs property is 5.

- **Maximum Iterations** — When the Preliminary Training property is set to **Yes**, use the Maximum Iterations property of the TwoStage node to specify the maximum number of iterations that you want to perform during preliminary training. Permissible values are any integer from 1 to 50. The default value for the Maximum Iterations property is 10.

TwoStage Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

TwoStage Node Results

You can open the Results window of the TwoStage node after a successful run by clicking **Results** in the Run Status — Run completed window. You can also open the window by right-clicking the node in the Diagram Workspace and clicking Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu in the Results — TwoStage window to view the following results:

- **Properties**
  - **Settings** — displays a window with a read-only table of the TwoStage node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the TwoStage node run. Information on the run start time, run duration, and run completion status is displayed in this window.
  - **Variables** — displays a table of the variables in the training data set.
  - **Train Code** — displays the code that Enterprise Miner used to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**
  - **Log** — displays the SAS log of the TwoStage run.
  - **Output** — displays the SAS output of the TwoStage run. SAS output for the TwoStage node includes the following:
    - variables summary
    - model events table
    - decision matrix
    - predicted and decision variables
    - target frequency tables for train and validation data sets

    - rip leaf table
    - classification tables for train and validation data sets
    - cleanup model table
    - fit statistics for train, validation, and test data sets
    - decision tables for train and validate data sets
    - event classification table
    - assessment score rankings for train and validation data sets
    - assessment score distribution tables for train and validate data sets
  - **Flow Code** — displays the SAS code used to produce the output that the TwoStage node passes to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — displays the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the TwoStage node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — The Fit Statistics table in the TwoStage node displays statistics from both the class and the value models. Depending on the type of value model (tree, regression, or neural network), different statistics are displayed in the table. See "Decision Tree Node Results" on page 750 , "Regression Node Results" on page 934 , and "Neural Network Node Results" on page 897 for more information.

    The Fit Statistics table displays the following statistics for the training, validation, and test data sets (if available):

    - _AVERR_ — Average Error Function
    - _DFE_ — Degrees of Freedom for Error
    - _DIV_ — Divisor for _ASE_
    - _ERR_ — Error Function
    - _MISC_ — Misclassification Rate
    - _NOBS_ — Sum of Frequencies
    - _WRONG_ — Number of Wrong Classifications
    - _SUMW_ — Sum of Case Weights Times Freq.
    - _MAX_ — Maximum Absolute Error
    - _SSE_ — Sum of Squared Errors
    - _ASE_ — Average Squared Error
    - _RASE_ — Root Average Squared Error
    - _DFT_ — Total Degrees of Freedom
    - _DFM_ — Model Degrees of Freedom
    - _NW_ — Number of Estimated Weights
    - _AIC_ — Akaike's Information Criterion

- _SBC_ — Schwarz's Bayesian Criterion
- _FPE_ — Final Prediction Error
- _MSE_ — Mean Squared Error
- _RFPE_ — Root Final Prediction Error
- _RMSE_ — Root Mean Squared Error

- **Residual Statistics** — The Residual Statistics displays a box-and-whisker plot for the residual measurements when the target is interval.
- **Score Rankings Matrix** — The Score Rankings Matrix plot overlays the selected statistics for standard, baseline and best models in a lattice that is defined by the training and validation data sets. The Score Ranking Matrix plots
  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response
  - Cumulative % Captured Response.
- **Score Rankings Overlay** — In a Score Rankings Overlay chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from highest expected profit to lowest expected profit (or from lowest expected loss to highest expected loss). Then the sorted observations are grouped into deciles and observations in a decile are used to calculate the statistics that are plotted in decile charts. By default, the horizontal axis of a Score Rankings Overlay chart displays the deciles (groups) of the observations.

  The vertical axis displays the following values and their mean, minimum, and maximum (if any):
  - Posterior probability of target event
  - Number of events
  - Cumulative and noncumulative lift values
  - Cumulative and noncumulative % response
  - Cumulative and noncumulative % captured response
  - Gain
  - Actual profit or loss
  - Expected profit or loss.
- **Score Distribution Chart** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score depends on the prediction of the target and the number of buckets used. For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets. The Score Distribution chart of a useful model shows a higher percentage of events for higher model

    score and a higher percentage of nonevents for lower model scores. For interval targets, observations are grouped into bins, based on the actual predicted values of the target. The default chart choice is Percentage of Events. The chart choices are

        - Percentage of Events — for categorical targets
        - Number of Events — for categorical targets
        - Cumulative Percentage of Events — for interval targets
        - Mean for Predicted — for interval targets
        - Max. for Predicted — for interval targets
        - Min. for Predicted — for interval targets

    - **Class Model Tree** — The Class Model Tree is a tree model that is created for a class target variable. This menu item is available only when you fit a tree class model.

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph wizard to modify an existing results plot or to create a results plot of your own.

TwoStage Node Output Data

After you have successfully run the TwoStage node in a diagram, you can view the output data sets that it generates. With the TwoStage node selected in the Diagram Workspace, select the ... button to the right of the Exported Data property. This opens a table of the exported TwoStage data.

If data exists for an exported data set, you can select the row in the table and click one of the following:

- **Browse** to open a window where you can browse the first 100 observations of the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data set. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

By default, the scored score data set is exported to successor nodes.

In addition to the original input, target, and frequency variables from predecessor nodes, the scored data set (training, validation, test, or sco re) contains the following new variables:

- BP_ or BL_ — best profit or loss, depending on the target profile.
- CP_ or CL_ — computed profit or loss, depending on the target profile.
- D_ — the decision variable.
- EP_ or EL_ — expected profit or loss, depending on the target profile.
- I_ — the normalized category that the case is classified into.
- S_ — the standardized input variables. This variable is generated only for neural models.

- H<number> — the hidden units. This variable is generated only for neural models.
- U_ — the u-normalized category that the case is classified into.
- P_ — the posterior probabilities for categorical targets or the predicted values for interval targets.
- R_ — residuals.

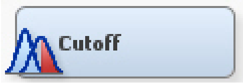
For detailed information about these variables, read the Predictive Modeling section.

Part 14

Node Reference: Assess Nodes

Chapter 65
Cutoff Node

Cutoff Node



Overview of the Cutoff Node

The Cutoff node belongs to the Assess category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA).

The node provides tabular and graphical information to assist users in determining appropriate probability cutoff point(s) for decision making with binary target models. The establishment of a cutoff decision point entails the risk of generating false positives and false negatives, but an appropriate use of the Cutoff node can help minimize those risks.

You will typically run the node at least twice. In the first run, you obtain all the plots and tables. In a subsequent runs, you can change the values of the Cutoff Method and Cutoff User Input properties, customizing the plots, until an optimal cutoff value is obtained.

Cutoff Node and Data Sets with Missing Values

If an observation contains missing values, the Cutoff node excludes the observation from the analysis. If the input data set that you want to perform variable clustering on contains many observations with missing values, it might be beneficial to use the Replacement or Impute nodes to replace or impute missing variable values before submitting the data set to the Cutoff node.

Cutoff Node Properties

Cutoff Node General Properties

The following general properties are associated with the Cutoff Node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Cutoff node added to a diagram will have a Node ID of CUT. The second Cutoff node added to a diagram will have a Node ID of CUT2, and so on.
- **Imported Data** — provides access to the Imported Data — Cutoff window. The Imported Data — Cutoff window contains a list of the ports that provide data sources to the Cutoff node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — provides access to the Exported Data — Cutoff window. The Exported Data — Cutoff window contains a list of the output data ports that the Cutoff node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Cutoff Node Train Properties

The following train properties are associated with the Cutoff node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the Cutoff node. Select the ... button to open a window containing the variables table. You can set **Use** and **Report** variable values in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or distribution plot.
- **Depth Scale %** — provides finer details in plotting of curves in increments of 1%, 0.1%, 0.01%, and 0.001%

Cutoff Node Score Properties

The following score properties are associated with the Cutoff node:

- **Cutoff User Input** — The default value is .5, with a range between .01 and .99. The value chosen by the user appears as a vertical line in graphs that contain cutoffs on the x-axis. To change the value simply type in the desired cutoff value and press ENTER. If you fail to press ENTER, the change will not be recognized and the previous value will be used.
- **Cutoff Method** — specifies the method to be used for determining the cutoff value.
    - **User Input** — enter or accept the value in the Cutoff User Input property. This is the default method.
    - **Maximum KS Statistic** — the cutoff value is either the training data set prior, or the prior selected by the user in the target profiler.
    - **Minimum Misclassification Cost Training Prior** — Assuming an equal cost of a False Positive (FP) and a False Negative (FN), 1 by default, the minimal misclassification cost is given at the cutoff where FP*(1 - prior) + FN*prior is minimized, with FP and FN being either 0 or 1.
    - **Maximum True Pos Rate** — this method maximizes sensitivity, which is defined as the proportion of true positives or the proportion of cases correctly identified.
    - **Maximum Event Precision From Training Prior** — this method maximizes the objective function: % true predicted events / (true predicted + false predicted).
    - **Event Precision Equal Recall** — With precision defined as % true predicted events / (true predicted + false predicted) and recall defined as the event classification rate, this method chooses the point at which precision and recall are equal.
    - **Maximum Cumulative Profit** — Probabilities are sorted in descending order and the profit per decile is accumulated. A cutoff point is chosen that maximizes this accumulated profit. Profit is defined in the target profiler.

Cutoff Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Cutoff Node Results

You can open the Results window of the Cutoff node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- Properties
  - **Settings** — displays a window with a read-only table of the configuration information in the Cutoff node properties. The information was captured when the node was last run.
  - **Run Status** — displays the status of the Cutoff node run. Information about the run start time, run duration, and completion status is displayed in this window.
  - **Variables** — displays a read-only table of variable meta information on the data set submitted to the Cutoff node. The table includes columns for the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — displays the code that Enterprise Miner uses to train the node.
  - **Notes** — displays (in read-only mode) any notes that were previously entered in the General Properties - Notes window.
- **SAS Results**
  - **Log** — the SAS log of the Cutoff node run.
  - **Output** — the SAS output of the Cutoff node includes
    - Training output — variable summaries, model events, decision matrices, predicted variables, and decision variables
    - Score output — There is no default score output for the Cutoff Node.
    - Report output — cutoff diagnostics with prior simulations and a gains table
  - **Flow Code** — the SAS code used to produce the output that the Cutoff node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the Cutoff node does not generate SAS code. The SAS Code menu item is dimmed and unavailable in the Cutoff Results window.
  - **PMML Code** — the Cutoff node does not generate PMML code.
- **Analytical Results**
  - **Cutoff diagnostics with prior simulations** — displays a read-only table of the following:
    - Prior probability obtained from
    - Relative cost of FN over FP
    - Prior Probability
    - Min cost of classification
    - Optimal Cutoff
  - **Model Diagnostics** — displays a read-only table of the following information:
    - CUTOFF

    - Cumulative Expected Profit
    - Counts of True Positives
    - Counts of False Positives
    - Count of True Negatives
    - Counts of False Negatives
    - Counts of Predicted Positive
    - Counts of Predicted Negative
    - Counts of false positives and negatives
    - Counts of true positives and negatives
    - Overall Classification rate
    - Change Count True Positives
    - Change Count False Positives
    - True Positive Rate
    - True Negative Rate
    - False Positive Rate
    - Misclassification cost prior (observed prior) equal cost structure
    - Misclassification cost prior 0.1 equal cost structure
    - Misclassification cost prior 0.2 equal cost structure
    - Misclassification cost prior 0.3 equal cost structure
    - Misclassification cost prior 0.4 equal cost structure
    - Misclassification cost prior 0.5 equal cost structure
    - Event Precision Rate
    - Non-Event Precision Rate
    - Overall Precision Rate
    - Data Role
  - **Gains Table** — displays a read-only table of the following information:
    - Data Role
    - Percentile
    - Cumulative % captured response
    - Gain
    - Lift
    - Cumulative Lift
    - % Response
    - % Cumulative Response
    - % Captured response
    - Posterior Probability Max
- **Classification Rates**

  - **Priors and Minimal Cost** — displays a graph with the optimal cutoff on the vertical axis and prior probabilities on the horizontal axis.
  - **Overall Rates** — displays an overlay graph with the overall classification rate, true positive rate, false positive rate and true negative rate on the vertical axis and the cutoff on the horizontal axis. The cutoff point that is determined by the Cutoff Method is displayed as a vertical line emanating from the horizontal axis.
  - **True Rates** — displays a graph of the true positive rate and true negative rate on the vertical axis and the cutoff on the horizontal axis. The cutoff point that is determined by the Cutoff Method is displayed as a vertical line emanating from the horizontal axis.
  - **Positive Rates** — displays a graph of the true positive rate and false positive rate on the vertical axis and the cutoff on the horizontal axis. The cutoff point that is determined by the Cutoff Method is displayed as a vertical line emanating from the horizontal axis.
  - **Receiver Operating Characteristics** — displays a graph of the true positive rate versus the false positive rate.
- **Precision Rates**
  - **Precision Recall Cutoff Curve** — displays a graph with the event precision rate and true positive rate on the vertical axis and the cutoff on the horizontal axis. The cutoff point that is determined by the Cutoff Method is displayed as a vertical line emanating from the horizontal axis.
  - **All Precision Rates** — displays a graph with the event precision rate, non-event precision rate, and overall precision rate on the vertical axis and the cutoff on the horizontal axis. The cutoff point that is determined by the Cutoff Method is displayed as a vertical line emanating from the horizontal axis.
  - **Precision Recall Curve** — displays a graph of the event precision rate versus the true positive rate.
- **Table** — displays a table that contains the underlying data that is used to produce a chart.
- **Plot** — use the Graph wizard to modify an existing Results plot or to create a Results plot of your own.

Cutoff Node Example

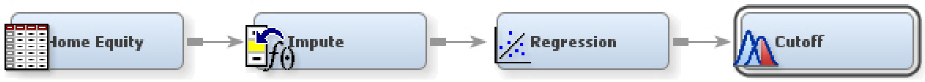
The following example builds a process flow diagram that illustrates the usage of the Cutoff node. For this example, you will use the HMEQ data source in the SAMPSIO library.

1. Open a new Enterprise Miner session and open or create a new project on page 226 .
2. To add the Home Equity data source, right-click the **Data Sources** selection in the Project Navigator and select **Create Data Source**.
   - Select **SAS Table** and press **Next** on the first page.
   - In the Table, type the name `SAMPSIO.HMEQ` or browse to the HMEQ data source in the SAMPSIO library. Press **Next**.
   - Press **Next** on the Table Information page.
   - Select the **Basic Advisor** and press **Next**.

- Change the **Role** for the variable BAD to Target. Change the **Level** for the variable BAD to **Binary**. Press **Next**.
- Select **No**. You will not need a sample of the data set. Press **Next**.
- Verify that the **Role** of the data source is set to **Raw** and press **Next**.
- Review the information in the Summary and select **Finish**.

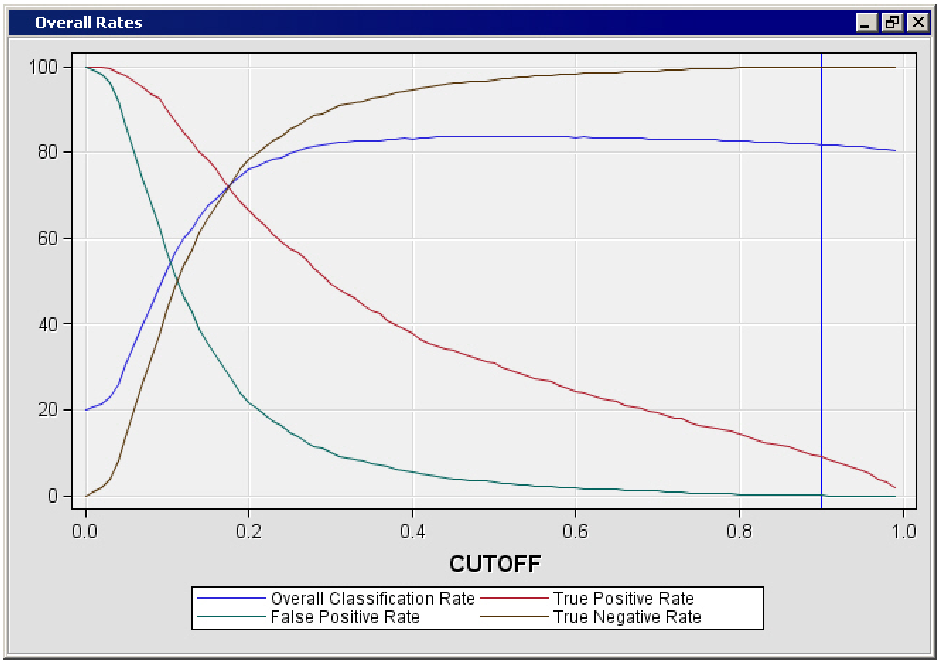
For an in-depth guide to adding a data source, see the Creating a Data Source on page 293 section in the Data Sources Help.

3. To create a new diagram right-click the **Diagrams** selection in the Project Navigator and select **Create Diagram**. Name the diagram `Cutoff Node Example`.
4. Select the Home Equity data source in the Project Navigator and drag it onto the diagram workspace.
5. Click the ellipsis next to the **Decisions** property in the **Columns** submenu of the Properties panel. In the Decisions Processing window, click **Build** to create a decisions matrix. Close the Decision Processing window.
6. The Home Equity data source contains observations with missing values, so you will impute these values using the Impute node. From the **Modify** tab, select the Impute node icon and drag it onto the diagram workspace. Now connect the Home Equity data source to the Impute node. You will run this node with the default settings.
7. Next, you will create a regression model of the data set by dragging a Regression node from the Model tab onto the diagram workspace. This node will run with the default settings
8. Now, drag a Cutoff node from the **Assess** tab onto the diagram workspace. For the first part of this example, you need to change the **Cutoff User Input** property to `0.9` under the Score section of the Properties panel. Your completed process flow diagram should resemble the one shown below.



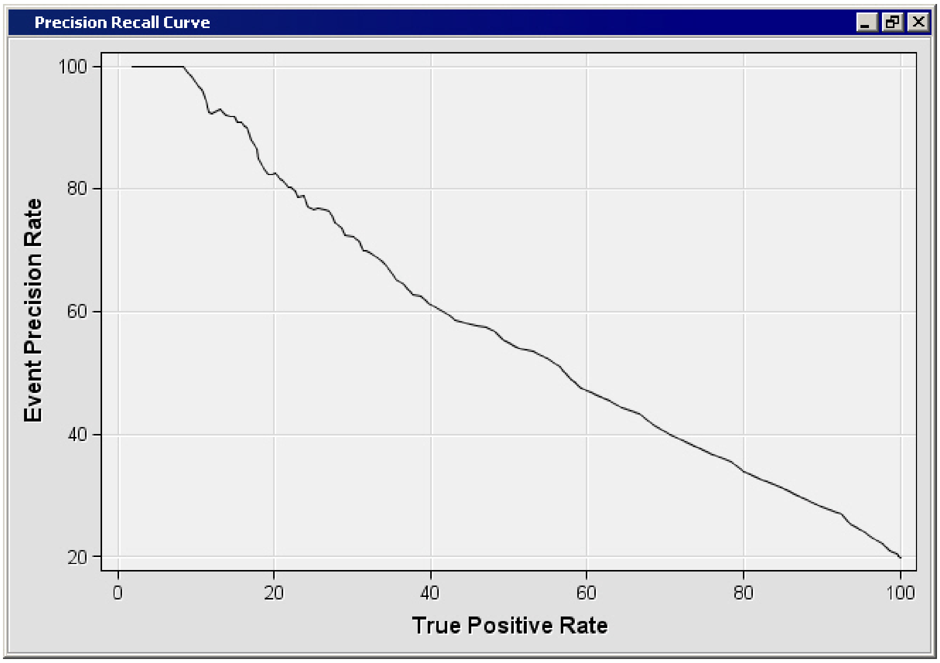
9. Right-click the Cutoff node and select **Run** from the pop-up menu. When all nodes in the process flow diagram run successfully, select the **Results...** button from the Run Status window that appears.
10. In the Results window, make note of the following observations. While these are useful points of information, they are not essential to this example.
    - The Cumulative Expected Profit graph shows a downward sloping line because the cost and revenue information in your decision matrix is fixed.
    - The Receiver Operating Characteristics (ROC) curve shows the relationship between the False Positive Rate and the True Positive Rate. From the graph, you can see that if you are willing to accept about a 40% false positive rate, then you will generate about an 80% true positive rate.
11. The two graphs that you want to focus on are the Overall Rates Curve and the Precision Recall Curve.
    - The Overall Rates Curve shows the measurement levels for different classifications of interest. Because you are trying to make accurate predictions, you are interested in the overall classification, false positive, true positive, and true negative rates. You can observe individual data points by placing the mouse over any of the four curves on the graph.

- Observe that these graphs represent measures of classification, not precision. For example, the true classification rate is defined as the probability that an observation is an event, given that it is an event among all the observations. In the absence of a model, the true classification rate would be the prior rate of events in a study and would be constant for all observations.



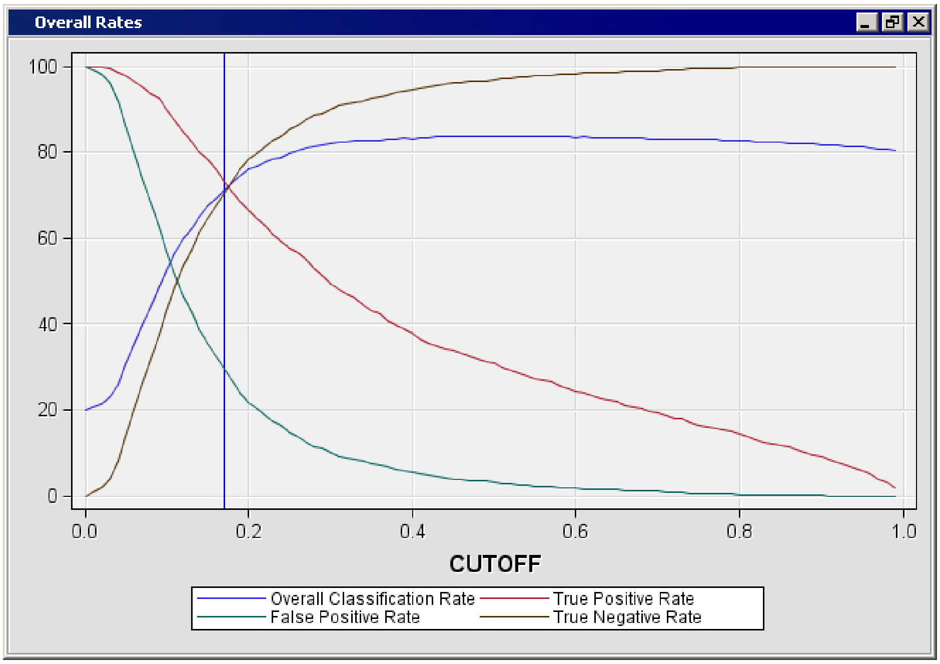
- The Precision Recall Curve shows that there is an inverse relationship between the event precision rate and the true positive rate. You can observe individual data points by placing the mouse over the curve on the graph.

  - The true precision rate is the probability of being an event among all of the observations predicted to be an event and is the Bayesian update of the prior rate given a new observation. Thus, for very high cutoff points, the predicted observations will most likely be predicted and true events.



12. Now, you might have noticed that the overall classification, true positive, and true negative rates all intersect around a cutoff value of 0.17. To do this, you need to change the **Cutoff User Input** property to `0.17` under the Score section of the Properties panel and rerun the node. Once again, you need to view the results.

    - Notice that the cutoff line in the Overall Rates graph has shifted to reflect the cutoff value, but it does not necessarily intersect the three curves at exactly the same point.



    - At this new cutoff value, the Cumulative Expected Profit graph shows a cumulative expected profit of 81.39.

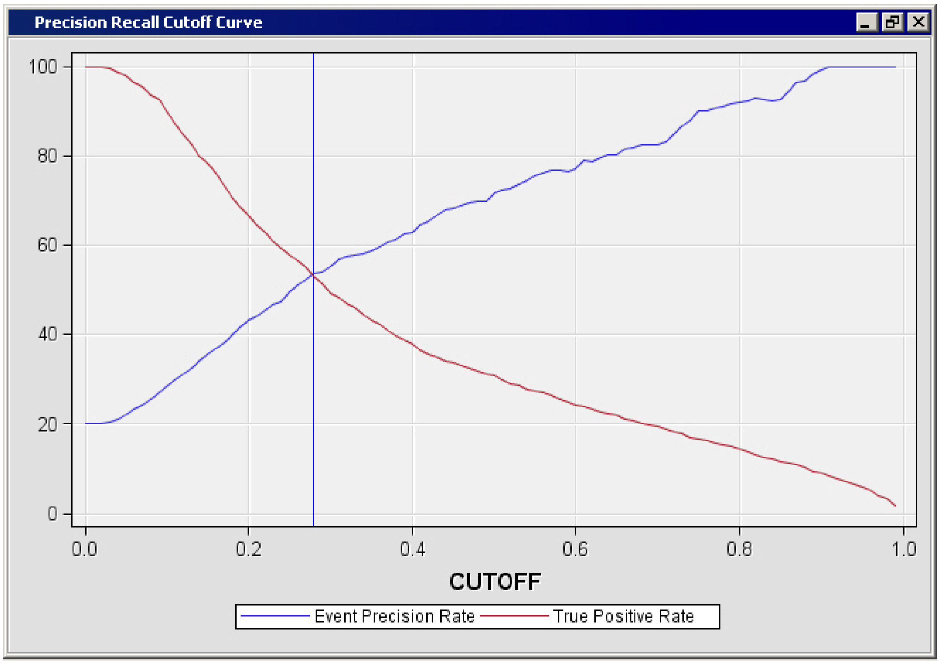
    - If you select **View** ⇨ **Precision Rates** ⇨ **Precision Recall Cutoff Curve**, you will see that the Event Precision Rate is given as 38.235 and the True Precision Rate as 73.255 with a cutoff value of 0.17.

13. For the final step of this example, you want to let Enterprise Miner determine the proper cutoff level by finding the point of intersection in the Event Precision Rate and the True Positive Rate. To accomplish this, close the results window and set the **Cutoff Method** property of the Score section to **Event Precision Equal Recall**. Notice that the **Cutoff User Input** property becomes dimmed when you make this change.

14. Rerun the Cutoff node with the updated settings and view the results once again.

    - Notice that the cutoff value was chosen by the system to be 0.28. If you view the Precision Recall Cutoff Curve, you will see that this is the intersection of the Event Precision Rate and the True Positive Rate.

    - At this cutoff, the Overall Classification Rate is 81.04%, the Event Precision Rate is 52.37%, and the True Positive Rate is 54.92%.



15. You are now done with this example and can close the Results window.

Chapter 66

Decisions Node

Decisions Node

Overview of Decisions Node

The Decisions node belongs to the Assess category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA). You can use the Decisions node to define profiles for a target that produces optimal decisions. The decisions are made using a user-specified decision matrix and output from a subsequent modeling procedure.

Decisions Node Properties

Decisions Node General Properties

The following general properties are associated with the Decisions node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Decisions node added to a diagram has a Node ID of Dec. The second Decisions node added to a diagram has a Node ID of Dec2, and so on.

- **Imported Data** — accesses the Imported Data — Decisions window. The Imported Data — Decisions window contains a list of the ports that provide data sources to the Decisions node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — accesses the Exported Data — Decisions window. The Exported Data — Decisions window lists the output data ports that the Decisions node creates when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Decisions Node Train Properties

The following train properties are associated with the Decisions node:

- **Variables** — Use the Variables property to specify how to use the variables in your data source. Select the ... button to the right of the Variables property to open a variables table. In the table, you can set the Use status for each variable to either Yes (the default) or No, and the Report status of each variable to **No** (the default) or **Yes**.
- **Apply Decisions** — Use the Apply Decisions property to specify whether to apply the changes to the decision matrix or to prior probabilities for the predicted and assessment variables that were generated by the preceding modeling nodes. If a target variable is not modeled, then the node generates a base model for that target variable. The base model uses the proportions observed in the training data. The default setting is **No**.
- **Custom Editor** — Launches the Decisions Editor. Use the Custom Editor property to define a decision matrix for the target variable. Select the ... button to the right of the Custom Editor property to open the Decision Processing window. See “Layout of a Target Profile” on page 208 for more information.
- **Decision** — Use the Decision property to specify the decision information to use.
  - **Custom** — Use the decision information specified in the Decisions Editor or the imported decision matrix.
  - **Property** — Use the decision information specified by the matrix and prior probabilities properties.
  - **None** — Neither a decision matrix nor prior probabilities are used.
- **Matrix** — Specifies which decision matrix to use.

- **Inverse Prior** — The off-diagonal elements of the decision matrix are inversely proportional to the prior probabilities.
- **Inverse Training** — The off-diagonal elements of the decision matrix are inversely proportional to the training probabilities.
- **None** — No decision matrix is used.

- **Prior Probabilities** — Specifies which prior probabilities to use.
  - **Equal Prior** — The prior probabilities are equal for all target levels.
  - **None** — No prior probabilities are used.

Decisions Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Decisions Node Results

After you run the Decisions node, you can open the Results window from the main menu by selecting **Actions** ⇨ **View Results** ⇨ **Results**. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu in the Results — Decisions window to view the following results:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Decisions node properties configuration when the node was last run.
  - **Run Status** — displays the status of the Decision node run. Information about the run start time, run duration, and completion status is displayed in this window.
  - **Variables** — displays a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle between sorting the columns in descending and ascending order by clicking the column headers.
  - **Train Code** — displays the code that Enterprise Miner used to train the node.
  - **Notes** — opens a window and displays (in read-only mode) any notes that were previously entered in the General Properties — Notes window.
- **SAS Results**

- **Log** — the SAS log of the Decisions node run.
- **Output** — the SAS output of the Decisions node run. The Decisions node output has three major sections
  - **Training Output** — This section contains the variable summary, model events, decision matrix, predicted variables, and decision variables.
  - **Score Output** — There is no default score output for the Decisions node.
  - **Report Output** — This section contains the fit statistics, classification table, decision table, event classification table, assessment score ranking, and assessment score distribution.
- **Flow Code** — the SAS code used to produce the output that the Decisions node passes on to the next node in the process flow diagram.

- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Decision node does not generate PMML code.
- **Assessment**
  - **Fit Statistics** — displays the Fit Statistics table.
  - **Classification Chart** — displays a stacked bar chart of the classification results for a categorical target variable. The horizontal axis displays the target levels that observations actually belong to. The color of the stacked bars identifies the target levels that observations are classified into.
  - **Adjusted Classification Chart** — The adjusted classification chart is the same as the classification chart except that the percentage of total observations has been adjusted such that adjusted percentage = unadjusted percentage * (prior/data prior).
  - **Score Rankings Overlay** — In a score rankings chart, several statistics for each decile (group) of observations are plotted on the vertical axis. For a binary target, all observations in the scored data set are sorted by the posterior probabilities of the event level in descending order. For a nominal or ordinal target, observations are sorted from the highest expected profit to the lowest expected profit (or from the lowest expected loss to the highest expected loss). Then the sorted observations are grouped into deciles, and observations in a decile are used to calculate the statistics that are plotted.

    By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations.

    The vertical axis displays the following values including their mean, minimum, and maximum (if any):

    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response

  - **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. For categorical targets, observations are grouped into bins, based on the posterior probabilities of the event level and the number of buckets.

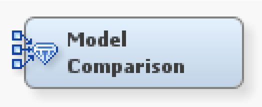
    The Score Distribution chart of a useful model shows a higher percentage of events for higher model scores and a higher percentage of nonevents for lower model scores.

- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Chapter 67

Model Comparison Node

Model Comparison Node



Overview of the Model Comparison Node

The **Model Comparison** node belongs to the Assess category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA). The **Model Comparison** node enables you to compare the performance of competing models using various benchmarking criteria.

There are many criteria that can be used to compare models. Comparing model performance depends on the specific application for the model. For example, with binary targets the **Model Comparison** node provides criteria that are derived from several sources.

- Classification Measures: comparative criteria include the Receiver Operating Characteristic (ROC) charts and corresponding area under the curve, classification rates, and so on.
- Data Mining Measures: comparative criteria from the study of data mining include lift and gain measures and profit and loss measures.
- Statistical Measures: comparative criteria from statistical literature include Bayesian Information Criterion (BIC), Akaike's Information Criterion (AIC), Gini statistics, Kolmogorov-Smirnov statistics, and Bin-Best Two-Way Kolmogorov-Smirnov tests.

The combined criteria enable the analyst to make cross-model comparisons and assessments.

When you train a modeling node, assessment statistics are computed on the train (and validation) data sets. The **Model Comparison** node computes the same statistics for the test set when it is present.

Comparing Models with Binary Targets

Overview

There are several measures that can be used to choose the best model out of a group of several candidate models. The comparative measures are grouped by the type of analysis: statistical, classification, and data mining. The analytical groupings of statistical, classification, and data mining measures appeal to the mixed group of SAS Enterprise Miner users who compare and evaluate statistical models. For example, statisticians might be more familiar with stopping measures such as Mallows' Cq (1973). But analysts might be more comfortable using ROC chart analysis to choose the best model, and direct marketers might prefer using lift and gains tables to benchmark model performance.

Classification Measures

Binary targets are targets whose states can be classified as event and non-event. Typically, event is associated with a value of 1 for the target variable, and non-event is associated with a value of 0 for the target variable. The following table describes the predictive relationships between event and non-event.

	Predicted Non-Event	Predicted Event	Total Actual Probability
Non-Event	A (true negative)	B (false positive)	(A + B) / (A + B + C + D)
Event	C (false negative)	D (true positive)	(C + D) / (A + B + C + D)
Total Predicted	A + C	B + D	A + B + C + D

The following classification measures are derived from the relationships in the table:

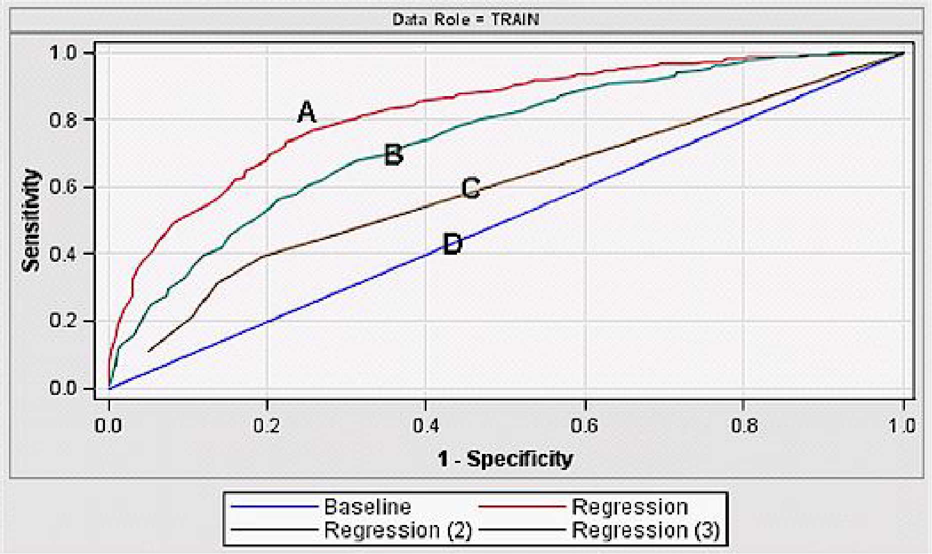
- Classification (Accuracy) Rate: 100 * (A + D) / (A + B + C + D)
- Misclassification Rate: 100 * (1 - ((A + D) / (A + B + C + D)))
- Sensitivity (True Positive Rate): 100 * D / (C + D)
- Specificity (True Negative Rate): 100 * A / (A + B)
- 1 – Specificity (False Positive Rate): 100 * B / (A + B)

These measures are used most often by users who are concerned with model performance over all submitted observations. However, Provost et al. (1998) argued that accuracy measures alone can be misleading. They advocate instead the use of the Receiver Operating Characteristic (ROC) curve.

The SAS Enterprise Miner **Model Comparison** node computes the ROC chart. The ROC chart graphically displays Sensitivity versus 1-Specificity, or the true positive rate versus the false positive rate.

The true positive rate and the false positive rate are both measures that depend on the selected cutoff value of the posterior probability. Therefore, the ROC curve is calculated for all possible cutoff values.

Consider the following chart with four ROC curves, A, B, C, and D:



Each point on curves A, B, C, and D represents cutoff probabilities. Points that are closer to the upper right corner correspond to low cutoff probabilities. Points that are closer to the lower left corner correspond to high cutoff probabilities. The extreme points (1,1) and (0,0) represent rules where all cases are classified into either class 1 (event) or class 0 (non-event). For a given false positive rate (the probability of a non-event that was predicted as an event), the curve indicates the corresponding true positive rate, the probability for an event to be correctly predicted as an event.

Therefore, for a given false positive rate on the 1-Specificity axis, the true positive rate should be as high as possible. The different curves in the chart exhibit various degrees of concavity. The higher the degree of concavity, the better the model is expected to be. In the chart above, A appears to be the best model. Conversely, a poor model of random predictions, such as D, appears as a flat 45-degree line. Curves that push upward and to the left represent better models.

In the above ROC chart, Model A is better than Model B, and Model B is better than Model C. Model D, the 45-degree line, shows no discriminatory power. All possible cutoff points for Model D show that the probability of predicting a non-event as an event is the same as the probability of correctly predicting an event.

In cases where ROC curves overlap, the best model might be selected based on the maximum allowable false positive rate, such as in clinical data applications (Cai and Dodd, 2006). When no false positive rate constraint exists, the preferred model can be selected by choosing the ROC curve that has the greatest area underneath. The area under an ROC curve ranges from a minimal proportion of 50% (random model) to 100%. The higher the value, the more a model is preferred.

The values for the area under an ROC curve can be explained heuristically. For a population with n1 events and n2 non-events, there are n1 x n2 possible pairs. The area under the ROC curve is the proportion of these pairs in which the posterior probability is higher for the event case. Thus, a value of 100 implies that all event cases have higher posterior probabilities than all non-event cases.

Data Mining Measures

The gains chart or lift chart is a data mining measure that is used prominently among marketing professionals to choose among competing models. Assume a model that predicts the probability of some event, such that a higher calculated probability value implies a higher likelihood of an event.

Consider submitting a data set to the model to score the data, and then rank the scored data by descending posterior probability values. In a model with good discriminatory performance, the cases that are predicted to be events should be at the top of the sorted list. The cases that are predicted to be non-events should be at the bottom.

A gains chart table divides the range of the posterior probability into deciles or, for greater detail, into demi-deciles. A model with no predictive power would be expected to predict around 10% of events in each of 10 deciles. A good discriminating model would predict a higher number of events in the top decile, from which the response will decline monotonically.

The following terminology is used for the Gains and Lift Chart:

Mean, min, and max posterior probability

minimum, average, and maximum posterior probability in the bin, decile, or demi-decile.

Decile or Demi-Decile

10% or 5% cuts of the posterior probability, arranged in descending order.

% Response

the proportion of true responders in each decile.

Cumulative % Response

the cumulative proportion of responders. At the last decile, the Cumulative % Response indicates the baseline percentage of responders.

% Captured Response

the proportion of total responders that are captured in a decile or demi-decile.

Cumulative % of Captured Response

the cumulative proportion of total responders that are captured in a decile or demi-decile.

Gain

((% of events in decile / random % events in decile) – 1)

Lift

the ratio of % Captured Response within each decile to the baseline % Response.

Cumulative Lift

the cumulative ratio of % Captured Responses within each decile to the baseline % Response.

Profit

the amount of total profit per decile.

Cumulative Profit

the cumulative amount of total profit per decile.

AUROC (Area Under the ROC Curve)

given the posterior probability for events (X1) and non-events (X0), AUROC = P(X1 > X0). For more information about AUROC computation, see Note on the AUROC Computation section below.

Gini

Somer's D = 2 AUROC – 1.

The baseline is the percentage of events in the population. If you select one case at random, the probability of it being an event is given by the baseline. In each decile, you can measure the percentage of events that are captured by the model. In addition, you can calculate the cumulative percentage of captured events.

Gain shows how much better (percentage wise) the model is performing compared to a random model or no model at all. A Cumulative Lift of one is what you would expect from no model at all. This is why one is subtracted from the Cumulative Lift in the calculation.

Lift is the ratio of the percentage of captured events to the baseline percentage. It shows the lift that the model provides in capturing the desired results (as compared to a 45-degree, straight-line random model). At the 10th decile (100% of the sample), because all responders have been captured, the cumulative percentage of captured events is equal to the baseline. Hence, the lift there is 0.

For information about how to use data mining measures to compare models in SAS Enterprise Miner, see Score Rankings Overlay Charts and Plots on page 255. For information about predictive modeling, see “Predictive Modeling” on page 149 .

Statistical Measures

Statistical measures of model performance are based on both model error and degrees of freedom. Some SAS Enterprise Miner models, such as Decision Tree models, do not output degrees of freedom and are not suitable for benchmarking using the statistical measures listed here. The information that follows pertains to Mallows’ Cq, Akaike’s Information Criterion, Bayesian Information Criterion, and Kolmogorov-Smirnov Statistic and is suitable only for specific models.

Mallow’s Cq

Mallows' Cq (Hosmer and Lemeshow, 2000) is a variant of Mallows Cp measure (1973), which can be used to analyze linear regression models for assessment. Hosmer and Lemeshow derived the corresponding Cq statistic to evaluate logistic regression models, using the following equation

$$C_q = \frac{X^2 + \lambda^*}{X^2 /(n - p - 1)} + 2(q + 1) - n$$

where

- *q* is the number of selected variables,
- *p* is the number of candidate variables
- $$X^2 = \sum \frac{(y_i - \hat{\pi}_i)^2}{\hat{\pi}_i(1 - \hat{\pi}_i)}$$

  is the Pearson chi-square statistic for the model with p variables
- λ is the multivariable Wald statistic that measures significance of p-q coefficients that are not in the model

The expected value of C_q is q + 1. Models with C_q values near q + 1 are candidates for final models.

Akaike’s Information Criterion

The Akaike's Information Criterion (AIC) (Akaike, 1973,1977) is a measure of the goodness of fit of an estimated statistical model. AIC uses the log likelihood function for a model with k parameters to select models, choosing the model that maximizes 2(LL — k) or the model that minimizes –2(LL + k).

Smaller AIC values indicate better models. AIC values can become negative. The log-likelihood term LL increases in value as variables are added to the model, the parameter term -k also increases in magnitude to counteract the log-likelihood term increase, a bias-variance tradeoff. AIC is based on the Kullback-Leibler (1951) information measure of discrepancy between the true distribution of the random variable and that specified by the model.

Bayesian Information Criterion

The Bayesian Information Criterion (BIC) (Schwarz, 1978) also called Schwarz’s Bayesian Information Criterion, is a statistical criterion for model selection. BIC selects models that maximizes the Bayesian log-likelihood expression –2LL + k*ln(n), where

- *n* is the number of observations or the sample size
- *k* is the number of free parameters to be estimated. If the estimated model is a linear regression, k is the number of regression terms, including the constant.
- LL is the maximized value of the log-likelihood function for the estimated model.

If the model errors or disturbances are normally distributed, BIC becomes n*ln(RSS / n) + k ln(n), where RSS is the residual sum of squares from the estimated model.

Given any two models to be compared, the model that has the lower BIC value is the one to be preferred. The BIC is an increasing function of the model's residual sum of squares and an increasing function of k. Unexplained variation in the target variable and the number of explanatory variables increases the value of the BIC. As a result, a lower BIC implies either fewer explanatory variables, better fit, or both. The BIC method penalizes free parameters more strongly that the AIC criterion does.

The BIC criterion can only be used to compare models with identical target variable values. Models being compared do not need to be nested, as is the case with the likelihood ratio test or an F test.

Kolmogorov-Smirnov Statistic

The **Model Comparison** node creates four statistics that are related to the Kolmogorov-Smirnov calculation. Two statistics are based on the ROC curve to create an unbinned Kolmogorov-Smirnov Statistic and two statistics are based on the posterior probability binning to create a binned Kolmogorov-Smirnov Statistic.

The unbinned Kolmogorov-Smirnov statistics are as follows:

- KS — The maximum vertical separation between the sensitivity and the baseline on the ROC curve. This is equal to one minus the specificity.
- _KS_PROB_CUTOFF — The cutoff probability for the sensitivity and specificity values associated with KS.

The binned Kolmogorov-Smirnov statistics are as follows:

- _KS_bin_ — The maximum difference between the cumulative percentage captured response and the cumulative percentage captured non-response among the bins.
- BINNED_KS_PROB_CUTOFF — The mean of the minimum posterior probability and the maximum posterior probability for the bin that is associated with _KS_bin_.

To compute _KS_bin_, assume you have bins B_1, B_2, ..., B_n sorted in descending order of posterior probabilities. The cumulative percentage captured response for the k^{th} bin, CPR_k, is defined as the cumulative number of events in the first k bins divided by the

total number of events. The cumulative percentage captured non-response for the k^{th} bin, $CPNR_k$ is defined as the cumulative number of nonevents in the first k bins divided by the total number of nonevents. The value of _KS_bin_ is given by
$\_KS\_bin\_ = \max(CPR_k - CPNR_k),\ k = 1,\ 2,\ \ldots,\ n.$

Because the first two statistics do not depend on the number of bins and the second two statistics do, the variables KS and _KS_bin_ evaluate the maximum Kolmogorov-Smirnov separation from different perspectives. The unbinned variables segment the observations by the probability cutoffs for predicting an observation as an event or nonevent. The binned variables segment the observations by the posterior probabilities. For a moderate sample size and all but the worst models, the values of KS and _KS_bin_ should be close.

Note on Computational Differences between SAS Enterprise Miner 7.1 and Its Predecessors

There are computational differences (detailed below) between SAS Enterprise Miner 7.1 and its predecessors. If you want to use the earlier version, add the following to the start code:

```
%LET EM_ASSESS = EM53;
```

SAS Enterprise Miner 7.1 and later versions contain a more efficient algorithm for computing model assessment measures (the ranking method). As a result, there might be a marginal difference in reported values in the case of large numbers of tied observations.

This version solves the issue of tied probabilities and it precisely allocates observations to corresponding bins. Tied probabilities emerge in modeling methods such as Classification trees.

Note on AUROC Computation

AUROC is estimated in any of the following ways:

- Non-parametrically using the trapezoidal rule (used in SAS Enterprise Miner 5.3 and predecessors):

$$AUC(\theta) \approx \sum_{i=2}^{m} \Delta FPR\ TPR(\theta_i) + \frac{1}{2}\Delta TPR\ \Delta FPR$$
$$\Delta TPR = TPR(\theta_i) - TPR(\theta_{i-1})$$
$$\Delta FPR = FPR(\theta_i) - FPR(\theta_{i-1})$$

- Ranking Method (Hand and Till, 2001, used in SAS Enterprise Miner 7.1 and beyond):

  $(S — n_1*(n_1 + 1) / 2) / (n_0*n_1)$ where

  - $S = \sum r_{i,1}$
  - $r_{i,1}$ = rank of i^{th} event observation after all target observations have been sorted ascending by the posterior probability of target observations
  - n_1 = number of events
  - n_0 = number of non-events

Note: SAS Enterprise Miner uses the second formula to estimate AUROC because it is easier to compute and to generalize to the multi-class problem.

Definition of Profit and Loss Measurements

We present below the formulas that define many of the profit/loss variables presented in reports for either binary or interval targets. As a general concept, "base" variables refer to "baseline" measures, that is, measures that refer to the average of the corresponding target variable. "Best" refers to the best measure within a specified context. For example, Total Best profit denotes the maximum best profit achieved within a bin multiplied by the number of observations in the bin.

It is also important to consider the difference between cumulative and non-cumulative. The non-cumulative ones, be they either averages or totals, refer to measures for the decile or bin itself. The cumulative ones, especially the averages, accumulate profits or losses, but they also divide by the cumulative number of observations up to the decile in question.

Here are some measure definitions and examples with loss and profit measures:

- TOTALBESTPROFIT — Best Decile Profit = Highest Decile Computed profit * Number of Observations in the decile
- CUMULATIVETOTALBESTPROFIT — Cumulative of Total Best Profit
- AVGCUMULATIVEBESTPROFIT — Average Cumulative Best Profit = CumulativeTotalBestProfit / Cumulative Number of observations so far
- AVGBASEPROFIT — Baseline Average Profit = Average of overall computed profit
- AVGCUMULATIVEPROFIT — Cumulative Decile Computed Profit = Aggregation of computed profit
- BASECUMULATIVEPROFIT — Baseline Average Cumulative Profit
- BASECUMULATIVETOTALPROFIT — Baseline Cumulative Decile Profit
- BESTCUMULATIVEPROFIT — Best Cumulative Profit = Best cumulative profit / cumulative number of observations
- BASETOTALPROFIT — Baseline Decile Profit = Average baseline profit * number of Observations in the decile

Model Comparison Node Properties

Model Comparison Node General Properties

The following general properties are associated with the Model Comparison Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Model Comparison** node that is added to a diagram will have a Node ID of MdlComp. The second Model Comparison node added to a diagram will have a Node ID of MdlComp2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Model Comparison window. The Imported Data — Model Comparison window contains a list of the ports that provide data sources to the **Model Comparison** node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Model Comparison window. The Exported Data — Model Comparison window contains a list of the output data ports for which the **Model Comparison** node creates data when it runs. Click the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Model Comparison Node Train Properties

The following train properties are associated with the **Model Comparison** node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **Model Comparison** node. Click the ... button to open a window containing the Variables table. You can set the Use property to Default, No, or Yes in the table, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a distribution plot.

Model Comparison Node Train Properties: Assessment Reports Properties

- **Number of Bins** — Use the Number of Bins property of the **Model Comparison** node to specify the number of bins that you want to use for the score rankings data set. Permissible values are 5, 10, 20, 25, 50, and 100. The default value for the Number of Bins property is 20.
- **ROC Chart** — When set to No, the ROC Chart property of the **Model Comparison** node suppresses the creation of the Receiver Operation Characteristic (ROC) chart. ROC charts require a binary target. The default setting for the ROC Chart property is **Yes**.
- **Recompute** — Specifies to recompute assessment reports and fit statistics for the incoming data sets.

Model Comparison Node Train Properties: Model Selection Properties

- **Selection Data** — Specifies the source of the fit statistics that are used to select the champion model.

When set to **Default**, models that exclusively use **High-Performance** nodes generate fit statistics that are based on the grid data. All other models will generate fit statistics that are based on the sample that SAS Enterprise Miner automatically downloads when the Data Source is created. (You can control the size of the sample in Step 7 of the Data Source Wizard.)

When set to **Sample**, the fit statistics that are based on the automatic SAS Enterprise Miner Data Source sample data are used.

When set to **Grid**, the fit statistics that are based on the training data are used only for the incoming **High-Performance** nodes. If no **High-Performance** nodes are used to build the candidate models, then the fit statistics based on the automatic SAS Enterprise Miner Data Source sample are used.

Note: When **High-Performance** nodes are used, you can control the size of the automatic sample that SAS Enterprise Miner creates in Step 7 of the Data Source Wizard. However, SAS Enterprise Miner will always create the automatic sample, even if it is not specified. If the Data Source sample is specified in Step 7 of the Data Source Wizard, then the sample will be the size that you specify in the wizard. Otherwise, a 100,000 row sample is created. If one or more class targets are defined, and/or one or more segment variables are defined, then the sample will be stratified by those variables, up to a limit of two stratification variables.

- **Selection Statistic** — Use the Selection Statistic property of the **Model Comparison** node to specify the fit statistic that you want to use to select the model. Depending on the availability, statistics from test, validation, or training data set (in that order) will be used. If you choose a selection statistic that is not appropriate for the target variable type (such as choosing Mean Square Error for a binary target), SAS Enterprise Miner ignores the user-specified selection statistic. Instead, SAS Enterprise Miner chooses and reports on the default Selection Statistic for the target variable type.

  The selections **Average Squared Error** and **Mean Squared Error** might appear similar. But they describe two completely different measures, where each is appropriate only for specific models. In linear models, statisticians routinely use the mean squared error (MSE) as the main measure of fit. The MSE is the sum of squared errors (SSE) divided by the degrees of freedom for error. (DFE is the number of cases less the number of weights in the model.) This process yields an unbiased estimate of the population noise variance under the usual assumptions.

  For neural networks and decision trees, there is no known unbiased estimator. Furthermore, the DFE is often negative for neural networks. There exist approximations for the effective degrees of freedom, but these are often prohibitively expensive and are based on assumptions that might not hold. Hence, the MSE is not nearly as useful for neural networks as it is for linear models. One common solution is to divide the SSE by the number of cases N, not the DFE. This quantity, SSE/N, is referred to as the average squared error (ASE).

  The MSE is not a useful estimator of the generalization error, even in linear models, unless the number of cases is much larger than the number of weights. Another estimator, the final prediction error (FPE), is unbiased under certain assumptions in linear models. The FPE is usually more useful than the MSE for data mining, in either linear models or neural networks.

  The Selection Statistic choices are as follows:

  - **Default** — The default selection uses different statistics based on the type of target variable and whether a profit/loss matrix has been defined.

        - If a profit/loss matrix is defined for a categorical target, the average profit or average loss is used.
        - If no profit/loss matrix is defined for a categorical target, the misclassification rate is used.
        - If the target variable is interval, the average squared error is used.
    - **Akaike's Information Criterion** — chooses the model with the smallest Akaike's Information Criterion value.
    - **Average Squared Error** — chooses the model with the smallest average squared error value.
    - **Mean Squared Error** — chooses the model with the smallest mean squared error value.
    - **ROC** — chooses the model with the greatest area under the ROC curve.
    - **Captured Response** — chooses the model with the greatest captured response values using the decile range that is specified in the Selection Depth property.
    - **Gain** — chooses the model with the greatest gain using the decile range that is specified in the Selection Depth property.
    - **Gini Coefficient** — chooses the model with the highest Gini coefficient value.
    - **Kolmogorov-Smirnov Statistic** — chooses the model with the highest Kolmogorov-Smirnov statistic value.
    - **Lift** — chooses the model with the greatest lift using the decile range that is specified in the Selection Depth property.
    - **Misclassification Rate** — chooses the model with the lowest misclassification rate.
    - **Average Profit/Loss** — chooses the model with the greatest average profit/loss.
    - **Percent Response** — chooses the model with the greatest % response.
    - **Cumulative Captured Response** — chooses the model with the greatest cumulative % captured response.
    - **Cumulative Lift** — chooses the model with the greatest cumulative lift.
    - **Cumulative Percent Response** — chooses the model with the greatest cumulative % response.
- **Grid Selection Statistic** — Specifies the fit statistics that is used to determine a champion model for data that exists on the grid. The selection statistics available here are a subset of the selection statistics available in the **Selection Statistic** property. Note that if you specify **Event Rate**, **Error Rate**, **Lift**, **Cumulative Lift**, or **Separation — KS** then you must specify a value for the **Selection Depth** property.
- **Selection Table** — Use the Selection Table property of the **Model Comparison** node to specify which partitioned data table you that want to use to select the best mode. The Selection Table property is dimmed and unavailable if the Selection Statistic property is set to Default. The available choices for the Selection Table property are
    - **Train** — Use the Train data set table.
    - **Validation** — Use the Validation data set table
    - **Test** — Use the Test data set table.

- **Selection Depth** — Use the Selection Depth property of the **Model Comparison** node to specify the decile range that you want to use when using the ROC, captured response, lift, and gain criteria. The Selection Depth property offers the demi-decile choices of 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50.

Model Comparison Node Score Properties

The following score property is associated with the **Model Comparison** node:

- **Selection Editor** — launches an editor that enables you to manually select the model that is exported to successor nodes in the process flow diagram. The node will automatically select the model based on the model selection properties. However, you can override that selection using this property. Click the ... button to the right of the Selection Editor property to open a Selection Editor window.

Model Comparison Node Report Properties

The following report properties are associated with the **Model Comparison** node:

- **Target** — displays the name of the target variable.
- **Model Node** — displays the node ID of the selected model.
- **Model Description** — displays the node description of the selected model.
- **Selection Criteria** — displays the criterion used to select the model. It is set to Manual Selection if the model was selected by using the Selection Editor.

Model Comparison Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Model Comparison Node Results Window

You can open the Results window of the **Model Comparison** node after a successful run by clicking the **Results** button in the Run Status window. You can also open the window by right-clicking the node in the Diagram Workspace and clicking Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

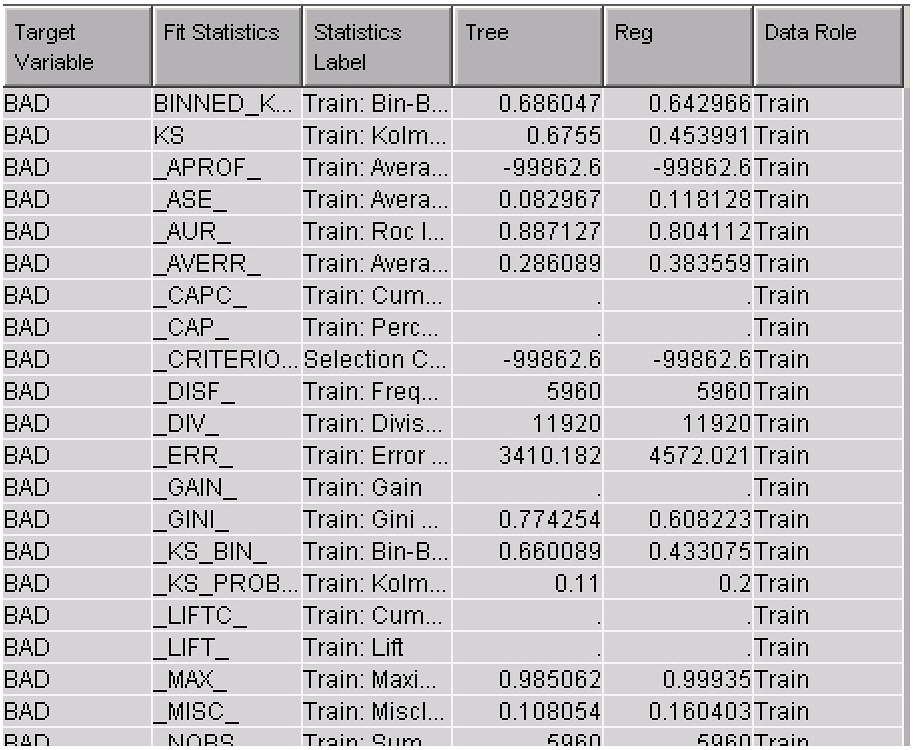
- **Properties**

  - **Settings** — displays a window with a read-only table of the configuration information in the Model Comparison node Properties panel. The information was captured during the last node run.
  - **Run Status** — indicates the status of the Model Comparison node run. Information about run start time, run duration, and whether the run completed successfully is displayed in this window.
  - **Variables** — displays a read-only table of variable metadata for the data set submitted to the **Model Comparison** node. The table includes columns for variable name, use, role, level, event, and label.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables you to read or create notes of interest.
- **SAS Results**
  - **Log** — displays the SAS log of the Model Comparison node run.
  - **Output** — displays the SAS output of the Model Comparison node run. Typical output includes information such as a variable summary; fit statistics tables for train, validation, and test data; and event classification tables.
  - **Flow Code** — displays the SAS flow code for the **Model Comparison** node.
- **Scoring**
  - **SAS Code** — The SAS score code can be used outside the SAS Enterprise Miner environment in custom user applications. Score code is generated to assign a new observation to a quantile. Use this code to select observations within a quantile range—for example, all observations predicted to be in the top 20% of all candidates.

    ```
    if (P_BAD1 ge 0.28845386514426) then do ;
    b_BAD = 1;
    end;
    else
    if (P_BAD1 ge 0.15793841246033) then do;
    b_BAD = 2;
    end;
    else
    if (P_BAD1 ge 0.09848007304609) then do;
    b_BAD = 3;
    end;
    else
    if (P_BAD1 ge 0.06148921253496) then do;
    b_BAD = 4;
    end;
    else
    do;
    b_BAD = 5;
    end;
    ```

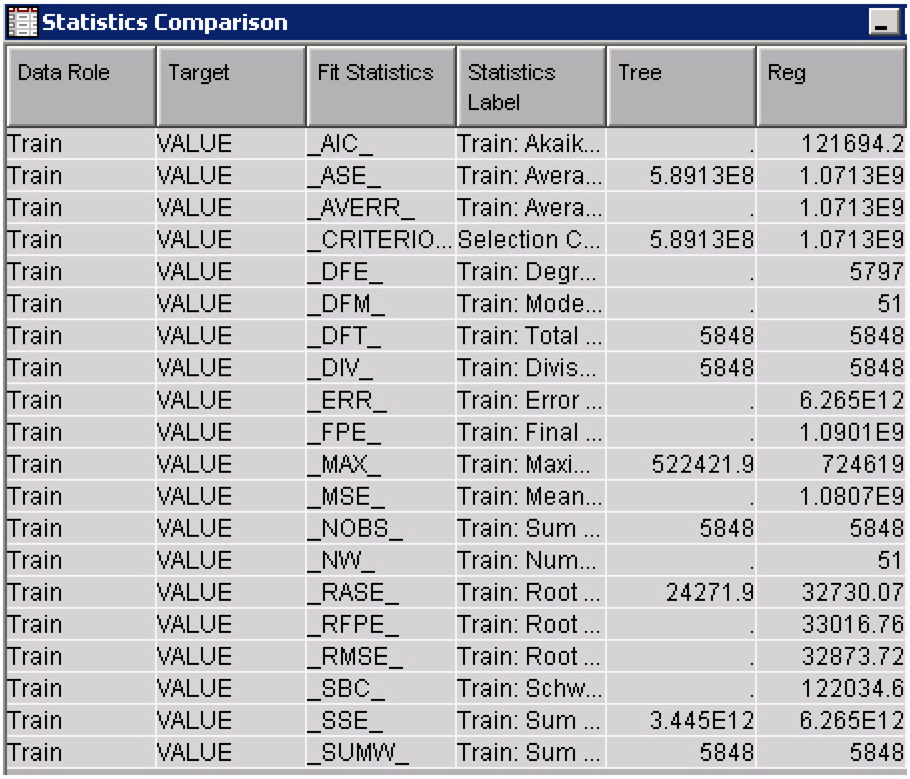
  - **PMML Code** — the **Model Comparison** node does not generate PMML code.
- **Model**
  - **Fit Statistics** — displays the Fit Statistics window.
  - **Statistics Comparison** — displays the Statistics Comparison window.

    Example of a Statistics Comparison table for a binary target:



Target Variable	Fit Statistics	Statistics Label	Tree	Reg	Data Role
BAD	BINNED_K...	Train: Bin-B...	0.686047	0.642966	Train
BAD	KS	Train: Kolm...	0.6755	0.453991	Train
BAD	_APROF_	Train: Avera...	-99862.6	-99862.6	Train
BAD	_ASE_	Train: Avera...	0.082967	0.118128	Train
BAD	_AUR_	Train: Roc I...	0.887127	0.804112	Train
BAD	_AVERR_	Train: Avera...	0.286089	0.383559	Train
BAD	_CAPC_	Train: Cum...	.	.	Train
BAD	_CAP_	Train: Perc...	.	.	Train
BAD	_CRITERIO...	Selection C...	-99862.6	-99862.6	Train
BAD	_DISF_	Train: Freq...	5960	5960	Train
BAD	_DIV_	Train: Divis...	11920	11920	Train
BAD	_ERR_	Train: Error ...	3410.182	4572.021	Train
BAD	_GAIN_	Train: Gain	.	.	Train
BAD	_GINI_	Train: Gini ...	0.774254	0.608223	Train
BAD	_KS_BIN_	Train: Bin-B...	0.660089	0.433075	Train
BAD	_KS_PROB...	Train: Kolm...	0.11	0.2	Train
BAD	_LIFTC_	Train: Cum...	.	.	Train
BAD	_LIFT_	Train: Lift	.	.	Train
BAD	_MAX_	Train: Maxi...	0.985062	0.99935	Train
BAD	_MISC_	Train: Miscl...	0.108054	0.160403	Train
BAD	_NOBS_	Train: Sum	5960	5960	Train

Example of a Statistics Comparison table for an interval target:

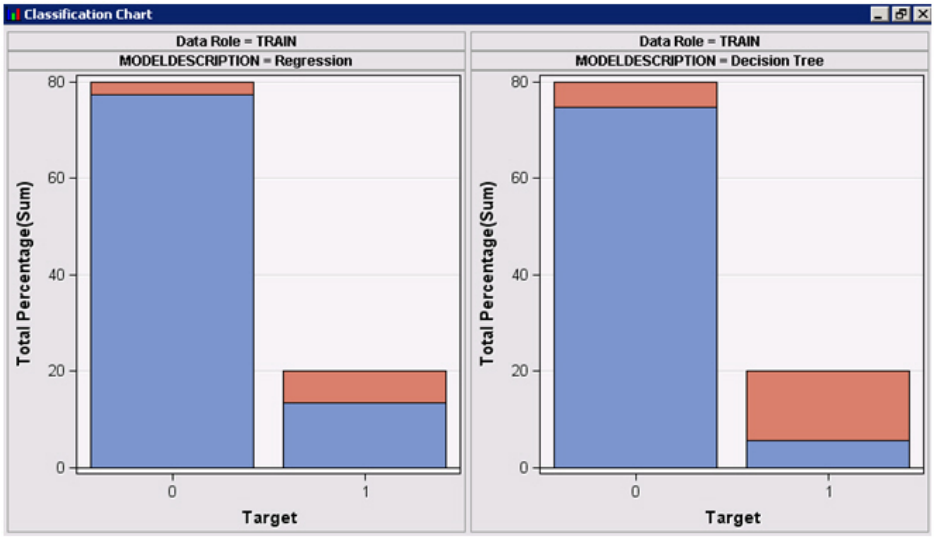


Statistics Comparison

Data Role	Target	Fit Statistics	Statistics Label	Tree	Reg
Train	VALUE	_AIC_	Train: Akaik...	.	121694.2
Train	VALUE	_ASE_	Train: Avera...	5.8913E8	1.0713E9
Train	VALUE	_AVERR_	Train: Avera...	.	1.0713E9
Train	VALUE	_CRITERIO...	Selection C...	5.8913E8	1.0713E9
Train	VALUE	_DFE_	Train: Degr...	.	5797
Train	VALUE	_DFM_	Train: Mode...	.	51
Train	VALUE	_DFT_	Train: Total ...	5848	5848
Train	VALUE	_DIV_	Train: Divis...	5848	5848
Train	VALUE	_ERR_	Train: Error ...	.	6.265E12
Train	VALUE	_FPE_	Train: Final ...	.	1.0901E9
Train	VALUE	_MAX_	Train: Maxi...	522421.9	724619
Train	VALUE	_MSE_	Train: Mean...	.	1.0807E9
Train	VALUE	_NOBS_	Train: Sum ...	5848	5848
Train	VALUE	_NW_	Train: Num...	.	51
Train	VALUE	_RASE_	Train: Root ...	24271.9	32730.07
Train	VALUE	_RFPE_	Train: Root ...	.	33016.76
Train	VALUE	_RMSE_	Train: Root ...	.	32873.72
Train	VALUE	_SBC_	Train: Schw...	.	122034.6
Train	VALUE	_SSE_	Train: Sum ...	3.445E12	6.265E12
Train	VALUE	_SUMW_	Train: Sum ...	5848	5848

- **Assessment**
  - **Classification chart** — graphs the model predictions and the true state of nature. When you hold your mouse pointer over the bars, the percentages of observations are displayed.

Example of a Classification chart for a binary target:



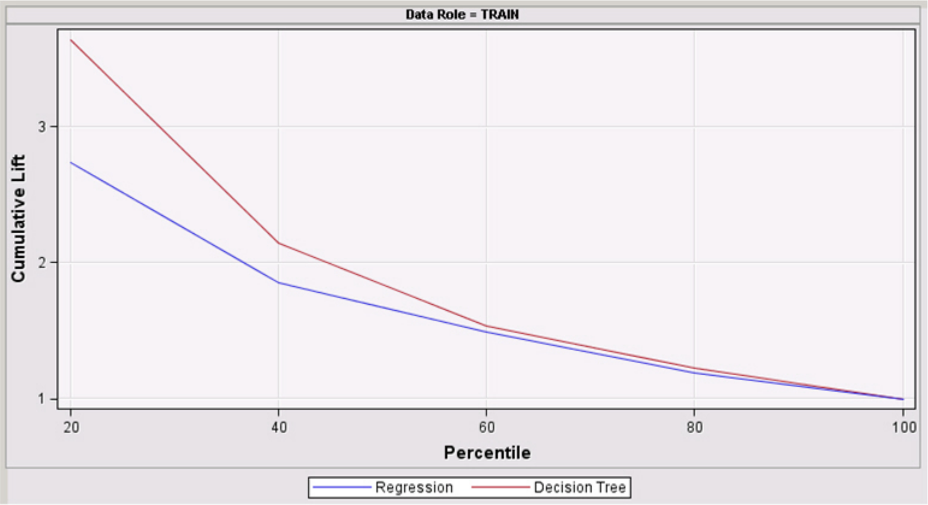
In this example, the logistic regression model performs better than the tree model in terms of classifying True Negatives and trees in terms of classifying true positives. The corresponding "Area under the Curve statistics", .80 and .89 for logistic and tree models respectively, indicate that the tree model outperforms the logistic regression in this case.

There is no corresponding Classification Chart for the case of an interval target.

- **Score Rankings Overlay** — displays the score rankings plots for the competing models overlaid on a single chart. The overlay score ranking plot displays assessment statistics for the competing models across the specified deciles. The score rankings overlay plot has a drop-down list that you use to choose the assessment statistic that is displayed on the y-axis. The available assessment statistics for the score rankings overlay chart are

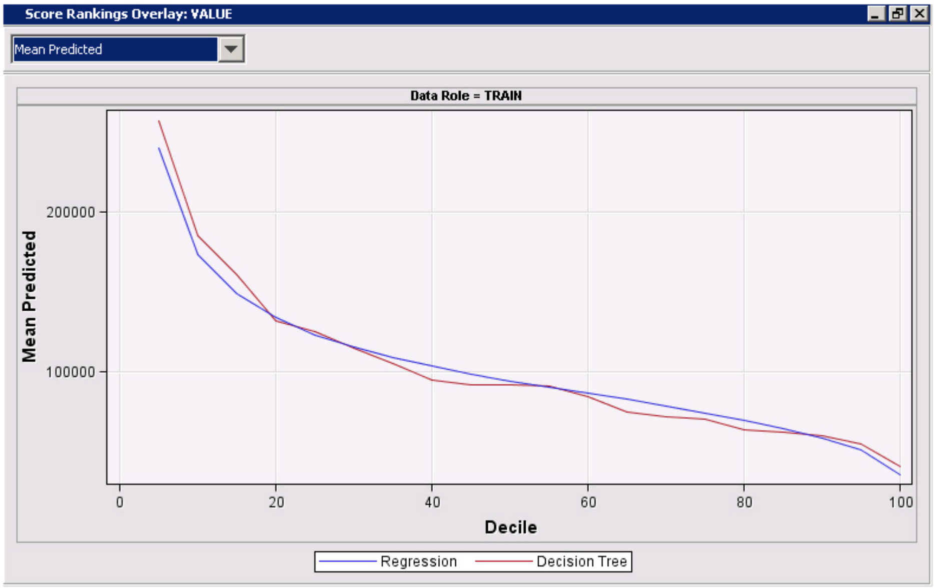
  - cumulative lift
  - lift
  - gain
  - % response
  - cumulative % response
  - % captured response
  - cumulative % captured response

  Example of a Score Ranking Overlay chart for a model with a binary target:



The lift plot confirms the previous inference via the area under the curve that the decision tree model outperforms the logistic regression. In this case, at any percentile level, the corresponding lift obtained by the tree model is superior to that of the logistic regression.

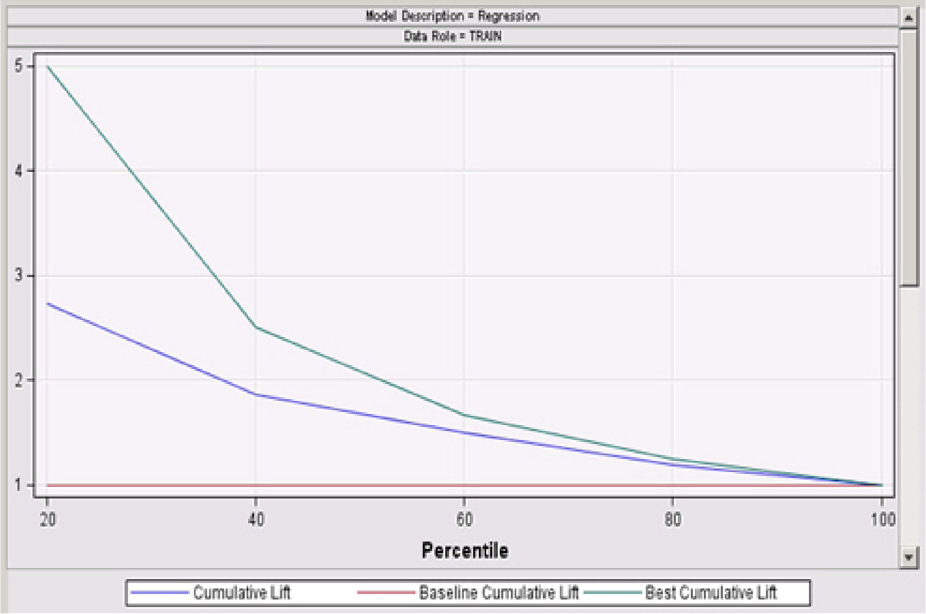
Example of a Score Rankings Overlay chart for a model with an interval target:



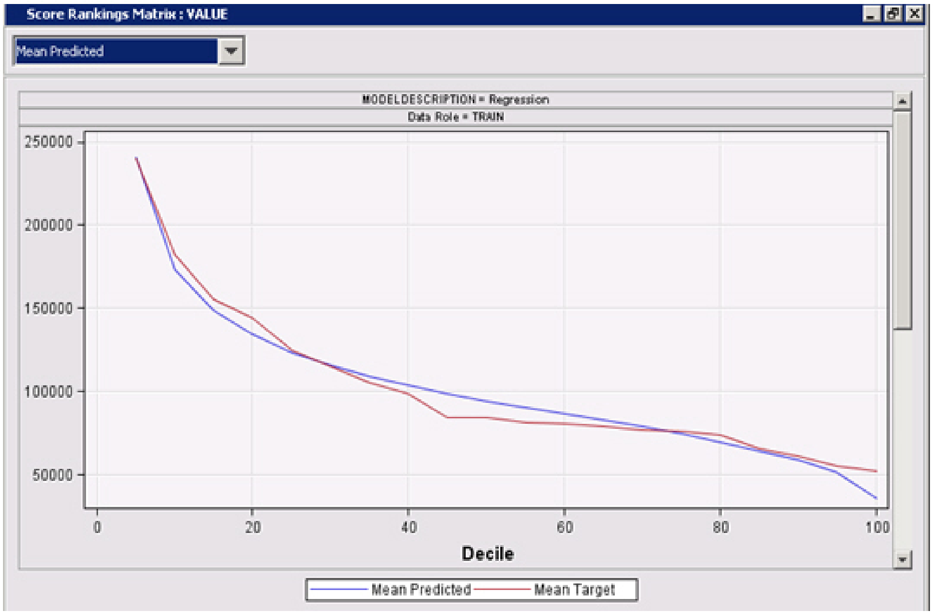
- **Score Rankings Matrix Plot** — displays a matrix of score ranking plots. Instead of multiple plots overlaid on a single chart, the matrix plot displays individual assessment statistic plots for each model and for each data role (train, validate, test) that is included in the model comparison analysis. You can use the score rankings matrix plot to compare the different models across the same data role. Or you can look for changes in one model's performance across train, validate, and test data roles. The score rankings matrix plot has a drop-down list that you use to choose the assessment statistic that is displayed on the y-axis. The available assessment statistics for the score rankings matrix plot are
  - cumulative lift
  - lift
  - gain

- % response
- cumulative % response
- % captured response
- cumulative % captured response

Example of a matrix plot for a model that has a binary target:

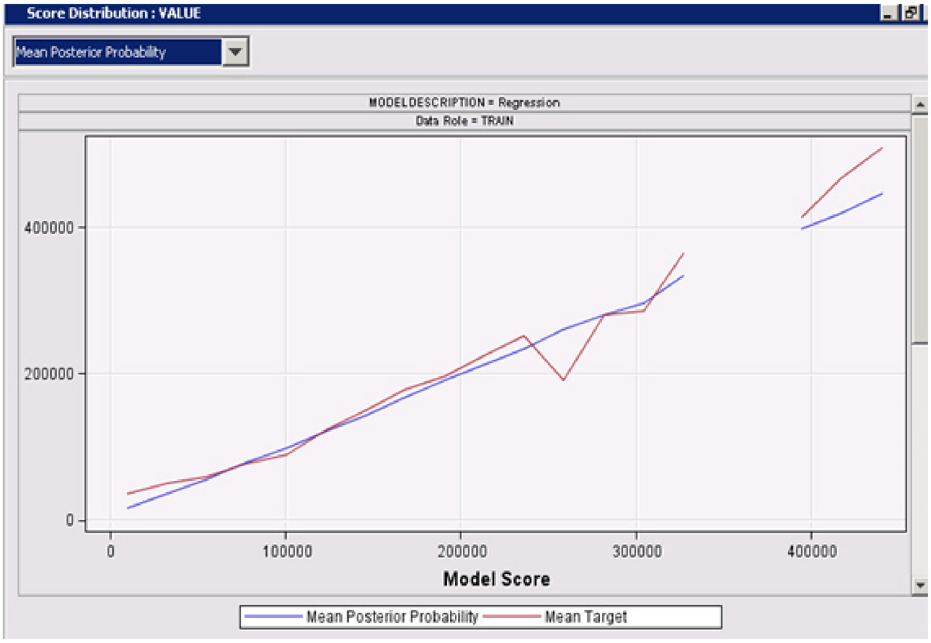


Example of a matrix plot for a model that has an interval target:

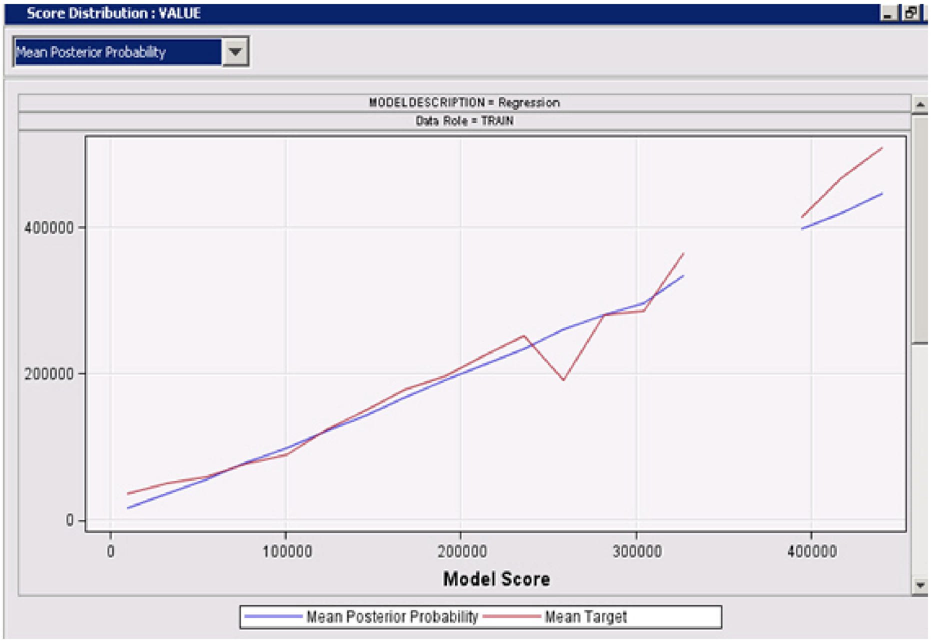


- **Score Distribution** — displays score distribution charts for each competitor model in each of the data roles (train, validate, test) that are included in the comparative analysis. The Score Distribution chart plots the proportions of events and nonevents by decile for each competitor model and data role. The Score Distribution chart has a drop-down list that you use to choose the score distribution that you want to display on the y-axis. The available distributions are Percentage of Events and Cumulative Percentage of Events.

Example score distribution chart for a model that has a binary target:

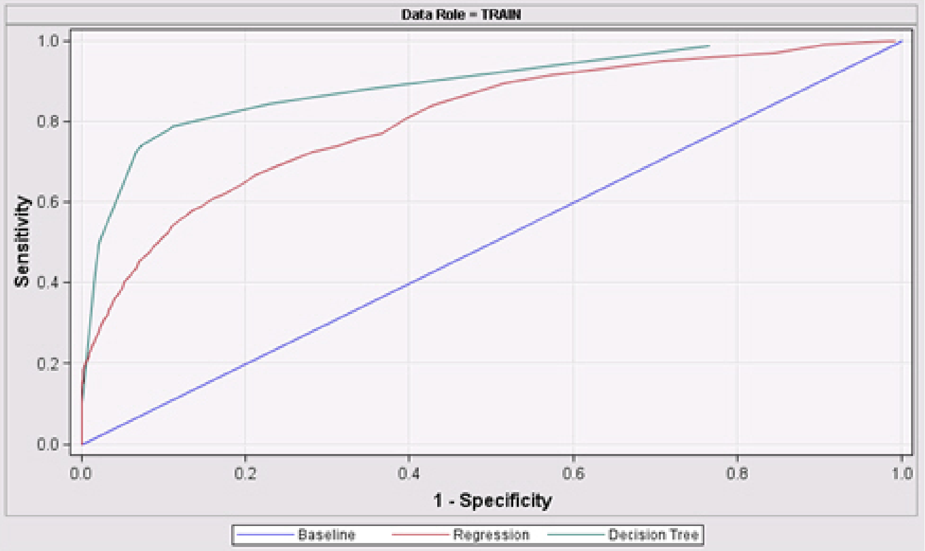

Score Distribution : VALUE
Mean Posterior Probability
MODELDESCRIPTION = Regression
Data Role = TRAIN
400000
200000
0
0
100000
200000
300000
400000
Model Score
Mean Posterior Probability
Mean Target

Example score distribution chart for a model that has an interval target:


Score Distribution : VALUE
Mean Posterior Probability
MODELDESCRIPTION = Regression
Data Role = TRAIN
400000
200000
0
0
100000
200000
300000
400000
Model Score
Mean Posterior Probability
Mean Target

- **ROC Chart** — displays multiple ROC curves for competing models on a single chart. The ROC chart is not available for models that do not have binary targets. The **ROC Chart** menu item is dimmed and unavailable when the models to be compared have ordinal or categorical targets.

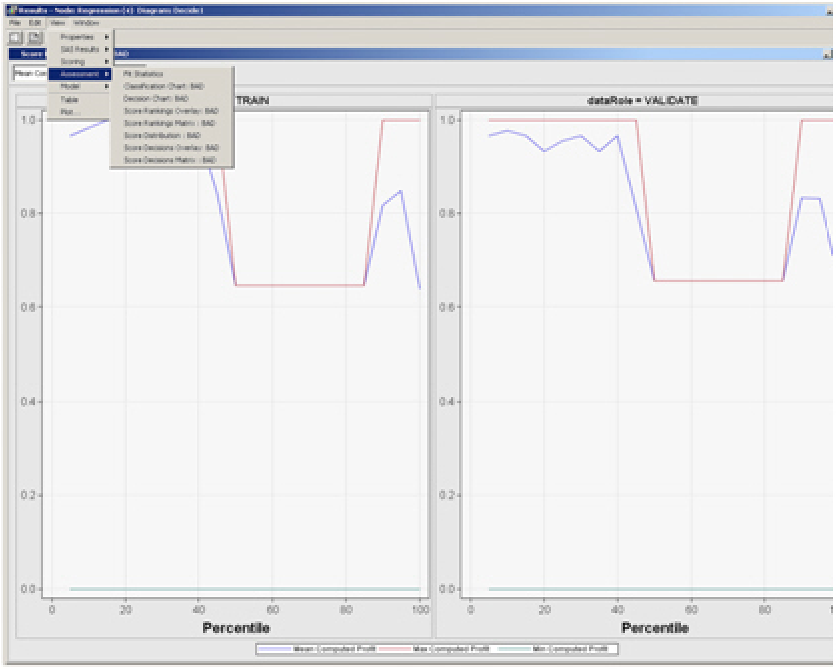
  Example ROC chart of a model that has a binary target:



There is no ROC chart for an interval target model run.

When profit or loss measurement information is provided, the following graphs and tables are presented:

- **Score Decisions Overlay** — presents a series of graphs for loss and profit. For example, the following is the corresponding mean computed profit for a binary target:



Profit and loss measurements also appear in other SAS Enterprise Miner tables and graphs. The distinctive aspect of this tab is that the sorting order of the percentile is the expected loss and profit created by the model. The models typically maximize the probability of the event and not expected loss or profit. In this sense, the graphs do not necessarily show a monotonic relationship because the expected measure is the product of the probability times the actual profit.

The corresponding graph for a continuous target is similar and is therefore omitted.

- **Score Decisions matrix** — this is the corresponding table that generates the profit/loss/ROI graphs detailed above.

- **Table** — displays the data table that was used to produce the selected chart in the Model Comparison Results window. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Select a Chart Type window, which you use to create ad hoc plots that are based on the underlying data used to produce the table that is selected in the Results window. The **Plot** menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Note: The new assessment computation improves the previous version in the following ways:

- Tied probabilities have been improved to measure more accurately the number of observations falling in each decile. Tied probabilities occur frequently in the output of tree methods.
- The processing is faster and more accurate.
- In the reports, if there is no information for a specific decile or group probability bin, missing values are reported instead of carrying on information from a previous bin. For cumulative measures however, information is accumulated.

Note: When the same model is compared across training, validation, and test data sets through the gains table for example, the limits of each decile or demi-decile will be different according to the variation in the distribution of the data itself. In general, it is expected that there should not be much variation from one data set to the next. But variation is certainly possible, which would indicate that the analyst should review the partitioning process and the data itself for differences in the structure of the overall data.

References

Akaike, H “Maximum Likelihood Identification of Gaussian Autoregressive Moving Average Models.” 1973. *Biometrika* 60 (2): 255–265.

Akaike, H. 1977. *On the Entropy Maximization Principle, Applications of Statistics*. Amsterdam, Holland: , .

Cai, T., and Dodd, L 2006. “Regression Analysis for the Partial Area Under the ROC Curve.” Working Paper 36, Harvard University Biostatistics Working Paper Series.

Hand, D. and Till, R “A Simple Generalization of the Area Under the Curve for Multiple Class Classification Problems.” 2001. *Machine Learning* 45: 171–186.

Kullback, S. and Leibler, R. “On Information and Sufficiency.” 1951. *Annals of Mathematical Statistics* 22: 79–86.

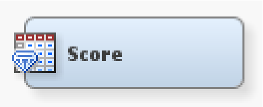
Mallows, C “Some Comments on Cp.” 1973. *Technometrics* 42: 87–94.

Provost, F., Fawcett, T., and Kohavi, R 1998. “The Case Against Accuracy Estimation for Comparing Induction Algorithms.” *Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, 445–453.

Chapter 68

Score Node

Score Node



Overview of the Score Node

Use the Score node to manage SAS scoring code that is generated from a trained model or models, to save the SAS scoring code to a location on your client computer, and to run the SAS scoring code. You can score a data set to generate predicted values for a data set that might not contain a target. You can also score a data set to create a segment variable or to impute missing values. The SAS scoring code can be converted into C and Java scoring code.

The Score node creates variables that have fixed names. These fixed name variables correspond to the variables that have system generated names, which depend on the target name and the target levels. For example, if you have the target BAD with the event level 1, the posterior variables generated by Enterprise Miner will be P_BAD1. To use these system generated names could be troublesome in some applications. In this case, a fixed name variable might be EM_EVENTPROBABILITY, which could be used for further analysis.

The Score node generates and manages scoring formulas that usually are, but not restricted to, the form of a single DATA step, which can be used in most SAS environments, even without the presence of Enterprise Miner.

An Enterprise Miner process flow diagram can contain divergent and convergent paths. Most nodes recognize and operate on a data set type of TRAIN. Score code is accumulated in the process flow based on the path traced by the training data set. Operations on only the validation or test data sets are not recognized as part of the score code.

Nodes that modify the observations of the input variables or that create scoring formulas generate components of score code. The following is a list of nodes that generate components of scoring code.

Node Name	What the Node Creates in the Score Code
Transform Variables	creates new variables from data set variables
Impute	replaces missing values
Principal Components	creates principal components
Cluster	creates a new segment ID column and replaces missing values
Variable Selection	creates predicted values
Filter	creates a variable identifying the filtered observations
SOM/Kohonen	creates a segment variable and SOM dimension variables
Replacement	replaces class levels and unknown levels
Model Comparison	creates the bin variable (b_)
Score Node	creates fixed variable names
Regression	creates posterior probabilities, predicted values, and classification variables
Neural Network	
Decision Tree	
AutoNeural	
Dmine Regression	
Rule Induction	
DMNeural	
Two Stage	
Ensemble	
Decision	

Score Node Properties

Score Node General Properties

The following general properties are associated with the Model Comparison Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Score node added to a diagram will have a Node ID of Score. The second Score node added to a diagram will have a Node ID of Score2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Score window. The Imported Data — Score window contains a list of the ports that provide data sources to the Score node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Score window. The Exported Data — Score window contains a list of the output data ports that the Score node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Score Node Train Properties

- **Variables** — Select the ... button to open the Variables — Score table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use variable values. The Name, Role, and Level values for a variable are displayed as read-only properties. The following buttons and check boxes provide additional options to view and modify variable metadata.

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.

    - **Reset** — Changes metadata back to its state before use of the Apply button.
    - **Label** — Adds a column for a label for each variable.
    - **Mining** — Adds columns for the Order, Report, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
    - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
    - **Statistics** — Adds statistics metadata for each variable.
    - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Type of Scored Data** — Use the Type of Scored Data property of the Score node to specify the scored data type. You can choose from **View** or **Data**. The default setting for the Type of Scored Data is **View**.
- **Use Fixed Output Names** — Set the Use Fixed Output Names property of the Score node to **No** to suppress fixed output names. The default value for the Use Fixed Output Names property is **Yes**. For more information, see "Fixed Output Names" on page 1040 .
- **Hide Variables** — Set the Hide Variables property of the Score node to **Yes** if you want to prevent the original variables from appearing in the exported metadata when the data set is scored. When set to **Yes**, the original variables will be removed from the exported metadata, but not removed from the exported data sets and data views. The default setting for the Hide Variables property is **No**.
- **Hide Selection** — Select the ... button to the right of the Hide Selection property to open a window that you use to specify which variables are kept in the scored data sets. Variables are not actually dropped but hidden. The exported data sets and views will retain all variables, but the exported metadata will not display variables that are "dropped" using these properties:
    - **Input** — Set the Input property of the Score node to **No** if you want to be able to see variables with a role of Input in the exported metadata. The default setting for the Drop Inputs property is **Yes**.
    - **Target** — Set the Target property of the Score node to **No** if you want to be able to see variables that have a role of Target in the exported metadata. The default setting for the Drop Targets property is **Yes**.
    - **Rejected** — Set the Rejected property of the Score node to **No** if you want to be able to see variables that have a role of Rejected in the exported metadata. The default setting for the Drop Rejected Variables property is **Yes**.
    - **Frequency** — Set the Frequency property of the Score node to **No** if you want to be able to see variables that have a role of Frequency in the exported metadata. The default setting for the Frequency property is **Yes**.
    - **Assess** — Set the Assess property of the Score node to **No** if you want to be able to see variables that have a role of Assess in the exported metadata. The default setting for the Drop Assess Variables property is **Yes**.
    - **Predict** — Set the Predict property of the Score node to **No** if you want to be able to see variables that have a role of Predict in the exported metadata. The default setting for the Drop Predict Variables property is **Yes**.
    - **Residual** — Set the Residual property of the Score node to **No** if you want to be able to see variables that have a role of Residual in the exported metadata. The default setting for the Drop Residual Variables property is **Yes**.

- **Classification** — Set the Classification property of the Score node to **No** if you want to be able to see non-interval variables in the exported metadata. The default setting for the Drop Classification Variables property is **Yes**.
- **Other** — Set the Other property of the Score node to No if you want to be able to see variables that have a role of Other in the exported metadata. The default setting for the Drop Other Variables property is **Yes**.

Train Properties: Score Data

Use the Score Data properties to determine whether or not to score validation and test data sets.

- **Validation** — Set the Validation property to **Yes** if you want to apply the score code to the incoming validation data set, if one exists. If the Score node does not process the validation data set, it will be exported by the node as a SAS view, in a *.sas7bvew file. The default setting for the Score Validate property is **No**.
- **Test** — Set the Test property to **Yes** if you want to apply the score code to the incoming test data set, if one exists. If the Score node does not process the test data set, it will be exported by the node as a SAS view, in a *.sas7bvew file. The default setting for the Score Test property is **No**.

Train Properties: Score Code Generation

Score code conversion from SAS code to C or Java code enables you to use the code in additional analyses and reports outside Enterprise Miner. The score code conversion occurs automatically. However, conversion is only performed when the SAS code is in the format of a single DATA step. No conversion is performed for the Enterprise Miner Desktop version.

- **Optimized Code** — Set the Optimized Code property to **No** if you do not want to optimize the score code that is generated. The default setting for the Optimized Code property is **Yes**.

  Note: The optimized score code does not support Filter node score code. You must set the **Optimized Code** property for the Score node to **No** if you want to score Filter node output code.
- **C Score** — Set the C Score property to **No** if you want to suppress the generation of C Score code. The default setting for the C Score property is **No**.
- **Java Score** — Set the Java Score property to **No** if you want to suppress the generation of Java Score code. The default setting for the Java Score property is **No**.
- **Java Package Name** — Identifies the source of the Java score code package name. The Default setting creates a package named according to the global preferences in the Application menu. To view the global preferences select **Options** ⇨ **Preferences**.

  Note: If no default is specified, then the parameter "eminer.user.nodeid.score" is used.
- **User Package Name** — When the Java Package Name property is set to User Defined, use the User Package Name property to specify the user-defined package name.

Note: C score code and Java score code is disabled if your input data set uses a SASHDAT library.

Score Node Report Properties

The following report property is associated with the Score node:

- **Graphical Reports** — Produces graphical reports with Score node output.

Score Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Fixed Output Names

When you set the Use Fixed Output Names to Yes, the Score node generates score code to map the output variables to fixed output names.

The following fixed output names are generated:

Fixed Output Name	Definition
EM_PREDICTION	The prediction variable for an interval target.
EM_PROBABILITY	Posterior probability associated with the predicted classification. That is, it corresponds the maximum of the posterior probabilities, max(P1, P2, ..., Pk).
EM_EVENTPROBABILITY	Posterior probability associated with target event.
EM_DECISION	D_targetname, a fixed-name EM variable that the Score node maps to target name-based variables. The formula for D_targetname varies with the data mining model. For example, a class target D_BAD could be the result of a transformation of the model probability, the prior probability, the decision matrix, and the cost variable (if any). The decision with the maximum profit or minimum loss becomes the value of D_BAD. You must check the score code of a model to find the actual formula for D_targetname.

Fixed Output Name	Definition
EM_PROFIT	EP_targetname, the expected profit predicted for a target variable.
EM_LOSS	EL_targetname, the expected loss predicted for a target variable.
EM_CLASSIFICATION	I_variable, the prediction variable for a class target.
EM_SEGMENT	segment

Scoring with Non-DATA-step Format Score Code

The Score node will score even if some nodes in your training process flow do not generate score code in DATA step format. For example, the MBR node score format contains the use of a procedure.

In addition, the Input Variables report in the Score Results window is only produced if the path generates score code in a single DATA step format.

Special Licensing Requirements

A limited number of SAS Enterprise Miner nodes produce scoring output that contains SAS procedure calls. The Score node cannot translate SAS procedure calls into standalone score code.

If the process flow diagram that you want to score contains one of the following nodes, you cannot use the Score node to save and export model score code for use outside of the SAS Enterprise Miner environment:

- Association node
- MBR node
- Text Miner node
- SVM Node

If your scoring model contains one of the listed nodes and you need to score new data, you must use the SAS Enterprise Miner environment to run your score code, on a server that holds SAS Enterprise Miner and SAS Text Miner licenses. If you export the score code for standalone use, the code will not run.

Score Node Results

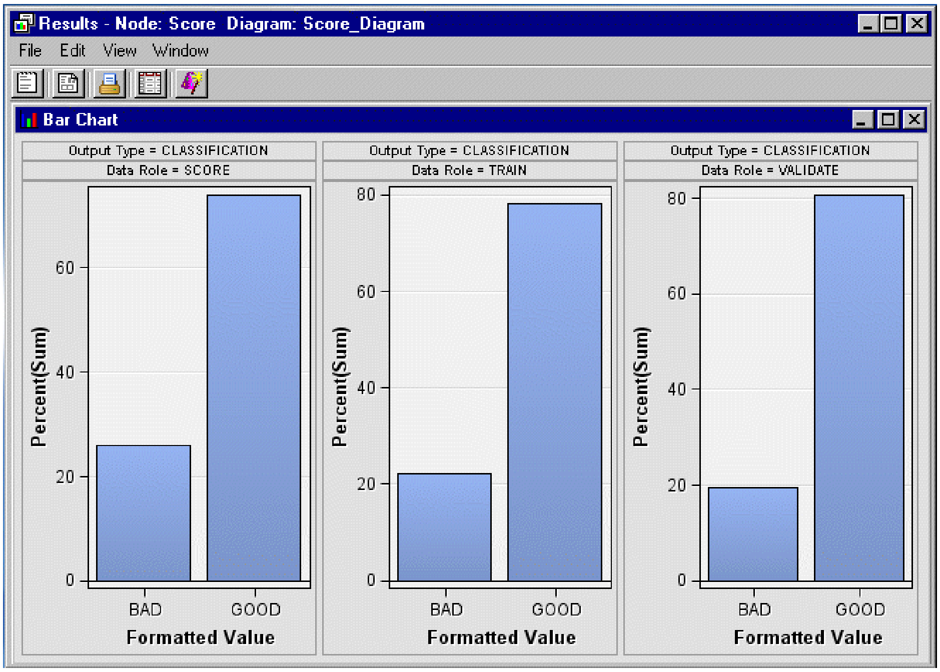
You can open the Results window of the Score node by right-clicking the node and selecting **Results** from the menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

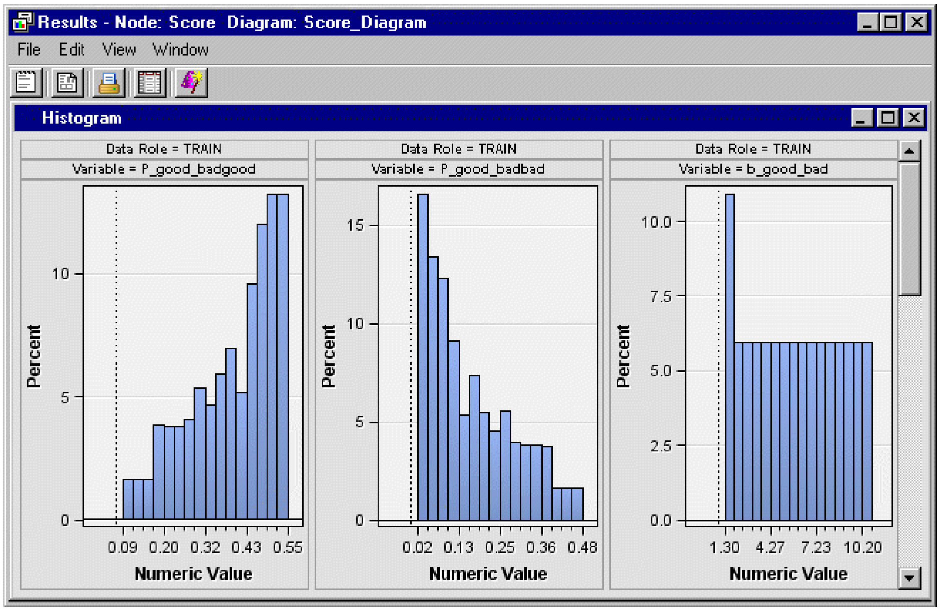
- **Properties**
  - **Settings** — displays a window with a read-only table of the Score node properties configuration when the node was last run.

- **Run Status** — indicates the status of the Score node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
- **Variables** — a read-only table of variable meta information.

- **Scoring**
  - **Input Variables** — a table of the input variables used in the score code generated by the process flow.
  - **Output Variables** — a table of the output variables created by the score code generated by the process flow.
  - **SAS Code** — the SAS score code that was generated by the process flow. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **Optimized SAS Code** — the optimized code generated.
  - **C Score Code** — the C score code translation of the SAS score code generated by the flow.
  - **Java Score Code** — the Java score code translation of the SAS score code generated by the flow.
- **SAS Results**
  - **Log** — the SAS log of the Score node run.
  - **Output** — the SAS output of the Score node run. The SAS output includes a variable summary, a distribution of missing observations in training data table, and an imputation summary by variable.
  - **Flow Code** — the SAS code used to produce the output that the Score node passes on to the next node in the process flow diagram.
- **Graphs**
  - **Bar Chart** — displays a bar chart of the values of the target variable for classification, decision, and segment output types, if applicable, for each data set.


Results - Node: Score Diagram: Score_Diagram
File
Edit
View
Window
Bar Chart
Output Type = CLASSIFICATION
Data Role = SCORE
Output Type = CLASSIFICATION
Data Role = TRAIN
Output Type = CLASSIFICATION
Data Role = VALIDATE
Percent(Sum)
0
20
40
60
80
BAD
GOOD
Formatted Value

- **Histogram** — displays a histogram of the values of the predicted, probability, profit, and loss output types, if available, for each of the data sets.



- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Graph Wizard menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Score Node Output Window

Class Variables Summary Statistics

You can view tables of the summary statistics for class variables. The tables contain the following columns:

- Variable
- Numeric Value
- Formatted Value
- Frequency
- Percent

The statistics are displayed for the following output types:

- Classification
- Model Decision
- Segment

Interval Variables Summary Statistics

The following statistics are displayed in the interval variables summary statistics tables:

- Mean

- Standard Deviation
- N
- Minimum
- 25th Percentile
- Median
- 75th Percentile
- Maximum

The following statistics are displayed in the interval variables summary statistics tables:

- Probability
- Profit
- Loss
- Predicted

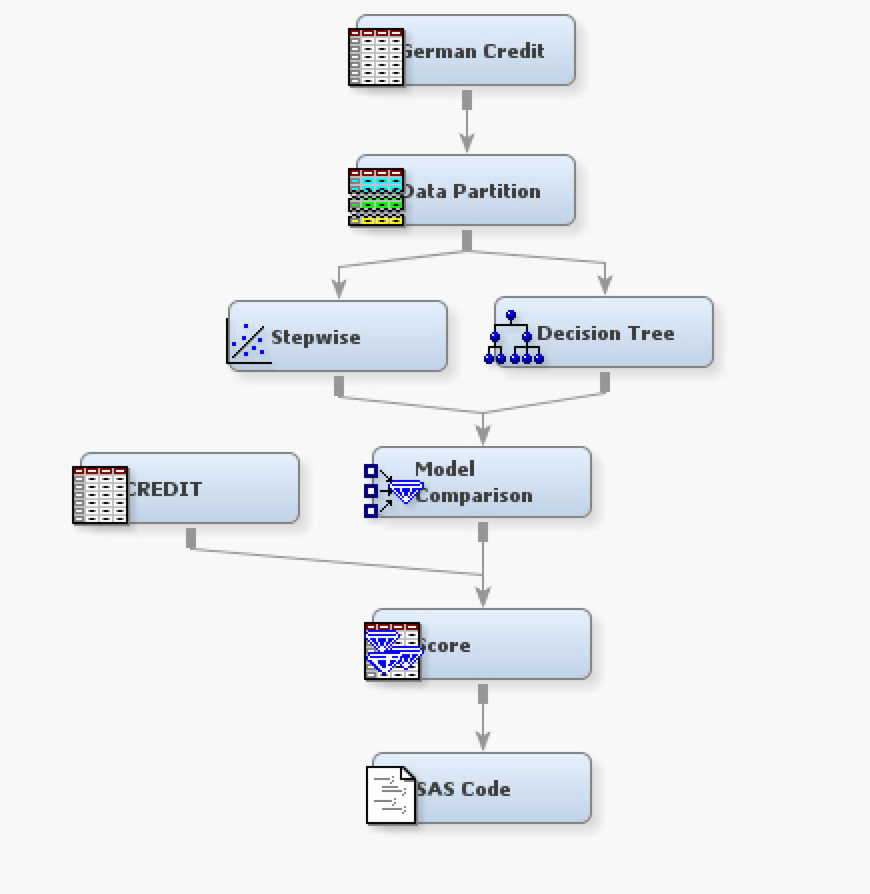
Score Node Example

Overview

Suppose you are a credit analyst and you want to build models that predict the credit worthiness of credit applicants. In this example, you build two models using the Tree and Regression nodes, use the Model Comparison node to choose a suitable model, and then use the training formula from the model to screen new applicants (score the Score data set).

The input data set that is used to train the model is named SAMPSIO.DMAGECR (stored in the sample library). The data set contains 1000 past applicants and their resulting credit rating (GOOD or BAD). There are 3 interval inputs and 17 class inputs for predicting the binary target GOOD_BAD.

The instructions for creating this process flow diagram are provided in the sections that follow.



Create a Score Data Set

Create a fictitious score data set without the target variable good_bad by submitting the following program from the Program Editor window:

```
/* Random sample without replacement */
data test.credit(drop = i j count good_bad);
   count=0;
   array obsnum(200) _temporary_;
   do i = 1 to 200;
    redo:
    select=ceil(ranuni(12345)*n);
    set sampsio.dmagecr point=select nobs=n;
    do j=1 to count;
      if obsnum(j)=select then goto redo;
    end;
    count=count+1;
    obsnum(count)=select;
    output;
   end;
   STOP;
 run;
```

For the code above to create the score data set, you must first add the TEST libname definition to your Enterprise Miner project start code. To modify the project start code, go to the Project Navigator in the upper left corner of the window, and select the project name at the root of the Project Navigator tree. With the project name highlighted, the Properties Panel directly below the Project Navigator will contain a Project Start Code property. Select the button to the right of the Project Start Code property to open the Project Start Code window for your project. Enter the following text in the Project Start Code window:

```
libname test 'c:\temp';
```

After you enter the text, select the Run Now button and close the Project Start Code window. After you define the TEST libname to Enterprise Miner, you can submit the above code to create a score data set via the Enterprise Miner Program Editor.

Create a New Project and a New Diagram

The Score node example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

Create the Data Sources

First, you will need to create the input data source by completing the steps below:

1. From the main menu, select **File** ⇨ **New** ⇨ **Data Source**.
2. Click **Next** in the Import Type window of the Data Source Wizard. You use the SAS Table import type.
3. Type **`SAMPSIO.DMAGECR`** in the Table field of the SAS Import Table window. Click **Next**.
4. Click **Next** in the Table Information window.
5. Select the **Advanced** button in the Metadata Advisor Options window. Click **Next**.
6. In the Column Metadata window, set the role of the variable GOOD_BAD to **Target**. Click **Next**.
7. If the Decision Processing window appears as an option, click **Next**. You do not use decision processing in this example.
8. In the Data Source Attributes window, type **`German Credit`** in the Name field. Click **Next**.
9. In the Summary window, click **Finish**.

Next, you are ready to create the score data source.

1. From the main menu, select **File** ⇨ **New** ⇨ **Data Source**.
2. Click **Next** in the Import Type window of the Data Source Wizard. You use the SAS Table import type.
3. Type **`TEST.CREDIT`** in the Table field of the SAS Import Table window. Click **Next**.
4. Click **Next** in the Table Information window.
5. Select the **Advanced** button in the Metadata Advisor Options window. Click **Next**.
6. Click **Next** in the Column Metadata window.
7. In the Data Source Attributes window, set the Role to **Score**. Click **Next**.
8. In the Summary window click **Finish**.

Finally, add the data source to the diagram.

1. Expand the Data Source folder in the Project Panel.

2. Drag the German Credit data source to the diagram workspace.

Partition the Data

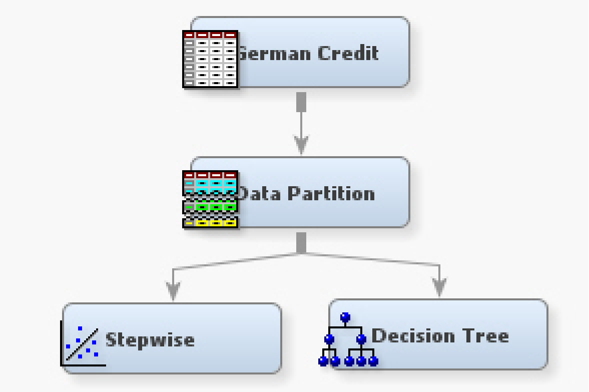
1. Drag a Data Partition node from the Sample tab of the node toolbar to the Diagram Workspace.
2. Connect the Data Partition node to the German Credit data node.
3. Click the Data Partition node to highlight it.
4. In the Properties Panel, set the Training property to `70`. The training data set is used to fit the model.
5. Set the Validation property to `30`. The validation data set is used for assessment; it can also be used to prevent the modeling nodes from overtraining.
6. Set the Test property to `0`. If the sum of the percentages for the Training, Validation, and Test properties does not add up to 100%, then the default values for these properties are used when the data is partitioned.

Build a Regression Model

1. Drag a Regression node from the Model tab of the node toolbar to the Diagram Workspace.
2. Connect the Regression node to the Data Partition node.
3. Select the Regression node and make the following property settings:
   - Set the Selection Model property to **Stepwise**.
   - Set the Selection Criteria property to **Validation Misclassification**.
   - Change the Use Selection Defaults property to **No**.
   - Select the ... button to the right of the Selection Options property to open the Selection Options window. In the Selection Options window, set the Entry Significance Level property and the Stay Significance Level property to `0.10`. Stepwise regression systematically adds and deletes variables from the model based on the entry and stay significance levels.
4. Right-click the Regression node and select **Rename**. Type Stepwise in the Rename window and click **OK**.

Build a Decision Tree Model

1. Drag a Decision Tree node from the Model tab of the nodes toolbar to the Diagram Workspace.
2. Connect the Decision Tree node to the Data Partition node.
3. Use the default settings of the Decision Tree node to predict the target GOOD_BAD.



Assess the Models

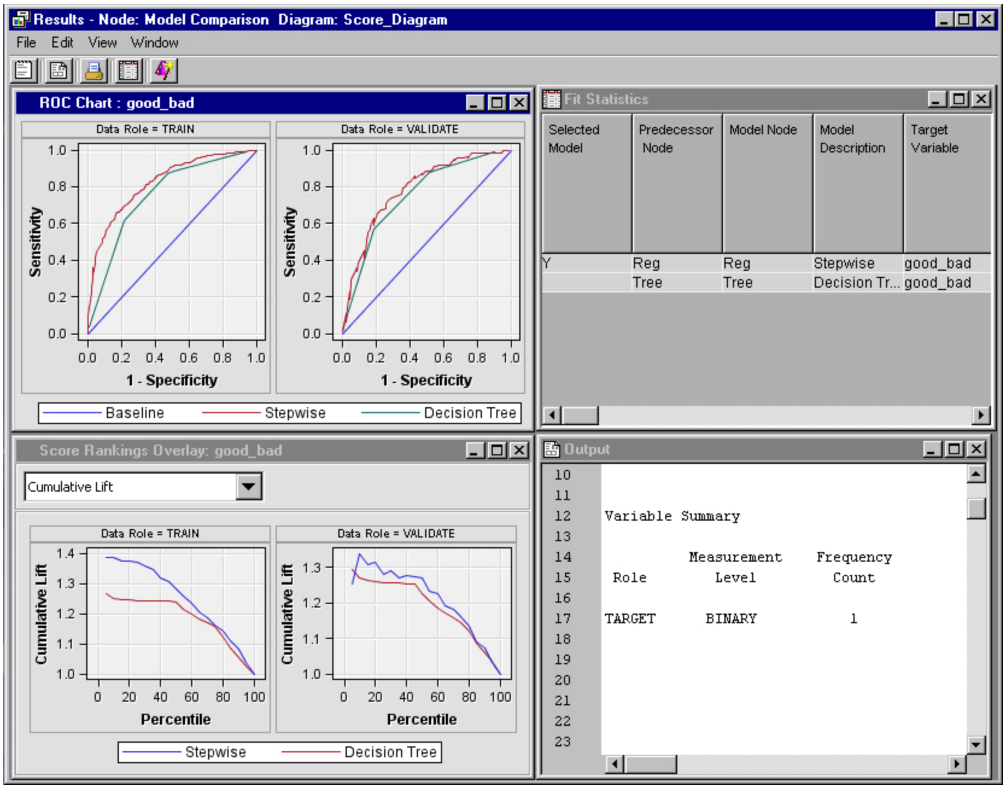
1. Add a Model Comparison node from the Assess tab to the Diagram Workspace.

2. Connect each modeling node (Stepwise and Decision Tree in this example) to the Model Comparison node.

3. Right-click the Model Comparison node and select **Run**.

4. Click **OK** in the Run Status window when the Model Comparison node finishes running.

   Right-click the Model Comparison node and select **Results**.

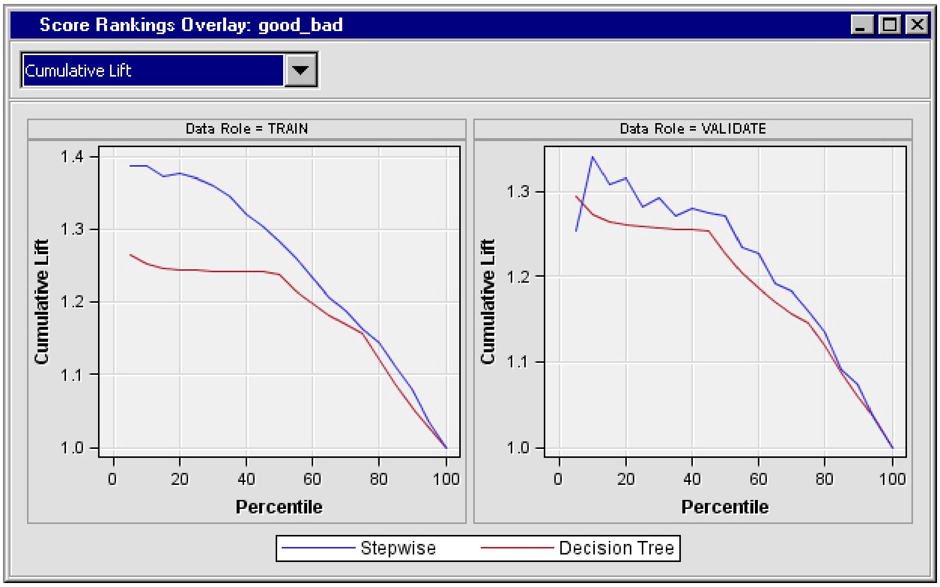


Selected Model	Predecessor Node	Model Node	Model Description	Target Variable
Y	Reg	Reg	Stepwise	good_bad
	Tree	Tree	Decision Tr...	good_bad

```


Variable Summary

          Measurement     Frequency
 Role        Level          Count

TARGET      BINARY            1
```

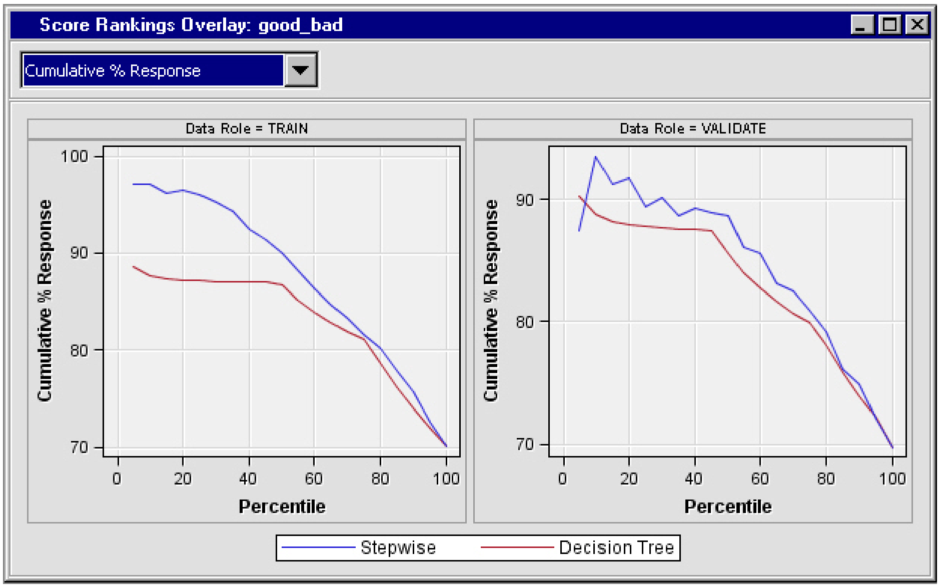
The Output window in the Model Comparison Results window displays assessment statistics for both models and indicates that the regression model has been selected as the best model based on the validation misclassification rate.

5. Expand the Score Rankings Overlay: good_bad window.



The stepwise regression model has higher lift values than the decision tree model over almost all deciles.

6. Select **Cumulative % Response** from the drop-down menu.



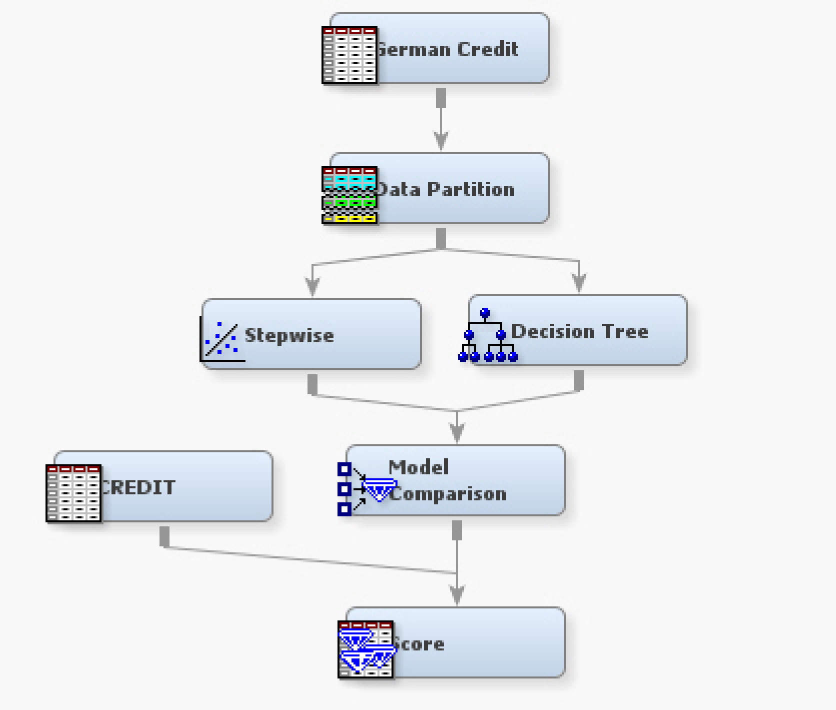
The stepwise regression model captures about 93% of the good credit applicants in the first decile of the validation data. The overall performance of this model is superior to the tree model. Therefore, we will use the scoring code from the regression model to score new applicants. You should devote more time and effort to the explore, modify, and model phases of the SEMMA methodology.

Add the Score Node to the Diagram Workspace and Score the Data

1. Drag the CREDIT score data source that you created from the Data Sources folder in the Project Panel to the Diagram Workspace.

2. Add a Score node from the Assess tab to the Diagram Workspace.

3. Connect the CREDIT data source to the Score node. Connect the Model Comparison node to the Score node. At this point, your process flow should resemble the following:



4. Click the Score node to select it. In the Properties Panel, change the Type of Scored Data property to **Data**.

5. Right-click the Score node and select **Run**. Click **OK** when the run is complete.

View the Distribution of Predicted Values for Good Applicants

1. With the Score node still highlighted, select "Exported Data" from the property sheet. In the Exported Data window, click the row associated with the Score port, and click the **Explore** button.

2. In the Explore window, click **Plot** on the Sample Properties window.

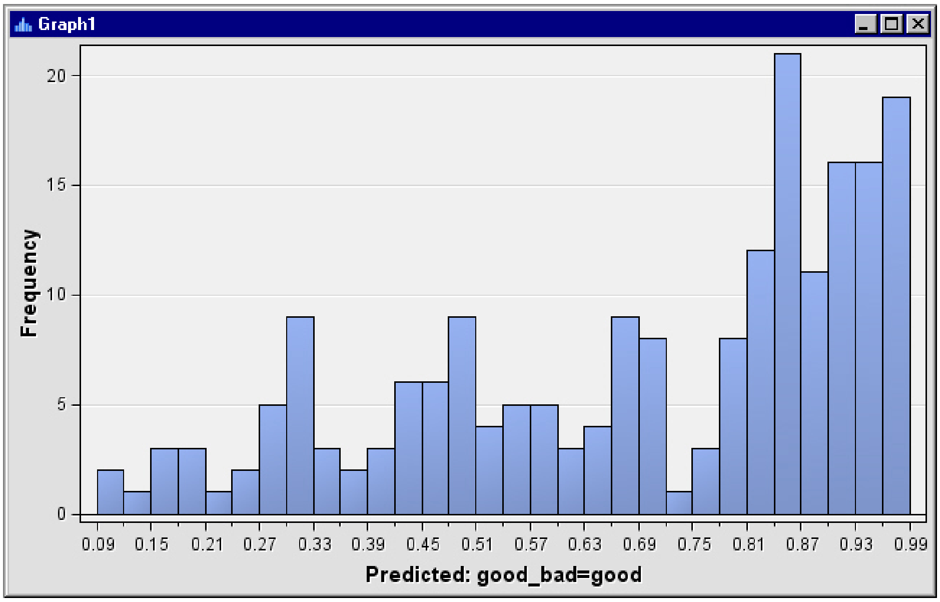
3. In the Select a Chart Type window, select **Histogram**. Use the default Histogram selection, and click **Next**.

4. In the Select Chart Roles window, find the row that contains the variable P_GOOD_BADGOOD. This variable represents the predicted values for GOOD_BAD=GOOD for each observation. Click the Role column of the row that contains the P_GOOD_BADGOOD variable, and select X from the menu. Click **Finish**.

5. A histogram opens in the Explore window. Follow these steps to increase the number of bins that are displayed in the plot.

   - Right-click the graph and select **Graph Properties**.
   - In the Bins pane, select 30 for the Number of x Bins field.
   - Click **OK**.

   The graph shows 30 bins.

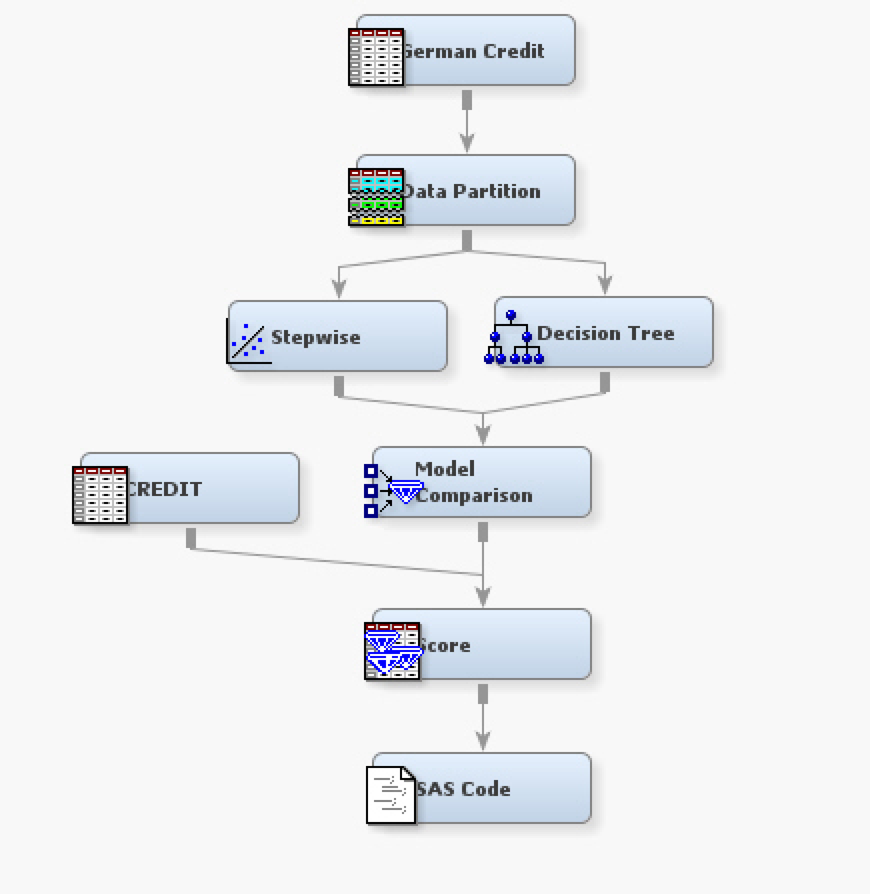


The histogram shows the distribution of the predicted values for credit applicants. These values can be interpreted as the probability that the applicant will have good credit. Applicants in bins to the right on the histogram are more likely to have good credit than applicants in bins to the left.

The next step is to select your threshold level for risk tolerance. For example, if you want to issue a credit card to any applicant that has more than a 75% probability of having good credit, then set the threshold to 0.75. Once you have chosen the threshold, you can create a data set that contains only those applicants that are deemed credit worthy.

Create a Report of Credit Worth Applicants

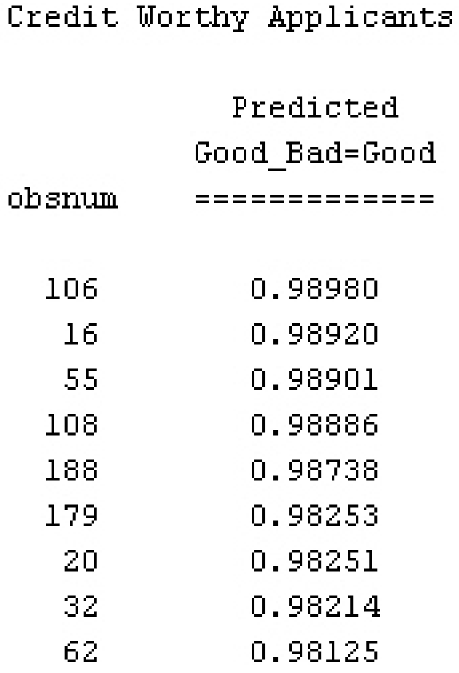
The last step is to create a list of credit worthy applicants. Candidates that are deemed to be good credit risks should have P_GOOD_BADGOOD predicted values that are greater than 0.75 (credit cutoff threshold).



1. Add a SAS Code node from the Utility tab to the Diagram Workspace.
2. Connect the SAS Code node to the Score node.
3. Click the SAS Code node to select it.
4. In the Train section of the Properties Panel, select the ... button to the right of the Code Editor property.
5. Type the following code in the code window.

```
DATA goodapps;
   SET &EM_IMPORT_SCORE
   obsnum=_N_;
   IF p_good_badgood <= 0.75
     THEN delete;
   run;

PROC SORT data=goodapps ;
   BY descending p_good_badgood ;
   run;

PROC PRINT data= goodapps noobs
   split='*' ;
   VAR obsnum p_good_badgood;
   LABEL p_good_badgood='Predicted*Good_Bad=Good*============';
   TITLE "Credit Worthy Applicants";
run;
```

The output displays a list of all observations that had a probability of good credit higher than 0.75. A snippet of this information is shown below.



```
Credit Worthy Applicants

                 Predicted
              Good_Bad=Good
obsnum        =============


  106            0.98980
   16            0.98920
   55            0.98901
  108            0.98886
  188            0.98738
  179            0.98253
   20            0.98251
   32            0.98214
   62            0.98125
```

Chapter 69
Segment Profile Node

Segment Profile Node

Overview of the Segment Profile Node

In the Enterprise Miner SEMMA data mining process, the Segment Profile node belongs to the Assess group. The Segment Profile node enables you to examine segmented or clustered data and identify factors that differentiate data segments from the population. The node generates various reports that aid in exploring and comparing the distribution of these factors within the segments and population.

The key variables for a specific segment are determined by creating a binary pseudo target variable based on the membership of the segment. Two methods can be used to determine the differentiating variables. In the default method, each of the interval and class target variables are ranked based on their logworth value. The value in turn, is based on the pseudo target. The interval variables are binned to identify their maximum logworth. The second method builds a decision tree of specified depth. Variables are selected based on their total differentiating capabilities. This second method permits a degree of interaction between the variables.

Segment Profile Node Data Set Requirements

The results that the Segment Profile node generates vary, depending on the variable roles that are assigned:

- segment class variables are treated as BY variables one at a time. If no segment variables are present, then no reports are generated.

- input variables, if present, are filtered for each level of every segment variable. Only the variables that are selected during filtering are summarized and included in the various reports.
- report variables are not filtered, but are summarized in the same way as selected input variables.

In addition, target variables can be processed as segment, input, or report variables when setting the Target Analysis Role property. Profile analysis is affected by the role of the target variable.

Variations include setting the target variable analysis role as follows:

- Input — If a segment variable has been set, the target variable will be filtered and might appear in reports that are generated. If no segment variable has been set, then the target variable is ignored.
- Segment — The values of the target variable are used as BY values.
- Report — The target variable should appear in all reports that are generated.
- None — The target variable is ignored.

Segment Profile Node Data Set Role

The data set that you analyze with the Segment Profile node must have a data set role of **Train** or **Raw**.

Segment Profile Node Properties

Segment Profile Node General Properties

The following general properties are associated with the Segment Profile Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Segment Profile node added to a diagram will have a Node ID of Prof. The second Segment Profile node added to a diagram will have a Node ID of Prof2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Segment Profile window. The Imported Data — Segment Profile window contains a list of the ports that provide data sources to the Segment Profile node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Segment Profile window. The Exported Data — Segment Profile window contains a list of the output data ports that the Segment Profile node creates data for

when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Segment Profile Node Train Properties

The following train properties are associated with the Segment Profile node:

- **Variables** — Select the ... button to open the Variables — Profile Segment table, which allows you to view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. You can specify Use and Report variable values. The Name, Role, and Level values for a variable are displayed as read-only properties.

  The following buttons and check boxes provide additional options to view and modify variable metadata:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the Apply button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that allows you to view a variable's sampling information, observation values, or a plot of variable distribution.

Segment Profile Node Train Properties: General

- **Number of Midpoints** — use the Number of Midpoints property of the Segment Profile node to specify the number of midpoints that will be used to compute the distribution of interval variables, or the number of bins in the distribution histogram. Acceptable values for Number of Midpoints are 8, 16, or 32. This property can be useful if some of the variables have a wide range. If an interval variable has a number of distinct values less than the number of midpoints specified, then each value will be used as a midpoint in the histogram. The default value for the Number of Midpoints property is 8.

- **Profile All** — Use the Profile All property of the Segment Profile node to specify if you want to profile all of the segments. When the property is set to **No**, small segments will be combined into the _OTHER_ category. The default setting for the binary Profile All property is **No**.

- **Cutoff Percentage** — Use the Cutoff Percentage property setting to specify the threshold level for combining segments. The segments are ordered by frequency from largest to smallest. When the Profile All property is set to No and the cumulative frequencies exceed the cutoff percentage value, the node groups the remaining segments into a collective _OTHER_ segment. This property is useful when the segment variable has a large number of segments and many of them have small membership. Permissible Cutoff Percentage values are: 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, and 99. The default setting for the Cutoff Percentage property is 95.

Segment Profile Node Train Properties: Input Variables

- **Number of Inputs** — Use the Number of Inputs property of the Segment Profile node to specify the maximum number of inputs that you want to appear in the results. For example, if the Number of Inputs property is set to 20, and a data set that has 10,000 input variables is submitted to the Segment Profile Node, the maximum number of inputs displayed in the results will be 20. The Number of Inputs property is ignored when the Maximum Depth property is greater than 1. Permissible values for the Number of Inputs property are integers greater than or equal to 1 and less than or equal to 25. The default value for the Number of Inputs property is 10.

- **Minimum Worth** — Use the Minimum Worth property of the Segment Profile node to specify the minimum worth value that you want to use when you are evaluating the logworth of each variable in your training data set. Only variables that meet these criteria are kept in the filtering process. The property is ignored when the Maximum Depth property is greater than 1. Permissible values for the Minimum Worth property are real numbers greater than 0 and less than or equal to 1. The default value for the Minimum Worth property is 0.01.

- **Maximum Depth** — Use the Maximum Depth property of the Segment Profile node to specify the maximum depth of the decision tree used to discriminate variables. When set to 1, the logworth is used to rank the variables. Otherwise, an importance measure is used to select the variables. Permissible values for the Max Depth property are 1, 2, 3, and 4. The default value of the Maximum Depth property is 1.

- **Print Worth Statistics** — Use the Print Worth Statistics property of the Segment Profile node to specify whether to print the worth statistics that are calculated for each segment in the data set. The default setting for the Print Worth Statistics property is **Yes**.

Segment Profile Node Train Properties: Target Variables

- **Analysis Role** — Use the Analysis Role property to specify the target variable role that you want to use during the profiling process. The permissible choices for the Analysis Role property are **None**, **Input**, **Report**, and **Segment**. The default setting for the Analysis Role property is **None**.

Segment Profile Node Train Properties: Report Variables

- **Use Report Variables** — Specifies whether or not to use report variables in the generated reports. Report variables are not filtered. The default setting for the Use Report property is **Yes**.

- **Number of Report Variables** — If the Use Report Variables property of the Segment Profile node is set to **Yes**, use the Number of Report Variables property to specify the number of report variables that you want to use when generating the various report data sets. The permissible values for the Number of Report Variables property are integers greater than or equal to 1 and less than or equal to 25. The default setting for the Number of Report Variables property is 10.

Segment Profile Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Segment Profile Node Results

Segment Profile Results Menu

You can open the Results window by right clicking the node and selecting **Results**. Besides the generated output, Segment Profile Node results display additional graphs, when relevant, to aid in segment profiling.

Right click each of the graphs to see an editing menu. Double click an individual graph to expand it. Place the mouse over a segment in a graph to display summary information about the segment. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view results. The options that are available vary based on variable roles set in your profiling.

- **Properties**
  - **Settings** — displays a window with a read-only table of the Segment Profile node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Segment Profile node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headers, and you can toggle column sorts between descending and ascending by clicking on the column headers.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Segment Profile node run.
  - **Output** — the SAS output of the Segment Profile node run. The output displays the summary statistics for the original partitioned data set and for each resulting partition.
  - **Flow Code** — flow code is dimmed and unavailable for the Segment Profile node.

- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Segment Profile node does not generate PMML code.
- **Plots** — contains information about the first segment generated for each designated segment variable. Any population values are assigned a segment variable of _OVERALL_.
  - **Profile: [Segment Variable Name]** — displays a lattice graph that was generated by the profile run. This is available only if the role of Segment has been designated. For more information, see "Segment Profile Plot" on page 1061 .
  - **Variable Worth: [Segment Variable Name]** — displays a bar chart that represents the calculated worth for each input variable in a segment. This option will be available only if at least one Segment variable is designated. For more information, see "Segment Profile Variable Worth Plot" on page 1063 .
  - **Segment Size: [Segment Variable Name]** — displays a pie chart that indicates the size of each segment that was generated for a single Segment variable. For more information, see "Segment Profile Segment Size Plot" on page 1063 .
- **Summary Statistics** — lists generated tables that contain various statistics about the segments and the population. These tables are used in the graphical displays.
  - **Size** — contains the frequency distribution (count and percent) of the processed segment variables. If no Segment variable is set, the Size data set is not created. When the depth of the decision tree that was created is greater than 1, this data set contains the misclassification rate for the training data set (_MISC_) and optionally the validation data set (_VMISC_).
  - **Profile Variables** — contains summary statistics for each selected variable and report variable per segment and for the population (_OVERALL_). This data set also contains either the logworth of selected variables or the importance of the selected variables and their rank. The table contains the following columns: Type, Segment Variable, Segment Value, Variable, Rank, Worth, Variable Label, Number of Levels, Missing, Minimum, Maximum, Mean, Standard Deviation, Skewness, and Kurtosis.
  - **Class Variables** — Contains the frequency distribution of the class variables for the various segments and the population. If no class variables are selected by the filtering process and no class Report variables are specified, then this data set is not generated.
  - **Interval Variables** — Contains the midpoints and associated count and percentage for the bars of the histogram of the segments and for the population. These are used to approximate the actual distributions. If no interval Input variables are selected by the filtering process and no interval Report variables are specified, then this data set is not generated.
- **Table** — displays a table that contains the underlying data used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard that you can use to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Segment Profile Plot

To open the Profile Plots, select **View** ⇨ **Plots** ⇨ **Profile:[Segment Variable Name]** from the Segment Profile Results View menu.

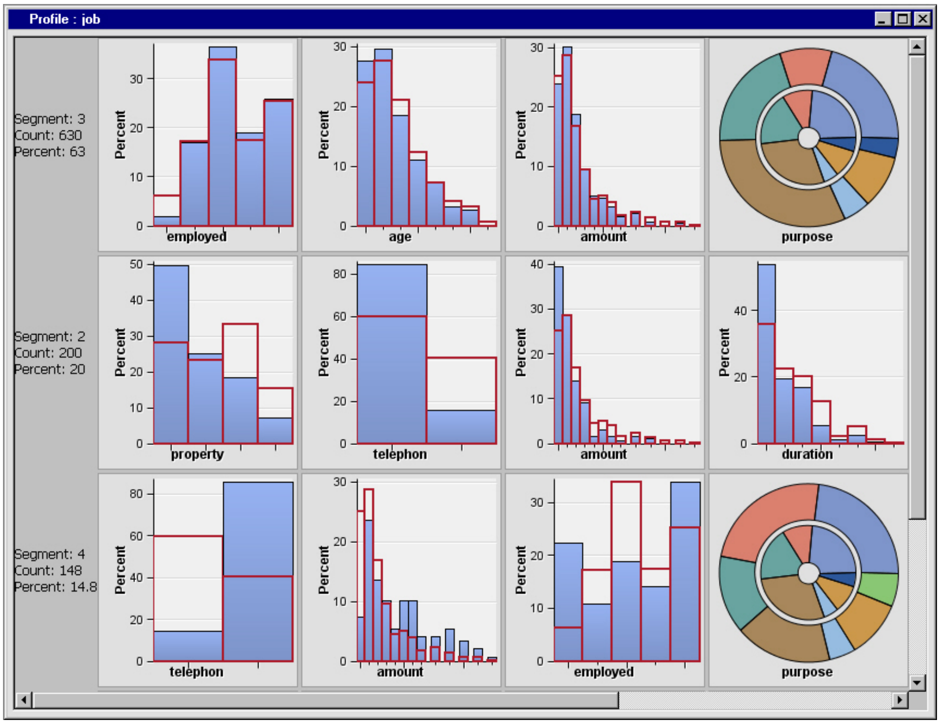
The Profile window displays a lattice, or grid, of plots comparing the distribution for the identified and report variables for both the segment and the population. Each row represents a single segment. The far left margin identifies the segment, its count, and percentage of the total population. By default, the rows are sorted in ascending size order from top to bottom. You can also sort rows alphanumerically by segment name by right-clicking to get the edit menu. Select **Sort Segments**. You can also change the response variable format to the count or the percent of the entire data and expand a graphic by using the edit menu.

The columns are organized from left to right according to their ability to discriminate that segment from the population. Report variables, if specified, will appear on the right in alphabetical order after the selected inputs.

The lattice graph has the following features:

- **Class Variable** — displayed as two nested pie charts that consist of two concentric rings. The inner ring represents the distribution of the total population. The outer ring represents the distribution for the given segment.
- **Interval Variable** — displayed as a histogram. The blue shaded region represents the within-segment distribution. The red outline represents the population distribution. The height of the histogram bars can be scaled by count or by percentage of the segment population. When you are using the percentage, the view shows the relative difference between the segment and the population. When you are using count, the view shows the absolute difference between the segment and the population.

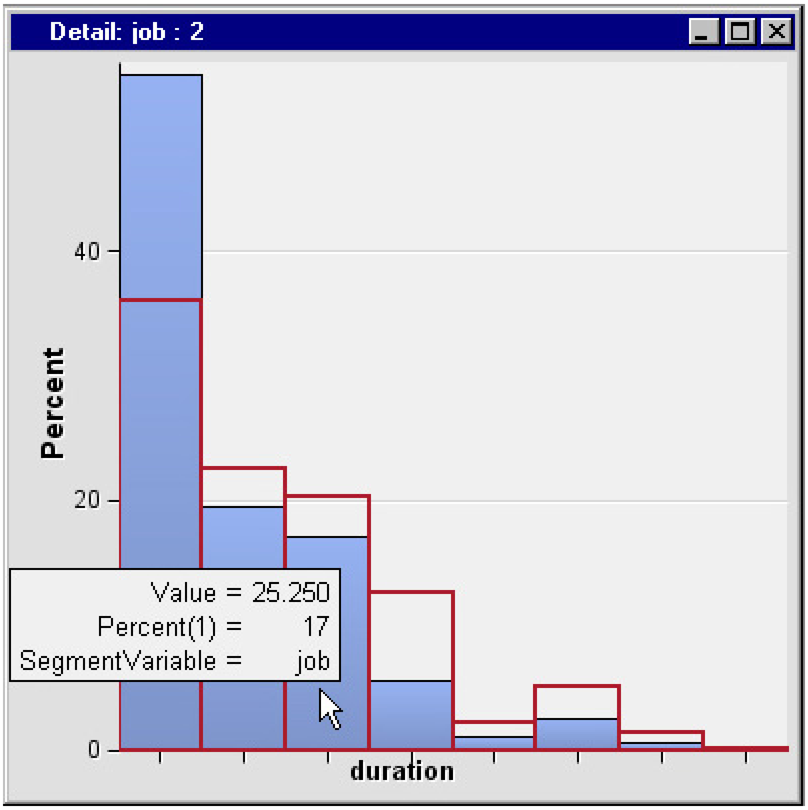
The following example displays a partial lattice graph for segments that were created, based on the Segment variable job. The graphs enable you to profile each segment and see the variable distribution in each segment that was created in the profile run.


Segment: 3
Count: 630
Percent: 63

Segment: 2
Count: 200
Percent: 20

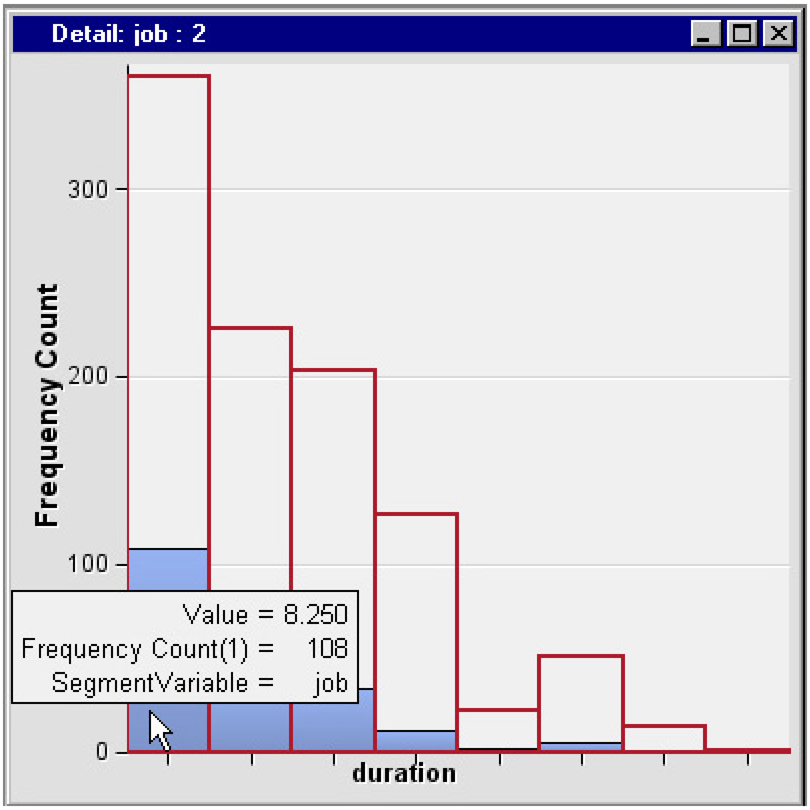
Segment: 4
Count: 148
Percent: 14.8

Below is the expanded histogram view for the variable duration in segment 2 using the percentage value. This view shows the relative difference in the distributions for the segment and the population for the variable duration in segment 2.



Value = 25.250
Percent(1) = 17
SegmentVariable = job

Below is the expanded view of the histogram. This view shows the variable duration using the count value. This view shows the absolute difference in the distributions for the segment and the population for the variable duration in segment 2.



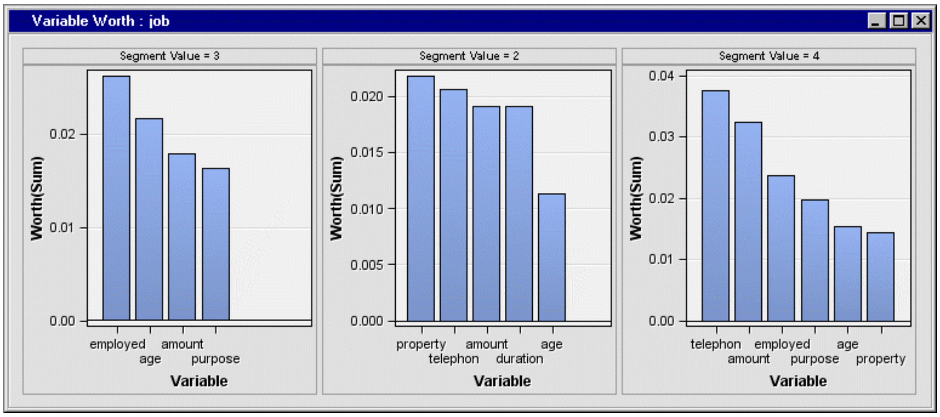
Value = 8.250
Frequency Count(1) = 108
SegmentVariable = job

Segment Profile Variable Worth Plot

To open the Variable Worth plot, select **View** ➪ **Plots** ➪ **Variable Worth:[Segement Variable Name]** from the Segment Profile Results View menu.

The variable worth plot shows the logworth of the factors (inputs) that have been identified for each segment. For example, the plot for Segment 4 shows that the variable telephone has more discriminant capability than variables such as age and property.

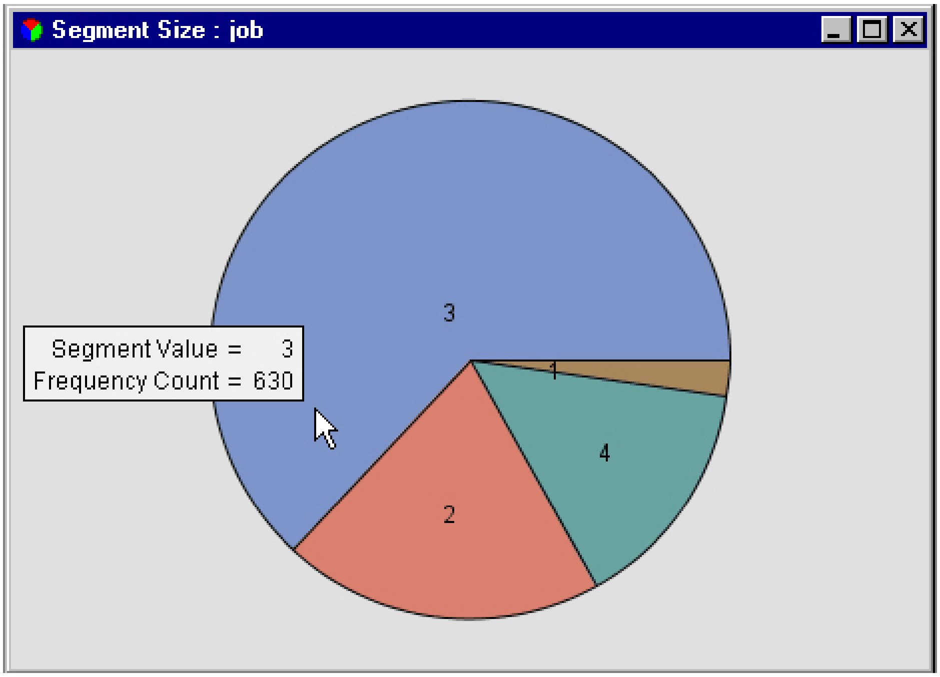
When the Maximum Depth property is greater than 1, this window is replaced by the Variable Importance window, which displays the importance statistic instead of the logworth.



Segment Profile Segment Size Plot

To open the Segment Size plot, select **View** ➪ **Plots** ➪ **Segment Size:[Segment Variable Name]** from the Results window View menu.

The segment size plot displays the relative sizes of the various segments. There might be multiple segments combined into the _OTHER_ category. You can change the threshold for the _OTHER_ category, using the Cutoff Percentage property, and rerun the node.



Place the mouse over a segment in the pie chart to display more information about that segment.

Segment Profile SAS Output Tables

The Output window from the Segment Profile Results displays the printed summary of the segmentation results. When a Segment variable is designated in the profiling, additional output is generated. The following sample output was generated from a Segment Profile node that was preceded by a Cluster node. The segment variable in this example is _SEGMENT_.

- Output that is generated for all the profiling runs displays variable summary information that is associated with the training data set as a whole.

```
Variable Summary
ROLE         LEVEL      COUNT

INPUT       INTERVAL      19
INPUT       NOMINAL        1
REJECTED    INTERVAL       1
SEGMENT     NOMINAL        1
TARGET      BINARY         1
```

- Output that is generated when a Segment variable is specified displays the following additional information about any Segment variable and the generated segments:
  - **Frequency tables** display frequency information for each Segment variable.

```
Frequencies: _SEGMENT_


                                                Percent of
 Segment      Segment     Frequency            Total
Variable       Value        Count            Frequency
```

```
_SEGMENT_    10               366        36.6
_SEGMENT_    3                271        27.1
_SEGMENT_    7                171        17.1
_SEGMENT_    2                 74         7.4
_SEGMENT_    4                 59         5.9
_SEGMENT_    9                 27         2.7
_SEGMENT_    _OTHER_           32         3.2
```

- **Decision Tree Importance Profiles** display the logworth or importance statistics for the variables that have been identified as factors that distinguish the segment from the population. When the Print Worth Statistics property is set to Yes, a table is produced for each segment of the specified segment variables. Note that if no variable meets the specified criteria of a particular segment, then the associated table will be omitted from the output listing. This table does not contain report variables because they are assessed to determine their contribution. For more information about the worth value see Decision Tree Node documentation.

```
Variable: _SEGMENT_ Segment: 10   Count: 366
Decision Tree Importance Profiles

Variable     Worth     Rank

amount      0.46409      1
duration    0.12920      2
purpose     0.04850      3
job         0.02934      4
property    0.02866      5
installp    0.02666      6
marital     0.02030      7
telephon    0.01903      8
```

Segment Profile Node Example

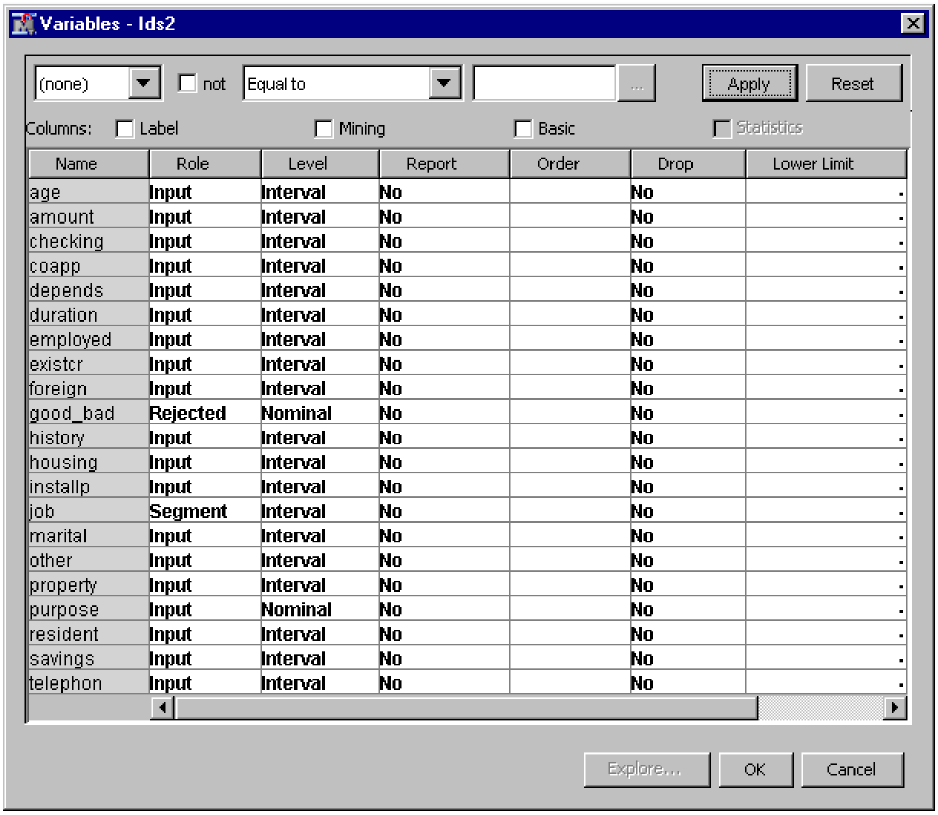
Introduction

The following scenario profiles customer credit input data. The Segment Profile node is used to identify factors that help to differentiate a segment from the overall population. This type of information is useful to understand your customer base and to aid in making informed decisions about potential actions.

- “Create a Process Flow Diagram” on page 1065
- “View the Results from the Segment Profile Node” on page 1066
- “Variable Layout for SAMPSIO.DMAGECR” on page 1068

Create a Process Flow Diagram

1. Create a data source named DMAGECR using the SAMPSIO.DMAGECR data set. Assign a role of SEGMENT to the variable Job and a role of REJECTED to the variable Good_bad.



Name	Role	Level	Report	Order	Drop	Lower Limit
age	Input	Interval	No		No	.
amount	Input	Interval	No		No	.
checking	Input	Interval	No		No	.
coapp	Input	Interval	No		No	.
depends	Input	Interval	No		No	.
duration	Input	Interval	No		No	.
employed	Input	Interval	No		No	.
existcr	Input	Interval	No		No	.
foreign	Input	Interval	No		No	.
good_bad	Rejected	Nominal	No		No	.
history	Input	Interval	No		No	.
housing	Input	Interval	No		No	.
installp	Input	Interval	No		No	.
job	Segment	Interval	No		No	.
marital	Input	Interval	No		No	.
other	Input	Interval	No		No	.
property	Input	Interval	No		No	.
purpose	Input	Nominal	No		No	.
resident	Input	Interval	No		No	.
savings	Input	Interval	No		No	.
telephon	Input	Interval	No		No	.

2. Add the data source DMAGECR to the diagram workspace.

3. Add a Segment Profile node to the diagram workspace and connect it to the data source DMAGECR.

4. Run the Segment Profile node using the default settings.

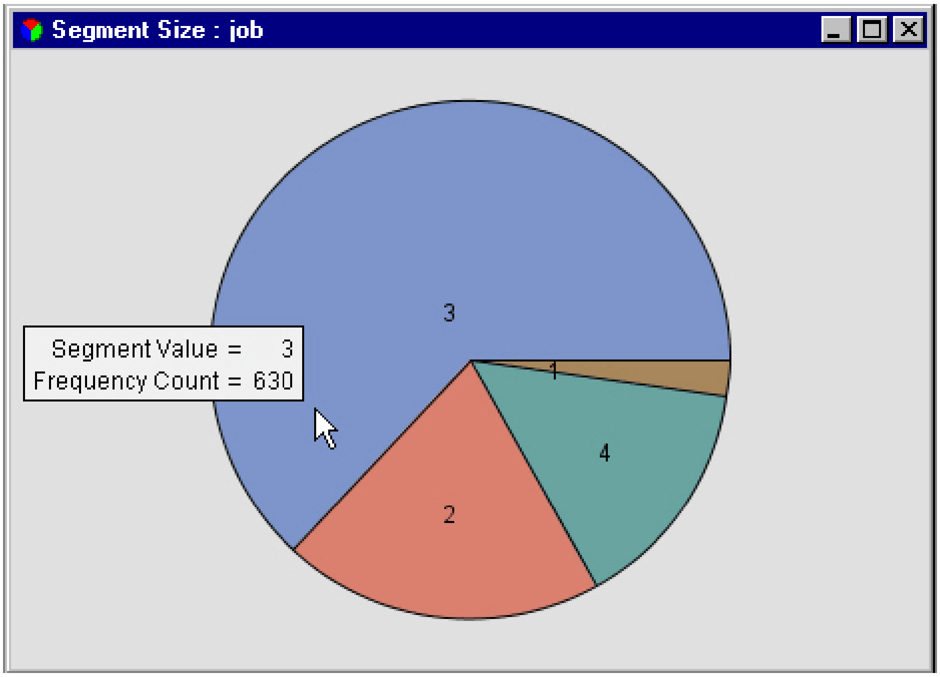
5. After the node runs successfully, click **OK** in the Run Status window. Open the Results window.

View the Results from the Segment Profile Node

The Results window displays several plots and the generated output from the profile run. Use the items under the **View** main menu to select results that are generated by the node.

1. Open the Output window and review information about the segments that were generated from the segment variable Job.

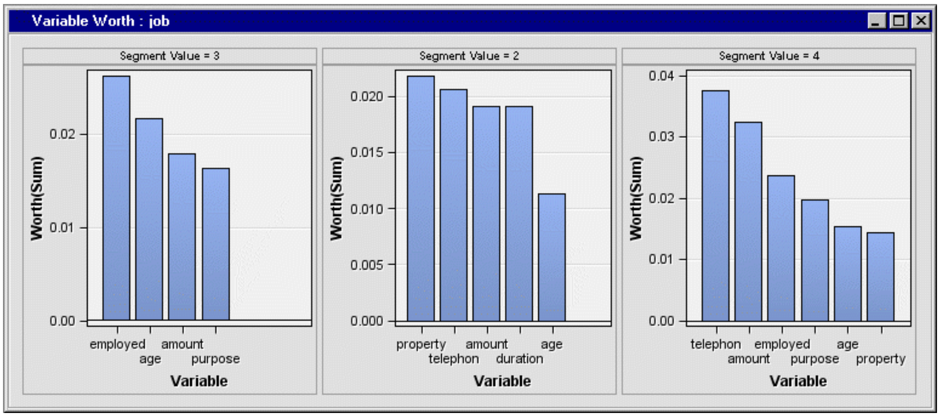
2. Open the **Segment Size:job** window. This displays the relative size of all segments that were generated for the segment variable Job.



Segment Value = 3
Frequency Count = 630

This graph shows that Segment 3 contains more than half of the customers in the credit data. Segment 1 is the smallest segment, containing only 2.2% of the customer credit data. Segment 1 might reveal interesting information about the data. However, this example focuses on the larger segments.

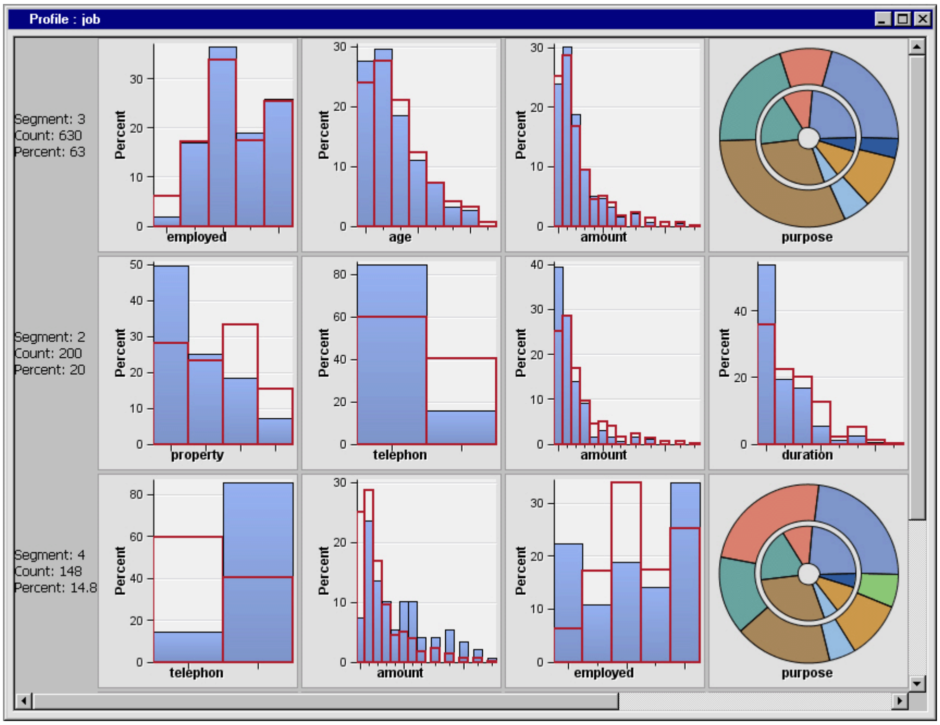
3. Open the **Variable Worth:job** window to see what variables were important in generating the different segments.



Looking at the worth of a variable in creating a segment can reveal different aspects of the segment. This type of information might be helpful in understanding the characteristic of certain customer segments and how they differ from the overall population.

The largest segment, Segment 3, resulted partly from a large distinction between the average segment value and the average overall population value for the variable that is named Employed. This variable represents the length of present employment. At first, it appears that this might be an effective customer segment to target in business decisions. For example, if your business goal was to offer additional credit to customers, this might be a good segment to target.

4. Open the **Profile:job** window to view the lattice graph.



In the lattice graph, we see how certain variables are distributed in each of the segments. This graph enables further profiling and investigating of the customer base. This new information, in turn, enables more productive decision making.

For example, Segment 3 contains only a small number of unemployed customers. This information agrees with the idea that this segment contains more customers to target for credit. In contrast, Segment 4, a smaller segment, contains a larger number of unemployed customers. This segment might not be effective to target in certain business decisions. Again, information that is gained through profiling can be used to generate or reinforce business decisions regarding customer segments.

Lattice graphs might also reveal new or unusual information. Notice that Segment 4 contains a smaller number of customers who do not have a registered phone than Segment 2 contains. Also, notice the variable that is named Employed is displayed in the lattice for segment 4. This signifies the variable had some importance in generating Segment 4. Also note, this segment contains workers who have the longest duration of employment in their present position.

In contrast, Segment 2 seems to contain a far fewer number of customers who have registered phones than Segment 4. Also, the variable that is named Employed had less worth in creating Segment 2. The amount of variable worth is noted by the fact that the variable does not appear in the lattice graph of a segment. This type of information requires further investigation into the characteristics of the segments. The profiling tools in the Segment Profile Node enable you to complete more in-depth profiling of customer segments to aid in more effective decision making.

Variable Layout for SAMPSIO.DMAGECR

- **CHECKING**: Status of existing checking account
  - 2: 0 to <200 DM
  - 3: >=200 DM/ salary assignments for at least 1 year
  - 4: no checking account
- **DURATION**: Duration in months

- **HISTORY**: Credit history
  - 0: no credits taken/all credits paid back duly
  - 1: all credits at this bank paid back duly
  - 2: existing credits paid back duly till now
  - 3: delay in paying off in the past
  - 4: critical account/other credits existing (not at this bank)
- **PURPOSE**: Purpose
  - 0: car (new)
  - 1: car (used)
  - 2: furniture/equipment
  - 3: radio/television
  - 4: domestic appliances
  - 5: repairs
  - 6: education
  - 7: vacation
  - 8: retraining
  - 9: business
  - X: others
- **AMOUNT**: Credit amount
- **SAVINGS**: Savings account/bonds
  - 1: < 100 DM
  - 2: 100 to < 500 DM
  - 3: 500 to < 1000 DM
  - 4: >= 1000 DM
  - 5: unknown/no savings account
- **EMPLOYED**: Present employment since
  - 1: unemployed
  - 2: < 1 year
  - 3: 1 to < 4 years
  - 4: 4 to < 7 years
  - 5: >= 7 years
- **INSTALLP**: Installment rate in percentage of disposable income
- **MARITAL**: Personal status and gender
  - 1: male: divorced/separated
  - 2: female: divorced/separated/married
  - 3: male: single
  - 4: male: married/widowed

  - 5: female: single
- **COAPP**: Other debtors/guarantors
  - 1: none
  - 2: co-applicant
  - 3: guarantor
- **RESIDENT**: Date beginning permanent residence
- **PROPERTY**: Property
  - 1: real estate
  - 2: if not 1: building society savings agreement/life insurance
  - 3: if not 1/2: car or other, not in attribute 6
  - 4: unknown/no property
- **AGE**: Age in years
- **OTHER**: Other installment plans
  - 1: bank
  - 2: stores
  - 3: none
- **HOUSING**: Housing
  - 1: rent
  - 2: own
  - 3: for free
- **EXISTCR**: Number of existing credits at this bank
- **JOB**: Job
  - 1: unemployed/unskilled - nonresident
  - 2: unskilled - resident
  - 3: skilled employee/official
  - 4: management/self-employed/highly qualified employee/officer
- **DEPENDS**: Number of dependents
- **TELEPHONE**: Telephone
  - 1: none
  - 2: yes, registered under the customer's name
- **FOREIGN**: foreign worker
  - 1: yes
  - 2: no
- **GOOD_BAD**: Good or Bad Credit rating
  - 0: bad
  - 1: good

Part 15

Node Reference: Utility Nodes

Chapter 70

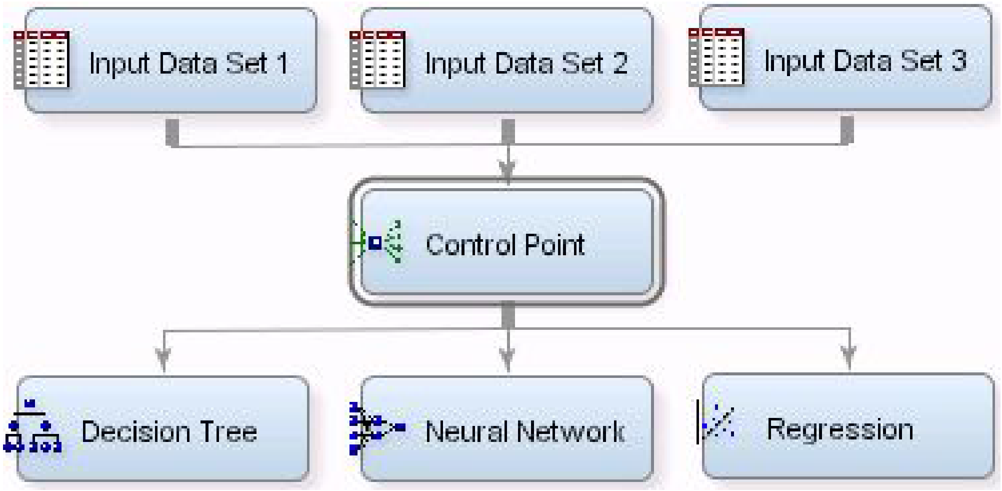
Control Point Node

Control Point Node

Overview of the Control Point Node

Use the Control Point node to establish a control point within process flow diagrams. A control point simplifies distributing the connections between process flow steps that have multiple interconnected nodes. The Control Point node can reduce the number of connections that are made.

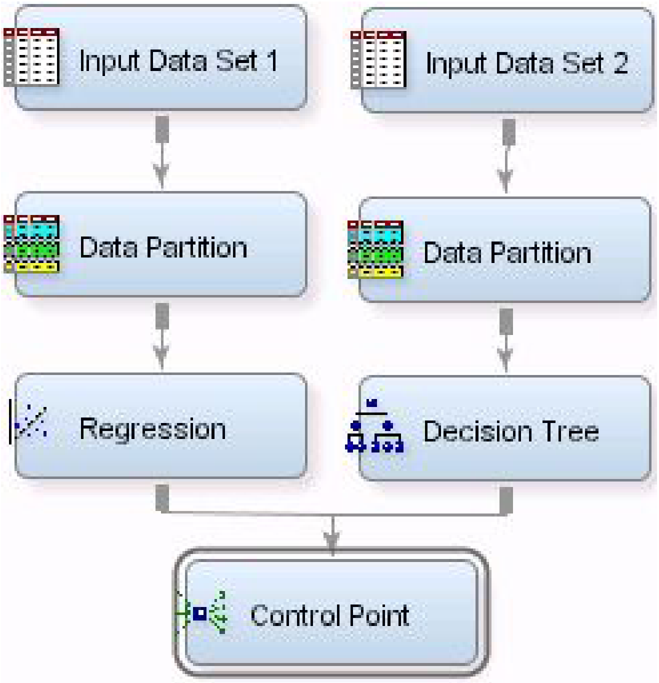
For example, suppose three Input Data nodes are to be connected to three modeling nodes. If no Control Point node is used, then you need nine connections to connect all of the Input Data nodes to all of the modeling nodes. However, if a Control Point node is used, you need only six connections as shown in the following graphic:



Updating and Running Diagrams with the Control Point Node

Running a process flow diagram from the Control Point node will run or update all preceding paths.

Consider the following process flow diagram:

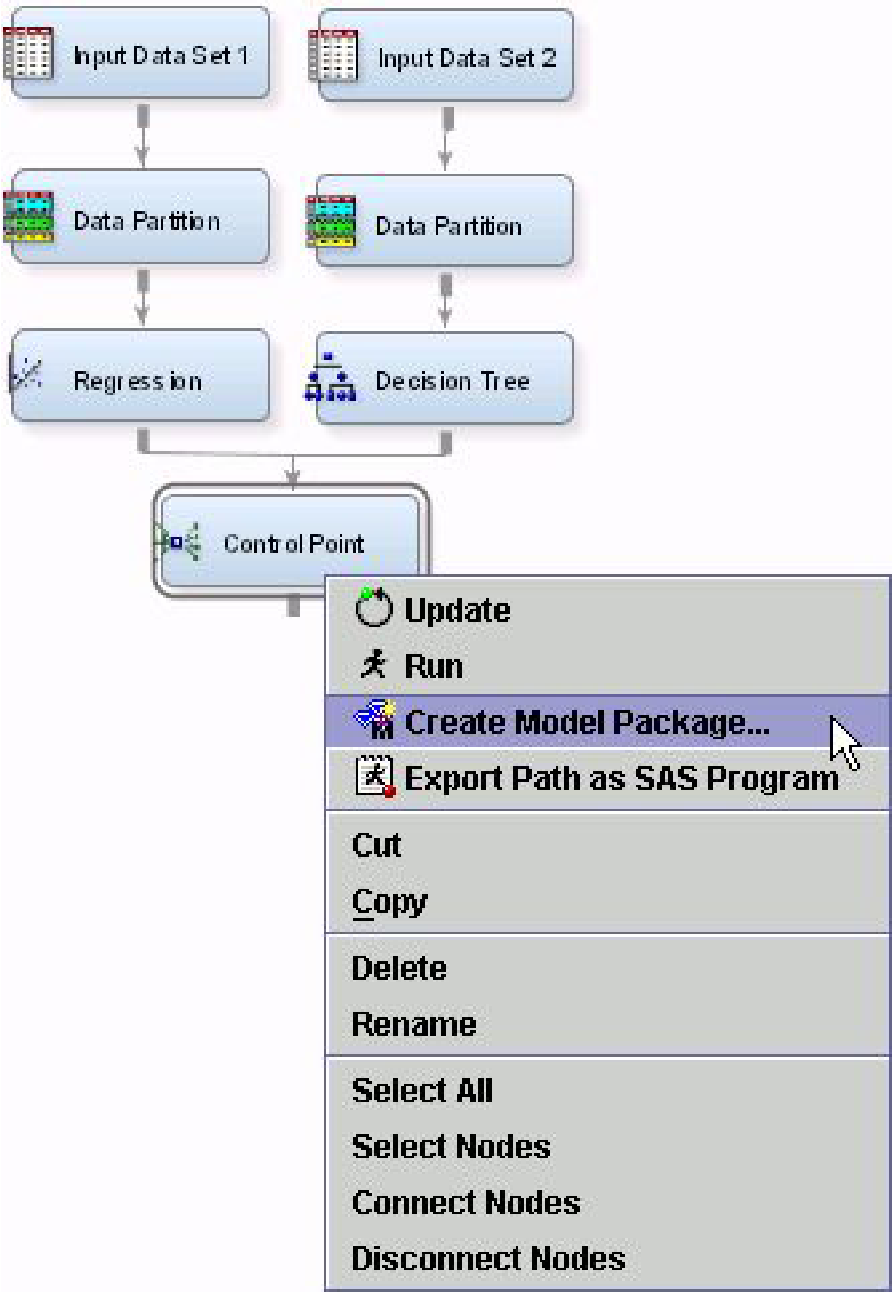


In this diagram, right-clicking the Control Point node and selecting Run will cause both of the parallel flows to execute. If the flows have been run previously, they will update all nodes preceding Control Point.

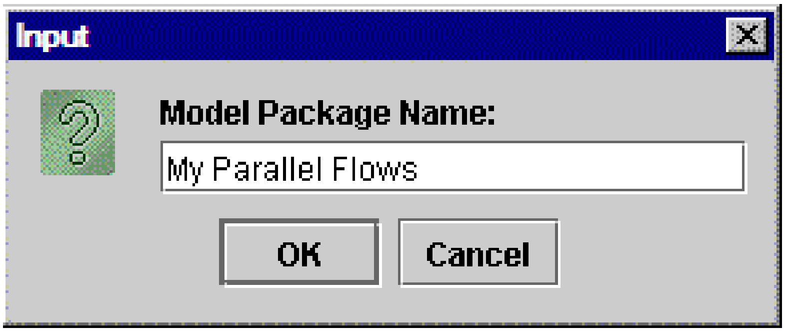
Creating a Model Package from the Control Point Node

You can use the Control Point node to create multiple Model Packages when you have parallel flows.

Consider the process flow diagram that was just run in the preceding section. Right-clicking the Control Point and selecting Create Model Package results in two Model Packages, one for the Regression flow and one for the Decision Tree flow:

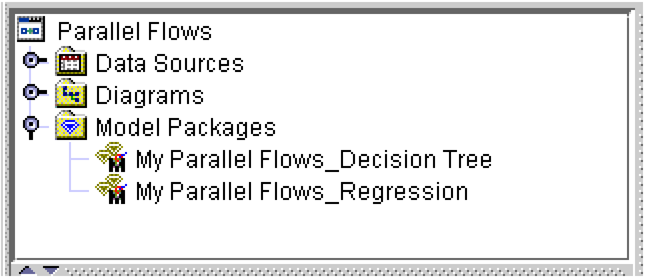


Save the package and name it. This example uses the name My Parallel Flows.



Model Package Name: My Parallel Flows

The Model Package creates two entries, as seen in the Project Navigator:



```
Parallel Flows
  Data Sources
  Diagrams
  Model Packages
    My Parallel Flows_Decision Tree
    My Parallel Flows_Regression
```

Using the Control Point node to generate Model Packages for parallel flows can save you time and effort.

Control Point Node Properties

Control Point Node General Properties

The following general properties are associated with the Control Point Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Control Point node added to a diagram will have a Node ID of CNTRL. The second Control Point node added to a diagram will have a Node ID of CNTRL2, and so on.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Control Point window. The Imported Data — Control Point window contains a list of the ports that provide data sources to the Control Point node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.

  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Control Point window. The Exported Data — Control Point window contains a list of the output data ports that the Control Point node creates data for when it runs.

Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Control Point Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Control Point Node Results

The Control Point node does not generate a results package.

Chapter 71

End Groups Node

End Groups Node

Overview of the End Groups Node

The **End Groups** node is located on the **Utility** tab of the SAS Enterprise Miner tools bar. The **End Groups** node is used only in conjunction with the **Start Groups** node. The **End Groups** node acts as a boundary marker that defines the end of group processing operations in a process flow. Group processing operations are performed on the portion of the process flow that exists between the **Start Groups** node and the **End Groups** node.

If the group processing function that is specified in the **Start Groups** node is stratified, bagging, or boosting, the **End Groups** node functions as a model node and presents the final aggregated model. (It is no longer necessary to use an **Ensemble** node to present the aggregated group processing model, as was required in SAS Enterprise Miner 4.3.) SAS Enterprise Miner tools that follow the **End Groups** node continue data mining processes normally.

End Groups Node Algorithm

The **End Groups** node identifies the boundaries of the group processing portion of a process flow and aggregates results for stratified, bagging, or boosting models. The **End Groups** node does not heuristically process group processing data.

Using the End Groups Node

The **End Groups** node can be used only in conjunction with the **Start Groups** node. The **End Groups** node must be preceded in the process flow diagram by the **Start Groups** node and at least one successor node to the **Start Groups** node.

It is possible to have more than one **Start Groups** and **End Groups** node pair in a process flow, as long as each group processing operation (within a **Start Groups** and **End Groups** node pair) is performed serially. In other words, each group processing operation must complete before the subsequent group processing operation begins. SAS Enterprise Miner cannot run more than one group processing operation at a time. As a result, you cannot nest a group processing operation within a group processing operation. The restriction also means that you also should not design process flows that perform group processing operations in competing parallel branches.

End Groups Node Properties

End Groups Node General Properties

The following general properties are associated with the **End Groups** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **End Groups** node that is added to a diagram will have a Node ID of EndGrp. The second **End Groups** node added to a diagram will have a Node ID of EndGrp2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — End Groups window. The Imported Data — End Groups window contains a list of the ports that provide data sources to the **End Groups** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — End Groups window. The Exported Data — End Groups window contains a list of the output data ports that the **End Groups** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

End Groups Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

End Groups Node Results

You can open the Results window of the **End Groups** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the **End Groups** node properties panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the **End Groups** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information about the data set submitted to the **End Groups** node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **End Groups** node run.
  - **Output** — the SAS output of the **End Groups** node run.
  - **Flow Code** — the SAS code used to produce the output that the **End Groups** node passes on to the next node in the process flow diagram.
  - **Group Log\Output** — the log for the group processing portion of the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was generated by the node.

- **Assessment**
  - **Classification Chart** — displays a histogram by classification and data role.
  - **Score Rankings Matrix** — displays the Score Rankings matrix plot. The score rankings matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the various models and the various data sets. Available assessment options in drop-down menu include the following:
    - Cumulative Lift
    - Lift
    - Gain
    - % Response
    - Cumulative % Response
    - % Captured Response
    - Cumulative % Captured Response
  - **Score Distribution** — displays the Score Distribution chart. The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis across the various bins in a lattice that is defined by the various models and the various data sets.
  - **Fit Statistics** — a table that contains fit statistics for predicted variables.
- **Group Processing**
  - **Summary** — displays a table that contains the Group Index, Group, and Frequency Count.

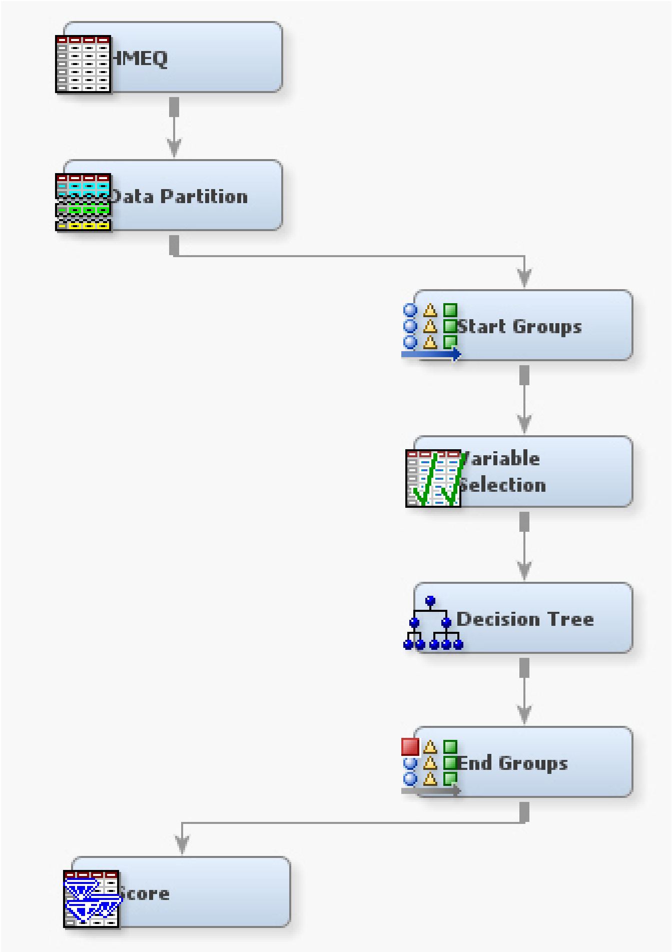
End Groups Node Example

The example below illustrates how the **End Groups** node indicates the end of the group processing portion of the process flow diagram.

The **Start Groups** node begins the group processing portion. The group processing portion of this process flow diagram is: **Start Groups** ⇨ **Variable Selection** ⇨ **Decision Tree** ⇨ **End Groups**.

The **Variable Selection** and **Decision Tree** nodes constitute the group processing portion of the process flow diagram below.

If the **Start Groups** node is configured to perform stratified, bagging, or boosting functions, then the **End Groups** node Results presents the aggregated model when the process flow diagram runs in its entirety.


HMEQ
Data Partition
Start Groups
Variable Selection
Decision Tree
End Groups
Score

Chapter 72
Metadata Node

Metadata Node

Overview of the Metadata Node

The Metadata node belongs to the Utility category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA). Use the Metadata node to modify metadata information in your process flow diagram. You can modify attributes such as variable roles, measurement levels, and order. You can also merge predecessor variables and modify data role and multiple roles in the Metadata node.

For example, it is common to generate a data set in the SAS Code node and then modify its metadata with the Metadata node. You cannot follow the SAS Code node with an Input Data node in your process flow diagram.

Metadata Node Properties

Metadata Node General Properties

The following general properties are associated with the Metadata Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Metadata node added to a diagram will have a Node ID of Meta. The second Metadata node added to a diagram will have a Node ID of Meta2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Metadata window. The Imported Data — Metadata window contains a list of the

ports that provide data sources to the Metadata node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Metadata window. The Exported Data — Metadata window contains a list of the output data ports that the Metadata node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Metadata Node Train Properties

The following train properties are associated with the Metadata node:

- **Import Selection** — Use the Import Selection property to specify the source to populate the import ports of the node. Select the ... button to the right of the Import Selection property to open the Import Selection Editor. The Import Selection editor displays two columns: Port and Data Set. There are five data port types: Train, Validate, Test, Score, and Transaction. The Data Set field is a drop-down list that contains the IDS designation for each data source that has a particular port type.
- **Summarize** — Use the Summarize property to specify to compute statistics for the active metadata. If the statistics already exist, then the statistics will be refreshed.
- **Advanced Advisor** — Indicates whether the node should refresh the exported metadata using the Advanced Advisor. When set to **Yes**, the Advanced Advisor is used to determine the level and role attributes of the variables in the data.

Metadata Node Train Properties: Rejected Variables

- **Hide Rejected Variables** — Set the Hide Rejected property to **Yes** if you want to drop rejected variables from your exported metadata. Rejected variables are "hidden" and will not appear in the exported metadata, but they are not dropped from exported data sets and views. The default setting for the Hide Rejected property is **No**.
- **Combine Rule** — Use the Combine Rule property to specify the rule for rejecting input variables based on multiple sources of metadata.
  - **None** — The role of input and rejected variables is based on the active metadata.

- **Any** — A variable is set to Rejected if it is rejected in at least one of the incoming metadata sources.
- **All** — A variable is rejected only if it is rejected in all of the incoming metadata sources.
- **Majority** — A variable is rejected if it is rejected in the majority of the incoming metadata sources. If there is a tie, the rejection is based on the active metadata source.

Metadata Node Train Properties: Variables

- **Variables** — Use the Variables property to specify how to use the variables in your data sources. The Metadata node recognizes five different types of data sources. The different types of data sources are determined by the Role property of each data source node.

  The five types are as follows:

  - **Train**
  - **Transaction**
  - **Validate**
  - **Test**
  - **Score**

  There are five corresponding properties in the Metadata node's Variables property group. Select the ... button to the right of any of the five properties to open a variables table for that data source type. For example, if you select the ... button to the right of the Train property, a variables table opens for the input data source that has a role of Train. When there are multiple data sources of the same type, the data set that is displayed in the variables table is determined by the value that you designate in the Metadata node's Import Selection property. In the table, you can perform the following metadata changes:

  - Set the Hide status for each variable to either Default, Yes, or No. The Default setting maintains the variable's previous Hide specification.
  - Set the New Role setting to change the role of a selected variable.
  - Specify a New Level setting if you want to change a variable's level. The available settings for the New Level column are Default, Unary, Binary, Nominal, Ordinal, and Interval. The Default setting maintains the variable's previous Level specification.
  - Specify a New Order setting if you want to change a variable's sort order. The available settings for the New Order column are Default, Ascending, Descending, Formatted Ascending, and Formatted Descending. The Default setting maintains the variable's previous Order specification.
  - Specify a New Report setting if you want to change a variable's Report specification. The available values for the New Report column are Default, Yes, and No. The Default setting maintains the variable's previous Report specification.

Metadata Node Status Properties

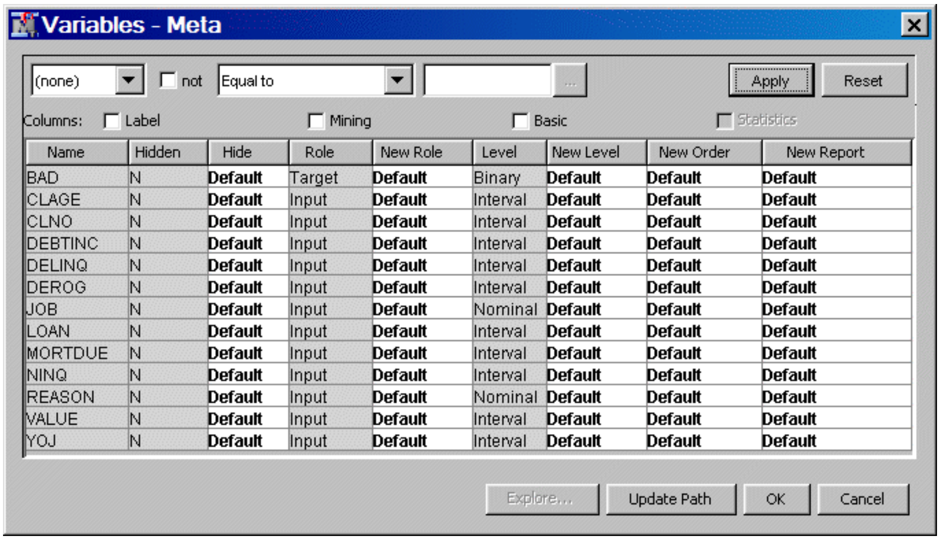
The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Metadata Node Variables Table

Use the table in the Variables — Meta window to change or specify metadata that pertains to data that has been exported by an Enterprise Miner node. You can specify new metadata information on a variable-by-variable basis if you wish. To open the Variables — Meta window, select the ... button to the right of the Variables property in the Properties Panel while the Metadata node is selected in the diagram workspace. The example below uses the SAMPSIO.HMEQ data set.



Name	Hidden	Hide	Role	New Role	Level	New Level	New Order	New Report
BAD	N	Default	Target	Default	Binary	Default	Default	Default
CLAGE	N	Default	Input	Default	Interval	Default	Default	Default
CLNO	N	Default	Input	Default	Interval	Default	Default	Default
DEBTINC	N	Default	Input	Default	Interval	Default	Default	Default
DELINQ	N	Default	Input	Default	Interval	Default	Default	Default
DEROG	N	Default	Input	Default	Interval	Default	Default	Default
JOB	N	Default	Input	Default	Nominal	Default	Default	Default
LOAN	N	Default	Input	Default	Interval	Default	Default	Default
MORTDUE	N	Default	Input	Default	Interval	Default	Default	Default
NINQ	N	Default	Input	Default	Interval	Default	Default	Default
REASON	N	Default	Input	Default	Nominal	Default	Default	Default
VALUE	N	Default	Input	Default	Interval	Default	Default	Default
YOJ	N	Default	Input	Default	Interval	Default	Default	Default

You can resize the columns in the Variables — Meta window to enhance readability. Click a column heading to toggle between ascending and descending column and table sorts. Information in cells that have a white background can be configured. Cells that have gray backgrounds are read-only. To modify the following properties by individual variable, you can use the selections that are in the white columns. Selections that are dimmed or unavailable are not suitable for the type of variable that has been selected.

- **Hide** — Use the Hide column to specify whether to hide the variable when exporting output data to successor nodes. The New Hide column permits values of Default, Yes, and No.
- **New Role** — Use the New Role column to specify a new variable role. If you select Default, you continue to use the variable role that was assigned to the variable in the predecessor node. For a detailed list of roles, see Model Roles and Levels of Variables.

- **New Level** — Use the New Level column to specify a new measurement level for class variables. If you select Default, you continue to maintain the variable Level setting that was used in the predecessor node. For a detailed list of levels, see Model Roles and Levels of Variables.
- **New Order** — Use the New Order column to specify the sort order for class variables.
  - **Ascending**
  - **Descending**
  - **Default**
  - **Formatted Ascending**
  - **Formatted Descending**
- **New Report** — Use the New Report column to specify whether a variable should be automatically included in reports such as the predictive model assessment decile and percentile tables.
  - **Default** — Use the default variable report settings that are passed to this node.
  - **Yes** — Override any previous report settings for this variable and set them to Yes.
  - **No** — Override any previous report settings for this variable and set them to No.

Metadata Node Results

After a successful node run, you can open the Results window of the Metadata node by right-clicking the node and selecting Results from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

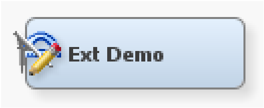
- **Properties**
  - **Settings** — displays a window with a read-only table of the Metadata node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Metadata node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that Enterprise Miner used to train the node. The Train Code property is, by default, dimmed and unavailable for the Metadata node.
  - **Notes** — displays the information typed by the user for the node.
- **SAS Results**
  - **Log** — the SAS log of the Metadata node run.
  - **Output** — The Output listing of the Metadata node contains a table of the variables that have modified attributes and a table of the metadata exported by the node for some of the attributes (name, role, level, creator, and label).
  - **Flow Code** — The Metadata node does not produce SAS flow code.
  - **Statistics Table** — The Metadata node does not produce a statistics table.

- **Scoring**
  - **SAS Code** — SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Metadata node does not generate PMML code.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Chapter 73

Ext Demo Node

Ext Demo Node



Overview of the Ext Demo Node

The **Ext Demo** node is on the **Utility** tab of the SAS Enterprise Miner tools bar. The **Ext Demo** node is designed to illustrate the various property types that can be implemented in SAS Enterprise Miner extension nodes. The properties of a SAS Enterprise Miner node enable users to pass arguments to the node's underlying SAS program. By choosing an appropriate property type, an extension node developer can control how information about the node's arguments are presented to the user and place restrictions on the values of the arguments. The Ext Demo node's results also provides examples of the various types of graphs that can be generated by an extension node using the %EM_REPORT macro.

Ext Demo Node Properties

Ext Demo Node General Properties

The following general properties are associated with the **Ext Demo** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Ext Demo node added to a diagram will have a Node ID of EXT. The second Ext Demo node added to a diagram will have a Node ID of EXT2, and so on.
- **Imported Data** — The Imported Data property accesses the Imported Data — Ext Demo window. The Imported Data — Ext Demo window contains a list of the ports

that provide data sources to the **Cluster** node. Select the ... button to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — accesses the Exported Data — Ext Demo window. The Exported Data — Ext Demo window contains a list of the output data ports that the **Cluster** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Ext Demo Node Train Properties: Cell Editors

- **Boolean** — an example of a Boolean Property element that enables the user to assign a value of Yes or No to the property.
- **String** — an example of a String Property element that enables the user to assign a character string to the property by entering the string into a text box.
- **Choice List** — an example of a String Property element that enables the user to assign a character string to the property by selecting a string from a predetermined choice list. The choice list is implemented using a ChoiceList control.
- **Integer** — an example of an integer Property element that enables the user to assign an integer value to the property by entering the integer value into a text box. If a user types in a non-integer value, the property value is set to missing.
- **Integer with Range Control** — an example of an integer Property element that enables the user to assign a restricted integer value to the property by entering the integer value into a text box. The range is determined by the min and max attributes of the Property element. If a user types in a value that is not an integer or falls outside of the permitted range, the property value reverts back to the property's last valid value.
- **Double** — an example of a double Property element that enables the user to assign an unrestricted real number to the property by entering a real number value into a text box. If a user types in a non-numeric value the property's value is set to missing.
- **Double with Range Control** — an example of a double Property element that enables the user to assign a restricted real number value to the property by entering a real number value into a text box. The range is determined by the min and max

attributes of the Property element. If a user types in a value that is not a real number or falls outside of the permitted range, the property value reverts back to the property's last valid value.

Ext Demo Node Train Properties: Table Editors

- **Table Edit Create Control Example** — an example of a String Property with a Table Editor Control that enables you to add or remove rows.
- **Table Editor Control Example** — an example of a String Property with a Table Editor Control. This configuration enables the user to edit or display character or numeric columns.
- **Table Editor with Choices** — an example of a String Property with a Table Editor Control and a ChoiceList Control. This configuration enables you to restrict the values of character columns to a predetermined list of values.
- **Table Editor with Dynamic Choices** — an example of a String Property with a Table Editor Control and a DynamicChoiceList Control. This configuration enables you to restrict the values of character columns to values that are dynamically generated by the server.
- **Table Editor with Restricted Choices** — an example of a String Property with a Table Editor Control and a DynamicChoiceList Control. This configuration enables you to restrict the values of character columns to values that are dynamically generated by the server. In this configuration, the Table Editor Control has an attribute that enables the choice lists to differ, depending on the value of another variable.
- **Ordering Editor** — an example of a String Property with a Table Editor Control. In this example, the Table Editor Control has an additional isOrderingEditor attribute that distinguishes it from the basic Table Editor Control. This configuration enables the user to change the order of the rows for a table.
- **Variables** — an example of a String Property element with a Dialog Control. This Property element configuration provides access to the variables exported by a predecessor Data Source node. It is common to all SAS distributed nodes.
- **SASTABLE Control** — an example of a String Property element with a SASTABLE Control. When the user clicks on the ... icon a Select a SAS Table window is displayed and the user is permitted to select a SAS data set from the SAS libraries that are accessible by SAS Enterprise Miner.
- **Text Editor** — an example of a String Property with a Dialog Control. A Property with this Control configuration enables the user to enter and modify text that is stored in an external file.
- **File Transfer** — an example of a File Transfer Control. A property with this control configuration enables the user to download an external file and then execute it with the proper program. Execution is performed automatically by the operating system.
- **Directory Selector** — an example of a Select Server Directory Control. A property with this control configuration enables the user to specify an external directory.
- **Model Selector** — an example of a Model Selector Control that enables the user to select a registered model. When a model is selected using this type of Control, the score code, score input variables, score output variables, target variables, training table, and fit statistics that are associated with the model are saved in the diagram folder and are associated with the node.
- **Register Model** — an example of a Model Registration Control that enables the user to and register a model.

Ext Demo Node Train Properties: Interaction Editor

- **Two-Factor** — an example of a String Property with a Dialog Control. A Property with this Control configuration allows the user to specify a two-factor interaction. An interaction editor Control has two attributes that determine the maximum number of effects that are allowed and whether main effects are allowed. In this example, the maximum number of effects is set to 2 and main effects are not allowed.
- **Terms** — an example of a String Property with a Dialog Control. A Property with this Control configuration allows the user to specify main effects and up to six-factor interactions. An interaction editor Control has two attributes that determine the maximum number of effects that are allowed and whether main effects are allowed. In this example, the maximum number of effects is set to 6 and main effects are allowed.
- **Text Editor with No Color Code** — an example of a String Property with a Dialog Control. A Property with this Control configuration enables the user to enter and modify plain text that is stored in an external file. An example of this property is the **Notes** property, which is available on all nodes.

Ext Demo Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Ext Demo Node Results

You can open the Results window of the **Ext Demo** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Ext Demo Node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Ext Demo node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information about the data set submitted to the **Ext Demo** node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.

  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Ext Demo node run.
  - **Output** — the SAS output of the Ext Demo node run.
  - **Flow Code** — the SAS code used to produce the output that the **Ext Demo** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the **Ext Demo** node does not generate SAS Code. The **SAS Code** menu item is dimmed and unavailable in the Ext Demo Results window.
  - **PMML Code** — the **Ext Demo** node does not generate PMML code.

The Ext Demo node results also include a collection of charts that can be generated using the %EM_REPORT macro.

These include the following:

- Bar Chart
  - Simple
  - Combo Choices
- Histogram
  - Simple
  - Combo Choices
- Line Plot
  - Simple
  - Overlay
  - Reference Lines
  - Combo Choices
  - Overlay Combo Choices
  - Two Y Axes
  - Two Y Axes Combo Choices
  - Line Band
  - Group
- Scatter Plot
  - Simple
  - Overlay
  - Combo Choices
  - Overlay Combo Choices
  - Group
- Pie
  - Simple

- Lattice
  - Simple Bar
  - Bar Combo Choices
  - Simple Histogram
  - Histogram Combo Choices
  - Simple Line Plot
  - Line Plot Overlay
  - Line Plot Reference Lines
  - Line Plot Combo Choices
  - Pie
- Box Plot
  - Grouped
- 3–D Graphs
  - Scatter Plot
  - Bar
  - Surface
- Data Specific
  - Dendrogram
  - Constellation: Link and Node Data
  - Constellation: Link Data

If you place an **`options mprint;`** statement in your project start code, the calls to %em_report are recorded in the Results log. You can also view the ExtDemo node's source code. It is stored in Sashelp.Emutil.Extdemo.source.

Chapter 74

Reporter Node

Reporter Node

Overview of the Reporter Node

The Reporter node is available from the Utility tab of the Enterprise Miner toolbar. The node uses SAS Output Delivery System (ODS) capability to create a single PDF or RTF file that contains information about the open process flow diagram. The PDF or RTF documents can be viewed and saved directly and are included in Enterprise Miner report package files.

The report contains a header that shows the Enterprise Miner settings, process flow diagram, and detailed information for each node. Based on the Nodes property setting, each node that is included in the open process flow diagram has a header, property settings and variable summary. Moreover, the report also includes results such as variable selection, model diagnostic tables, and plots from the Results browser. Score code, log, and output listing are not included in the report. Those are found in the Enterprise Miner package folder.

Reporter Node Properties

Reporter Node General Properties

The following general properties are associated with the Reporter Node:

- **Node ID** — the Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Reporter node that is added to a diagram will have a Node ID of Report. The second Reporter node added to a diagram will have a Node ID of Report2, and so on.

- **Imported Data** — the data set or data view that is imported into the Reporter node. Click the ... button to open the Imported Data — Reporter window. The Imported Data — Reporter window contains information on the imported data's port, source, table name, data role, and whether data exists.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — the data set or data view that is exported from the Reporter node. Click the button to open the Exported Data — Reporter window. The Exported Data - Reporter window contains a list of the exported data's port, table name, data role, and whether data exists. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Reporter Node Train Properties

The following train properties are associated with the Reporter node:

- **Document Format** — Use the Document Format property to specify the format of the generated document.

  The supported document formats are as follows:

  - **PDF** — Portable Document Format, opened by applications such as Adobe Acrobat. Many contemporary browsers also support PDF document display.
  - **RTF** — Rich Text Format, opened by applications such as email clients, simple text editors, and word processing applications such as Microsoft Word.

  Note: The software that is used to display PDF and RTF files on individual computers varies significantly. Some basic text editing utilities, such as Windows Wordpad, only support text display for RTF files. Charts and plots in the RTF output will not surface if you use a flat file utility as a viewer. To view the charts and plots that are embedded in your Reporter RTF document, open the output RTF file using MS Word or another full-featured word processing utility.

- **Style** — Use the Style property to control the overall look of the generated document. Each style produces the same content, but different visual appearance. The styles have different looks such as set of colors, fonts and line styles that integrate the graphics and tabular outputs into one presentation with a unified look. The example at the end of this section illustrates each style.

Choose from any of the following:

  - **Default** — uses the ODS Default style with a dark blue header and shade table background.
  - **Analysis** — uses the Analysis style.
  - **Journal** — uses the Journal style. The report creates gray-scale graphs and tables that are suitable for statistical journals, reports, and other publications that require black-and-white figures.
  - **Listing** — uses the Listing style. This is the Reporter node Style property default.
  - **Statistical** — uses the Statistical style by a blue header, black table text.

- **Nodes** — Use the Nodes property to specify which nodes the report should contain.
  - **Predecessor** — includes information of the immediate predecessor node.
  - **Path** — includes information on all the predecessor nodes in the process flow diagram training path.
  - **All** — includes information on all nodes that have been run in the process flow diagram.
  - **Summary** — generates a summary report for all nodes in the training path. If this option is selected, then it is possible to indicate specific report options using the Summary Report Options properties.
- **Show All** — Use the Show All property to specify the amount of data set information used for the generated document.
  - **Yes** — includes information on all data sets generated by each node.
  - **No** — includes information on all generated data sets that are automatically displayed in the Results browser.
- **Font Size** — Select the ... button to the right of the Font Size property to open the Font Size window. In the Font Size window, you are able to set the following properties:
  - **Data Font Size** — This property enables you to set the font size for the data that is published in your report.
  - **Header Font Size** — This property enables you to set the font size for the headers in your report.
  - **Title Font Size** — This property enables you to set the font size for the titles in your report.

Reporter Node Train Properties: Summary Report Options

- **Basic Reports** — Specify whether basic reports are generated. The basic reports include a model gains chart, variable importance table, and ROC chart. Note that the ROC Chart is available only for categorical targets.
- **Summarization** — This option prints the Model Summary Data section. This section contains a brief review of the input data source and the target variables.
- **Variable Ranking** — This option includes the variable rankings for all of the inputs that were selected by the model.

- **Classification Matrix** — This option provides the classification matrix in the summary report. This matrix is used to identify the percentage of true positives, true negatives, false positives, and false negatives.
- **Cross Tabs** — This option includes cross-tab reporting for nominal targets.
- **Lift Chart** — This option creates a graph of the cumulative lift chart in the summary report.
- **Fit Statistics** — This option displays a table with the statistics used to fit the model. The table contains both training data and validation data, assuming both are provided.
- **Model Comparison** — This option displays the model comparison table. This shows the information that was used by Enterprise Miner to select the champion model.

Reporter Node Report Properties

The Reporter node report properties are:

- **Save Report** — Use the Save Report property to specify the file location to save the generated document. Click the ... button to open the Export Data File window.
- **View Report** — Use the View Report property to open the generated document. Click the ... button to open the viewer associated with the Document Format property.

Note: Your computer must have a Windows file association between the PDF extension and the Adobe Acrobat Reader for Enterprise Miner to launch a viewer for your PDF report. If this file association does not exist, Enterprise Miner will not launch the PDF file in a browser. You computer must also have a Windows file association between RTF and MS Word. If the report opens with Wordpad, no graphs are displayed.

Reporter Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Reporter Node Results

After a successful node run, you can open the Results window of the Reporter node by selecting the **Results** button in the Run Status window, or by going to the diagram workspace, right-clicking the Reporter node and selecting Results from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the Reporter Node Properties Panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the Reporter node run. Information about whether the run completed successfully, the Run Start Time, Run Duration, and the Run ID are displayed in this window.
  - **Variables** — a read-only table of variable metadata information on the data set submitted to the Reporter Node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Reporter node's run.
  - **Output** — the SAS output of the Reporter node's run. Typical output includes header information such as user, date, and time. The Output window lists general reporting properties that are similar to the first page of the report. The output includes a variable summary showing the role, measurement level, and frequency count. The output also contains summary information for training output, score output, and report output.
  - **Flow Code** — the SAS code used to produce the output that the Reporter node passes to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the Reporter node does not generate SAS code.
  - **PMML Code** — the Reporter node does not generate PMML code.
- **Model**
  - **Report** — opens a Report window containing information about the name of the report that was created, the location where this report was created, and the size of the report.

EM_REPORT Global Macro Variables

You can customize the output of the Reporter node by setting the following global macro variables in your autoexec or startup code:

- EM_REPORT_TEXT_FONT — used as the GOPTIONS FTEXT= value; used in graphics output such as axes labels and graph titles. The GOPTIONS statement specifies graphics options that control the appearance of graphics elements by specifying characteristics such as default colors, fill patterns, fonts, or text height.
- EM_REPORT_NODE_FONT — used for the font within the nodes on PFD.
- EM_REPORT_HEADER_SIZE — set this to point sizes to controls the size of text used for data cells and headers for tables printed by ODS.

- EM_REPORT_TEXT_SIZE — used as the GOPTIONS text size in graphics output. Set this to point sizes to change your labels and axis within the graphs.
- EM_REPORT_TITLE_SIZE — used for the ODS text font size. This includes the text displayed at the top of each individual piece of the report. Set this to point sizes to change the Report title.
- EM_REPORT_ODS_FONT — the font used in the ODS template for all ODS output.

These variables are assigned as GLOBAL in the reporter code. You need to assign them in the startup code such as the example below:

```
%global EM_REPORT_TEXT_FONT EM_REPORT_HEADER_SIZE
    EM_REPORT_TEXT_SIZE EM_REPORT_TITLE_SIZE EM_REPORT_NODE_FONT;
%let EM_REPORT_TEXT_FONT = SWISSB;
%let EM_REPORT_HEADER_SIZE = 8;
%let EM_REPORT_TEXT_SIZE = 10;
%let EM_REPORT_TITLE_SIZE = 16;
%let EM_REPORT_NODE_FONT = COURIER;
```

Reporter Node Example

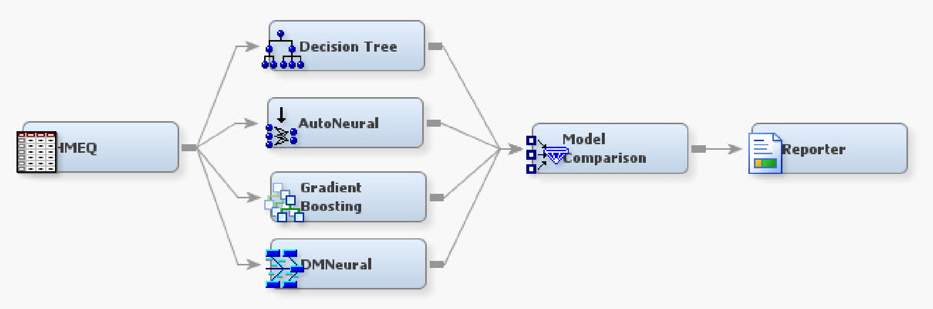
This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

In this example, you create four competitor models for the SAMPSIO.HMEQ data set, then use the Reporter node to view the model comparisons. The goal of this example is to display the different styles of output available in the Reporter node, not to analyze the accuracy of each model.

1. From the main menu, select **Help** ⇨ **Generate Sample Data Sources**. Select only the **Home Equity** data source. Click **OK**.
2. Drag the **Home Equity** data source from the Project Panel to the diagram workspace.
3. From the **Model** tab, add a **DMNeural**, **Gradient Boosting**, **Decision Tree**, and **AutoNeural** node. Connect the **Home Equity** node to each of these nodes.
4. From the **Assess** tab, add a **Model Comparison** node to your diagram workspace. Connect the four modelling nodes to the **Model Comparison** node.
5. From the **Utility** tab, add a **Reporter** node to your diagram workspace. Connect the **Model Comparison** node to the **Reporter** node.

   Select the Reporter node and set the following properties as follows:

   - Set Document Format to **PDF**.
   - Set Style to **Journal**.
   - Set Nodes to **All**.
   - Set Show All to **Yes**.

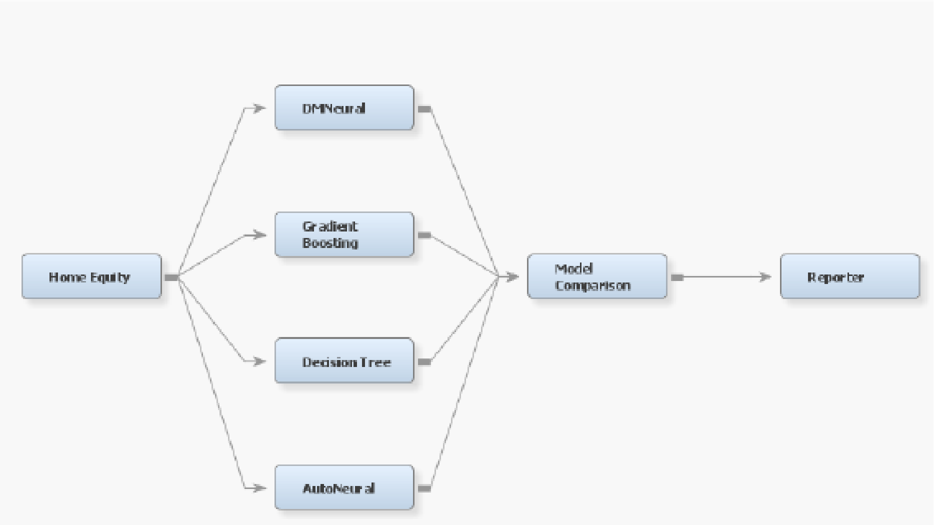


6. Right-click the Reporter node, and then select **Run** to run the process flow diagram. When the Run Status window indicates that the run is completed, click **Results** to open the Results window. Examine the output.

   Close the Results window.

Select the **Reporter** node in your diagram workspace. Click the ... button to the right of the **View Report** property to open Adobe Reader. The SAS Enterprise Miner Report shows a header with information about the user, date, project, diagram name, and properties of the Reporter node. A process flow diagram follows the header information.

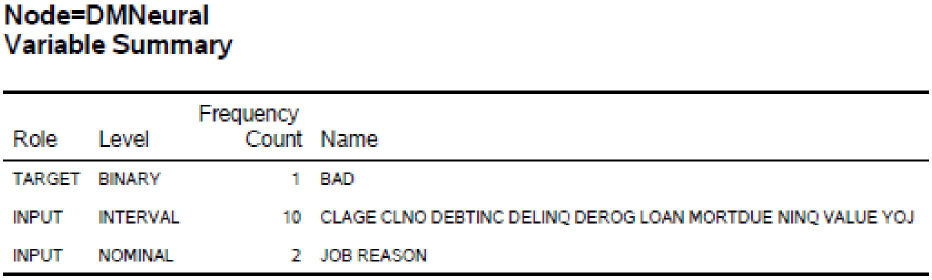
SAS Enterprise Miner Report
Process Flow Diagram



The report also contains a table of properties and values for each node, and variable summary information.

Node=DMNeural
Properties

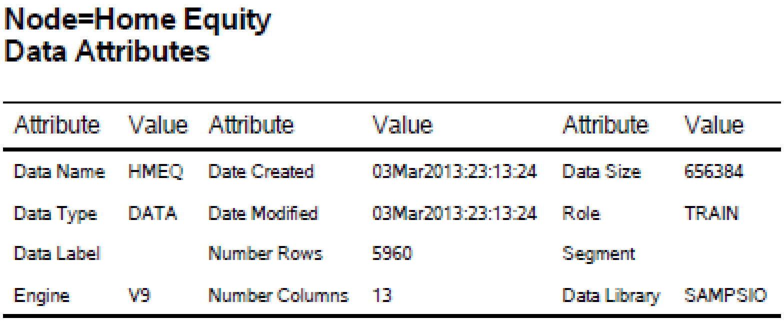
Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DMNeural		MaxFunction	500		PrintCovMatrix	N	
AbsGconv	0.0005		MaxIteration	200		PrintOptimizationHistory	N	
BinaryCutoff	0.5		MaxStage	3		PrintOption	DEFAULT	
Gconv	1E-8		MemSize	8		ScoreVarSuffix		
MaxComponent	3		ModelSelectionCriterion	DEFAULT		StatusMonitor	N	
MaxEigenVectors	400		OptimizationCriterion	SSE		StopR2	0.00005	



Node=DMNeural
Variable Summary

Role	Level	Frequency Count	Name
TARGET	BINARY	1	BAD
INPUT	INTERVAL	10	CLAGE CLNO DEBTINC DELINQ DEROG LOAN MORTDUE NINQ VALUE YOJ
INPUT	NOMINAL	2	JOB REASON

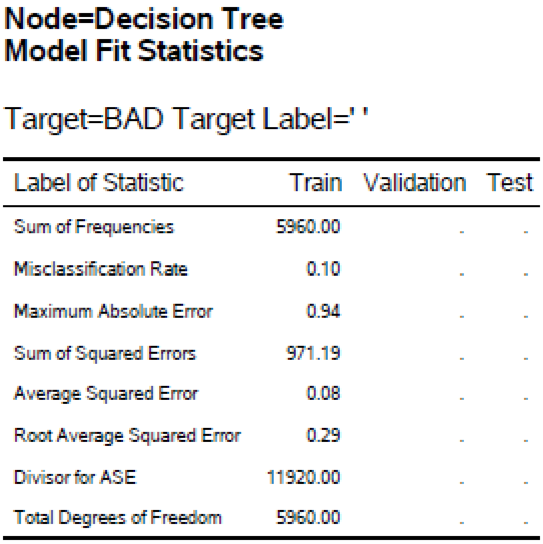
If the node is an Input Data Source, the report contains a table of data attributes.



Node=Home Equity
Data Attributes

Attribute	Value	Attribute	Value	Attribute	Value
Data Name	HMEQ	Date Created	03Mar2013:23:13:24	Data Size	656384
Data Type	DATA	Date Modified	03Mar2013:23:13:24	Role	TRAIN
Data Label		Number Rows	5960	Segment	
Engine	V9	Number Columns	13	Data Library	SAMPSIO

The report also contains all the data sets that are generated.

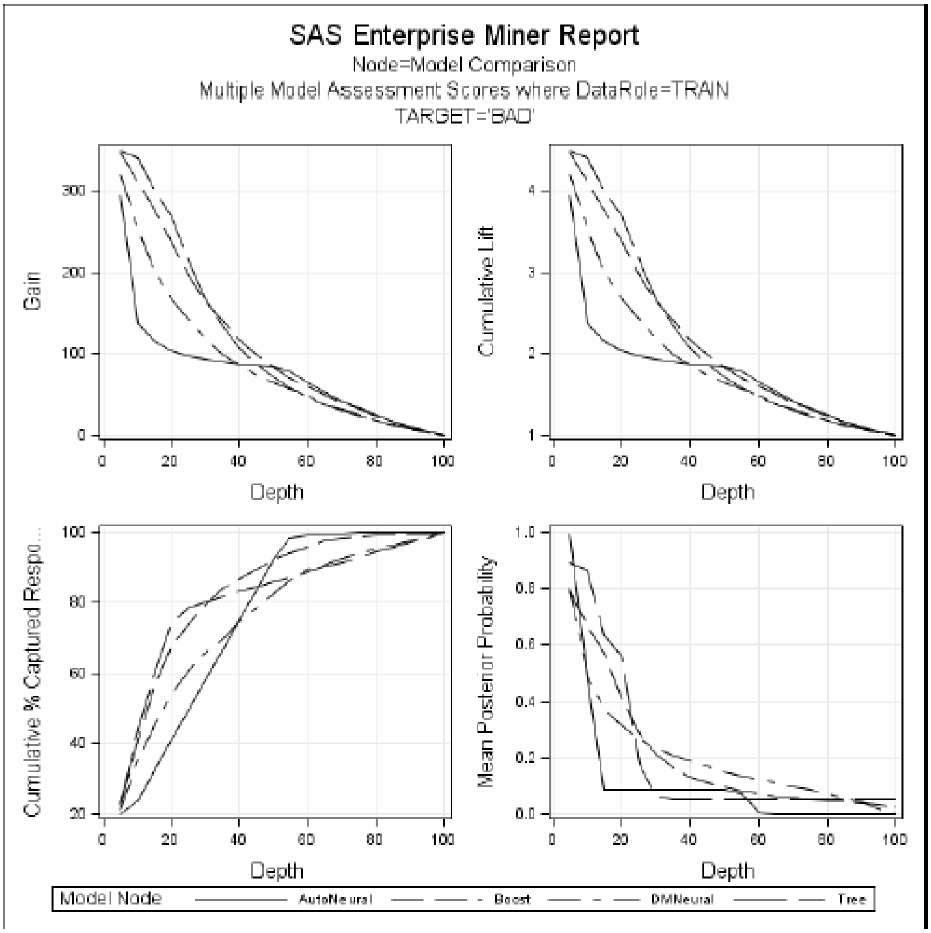


Node=Decision Tree
Model Fit Statistics

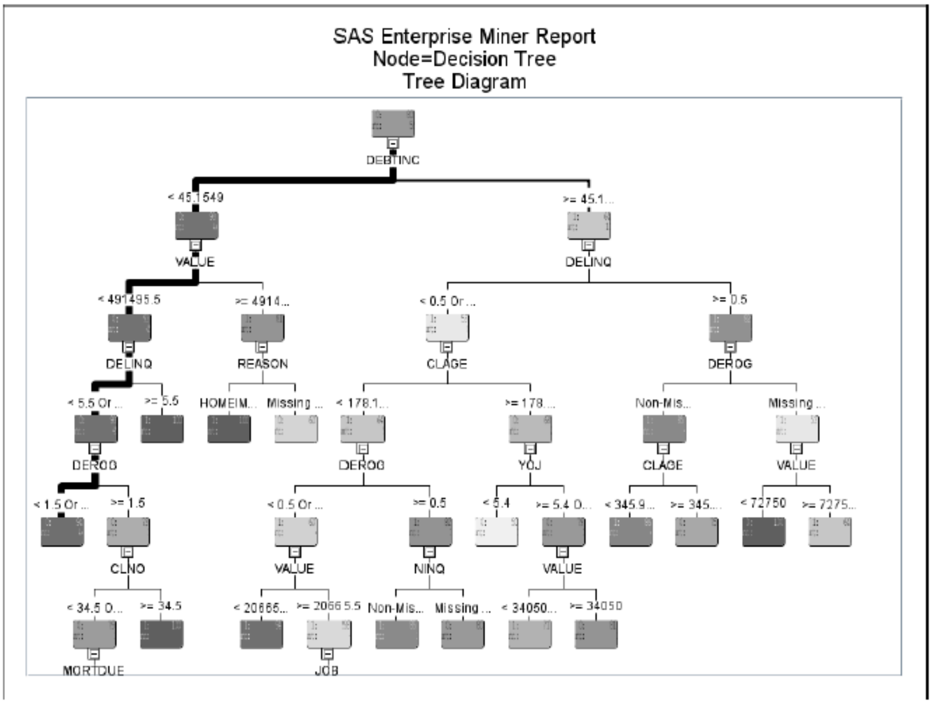
Target=BAD Target Label=' '

Label of Statistic	Train	Validation	Test
Sum of Frequencies	5960.00	.	.
Misclassification Rate	0.10	.	.
Maximum Absolute Error	0.94	.	.
Sum of Squared Errors	971.19	.	.
Average Squared Error	0.08	.	.
Root Average Squared Error	0.29	.	.
Divisor for ASE	11920.00	.	.
Total Degrees of Freedom	5960.00	.	.

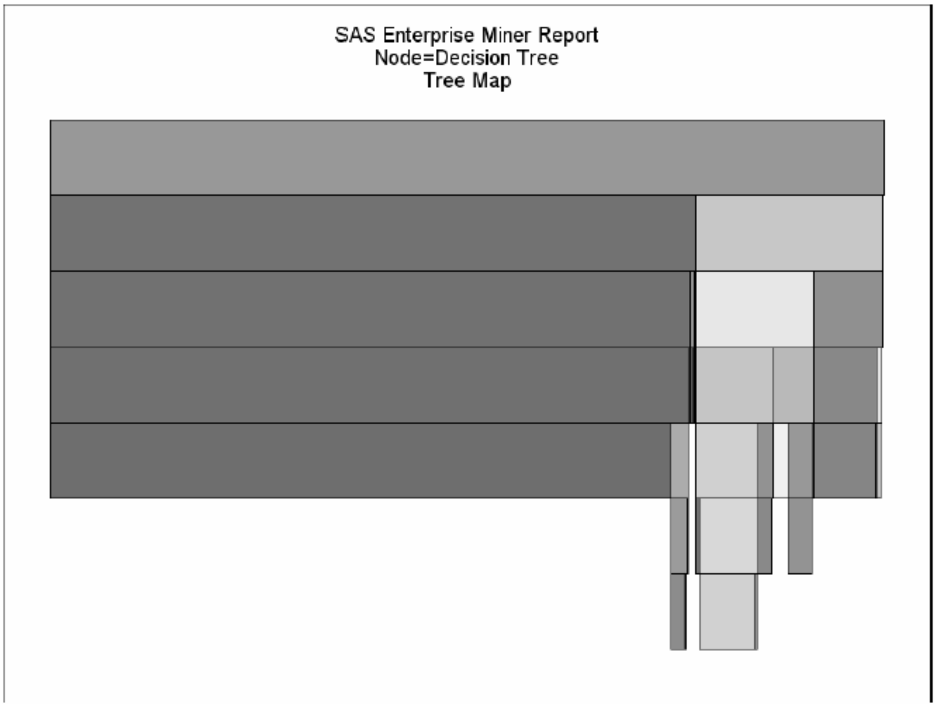
Some graphs, output, and tables in the Results window are included in the report. The following illustrates a comparison report.



The following illustrates a tree diagram:



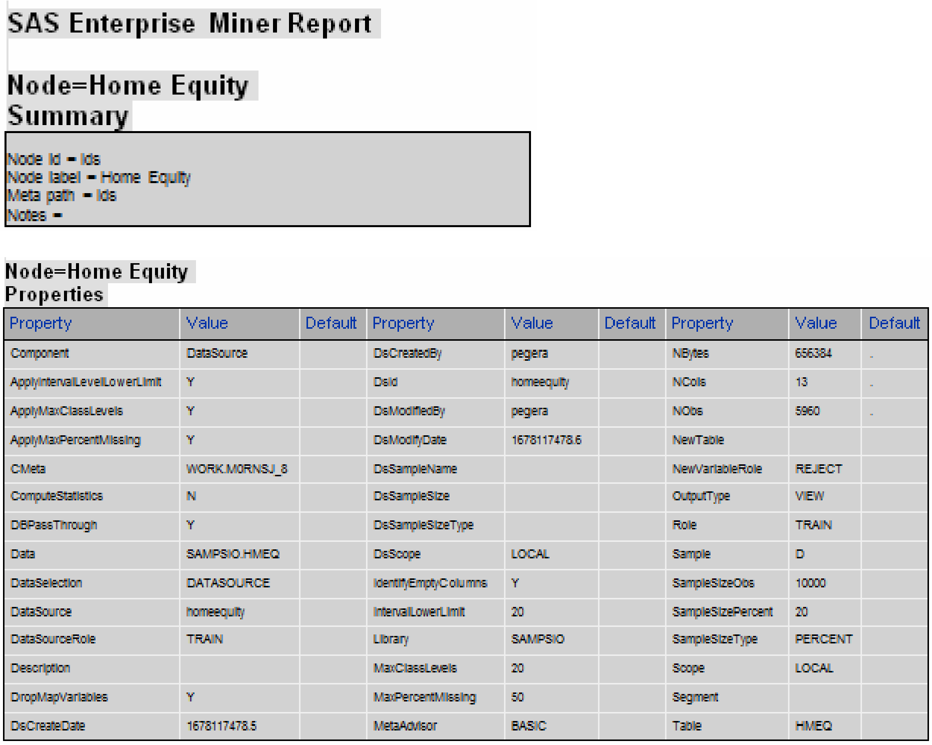
The following illustrates an icicle plot:



Close the Adobe Reader window. Select the **Reporter** node on your process flow diagram. Click the ... button to the right of the **Save Report** property to open the Export Data File window. Specify a location on the **Save in** field and type a filename on the **File name** field, and then click **Save**.

The following are RTF examples that illustrate the available Style properties of default, analysis, journal, listing, and statistical:

- Default



SAS Enterprise Miner Report

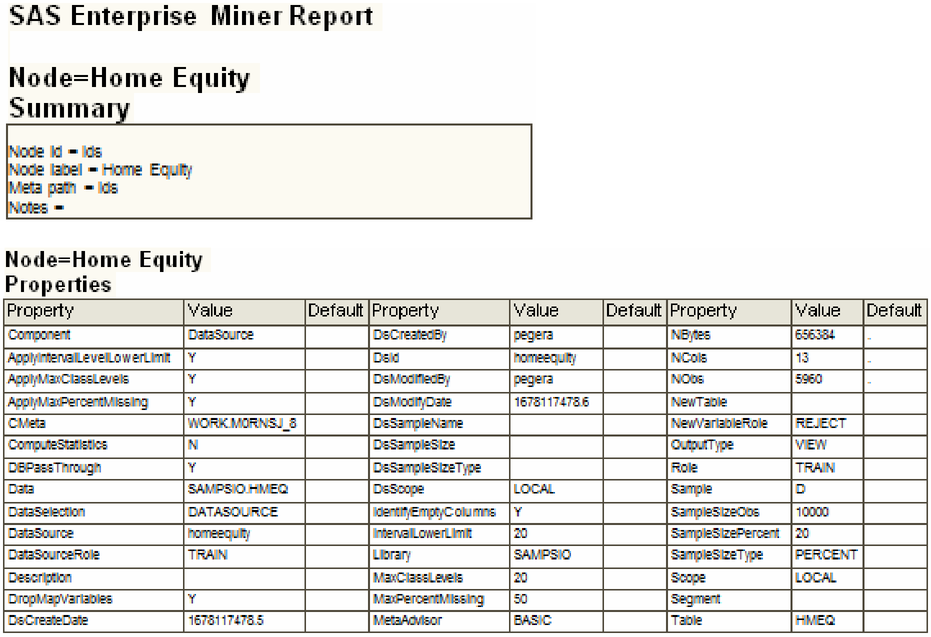
Node=Home Equity
Summary

Node Id = Ids
Node label = Home Equity
Meta path = Ids
Notes =

Node=Home Equity
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DsCreatedBy	pegera		NBytes	656384	.
ApplyIntervalLevelLowerLimit	Y		DsId	homeequity		NCols	13	.
ApplyMaxClassLevels	Y		DsModifiedBy	pegera		NObs	5960	.
ApplyMaxPercentMissing	Y		DsModifyDate	1678117478.6		NewTable		
CMeta	WORK.M0RNSJ_8		DsSampleName			NewVariableRole	REJECT	
ComputeStatistics	N		DsSampleSize			OutputType	VIEW	
DBPassThrough	Y		DsSampleSizeType			Role	TRAIN	
Data	SAMPSIO.HMEQ		DsScope	LOCAL		Sample	D	
DataSelection	DATASOURCE		IdentifyEmptyColumns	Y		SampleSizeObs	10000	
DataSource	homeequity		IntervalLowerLimit	20		SampleSizePercent	20	
DataSourceRole	TRAIN		Library	SAMPSIO		SampleSizeType	PERCENT	
Description			MaxClassLevels	20		Scope	LOCAL	
DropMapVariables	Y		MaxPercentMissing	50		Segment		
DsCreateDate	1678117478.5		MetaAdvisor	BASIC		Table	HMEQ	

- Analysis



SAS Enterprise Miner Report

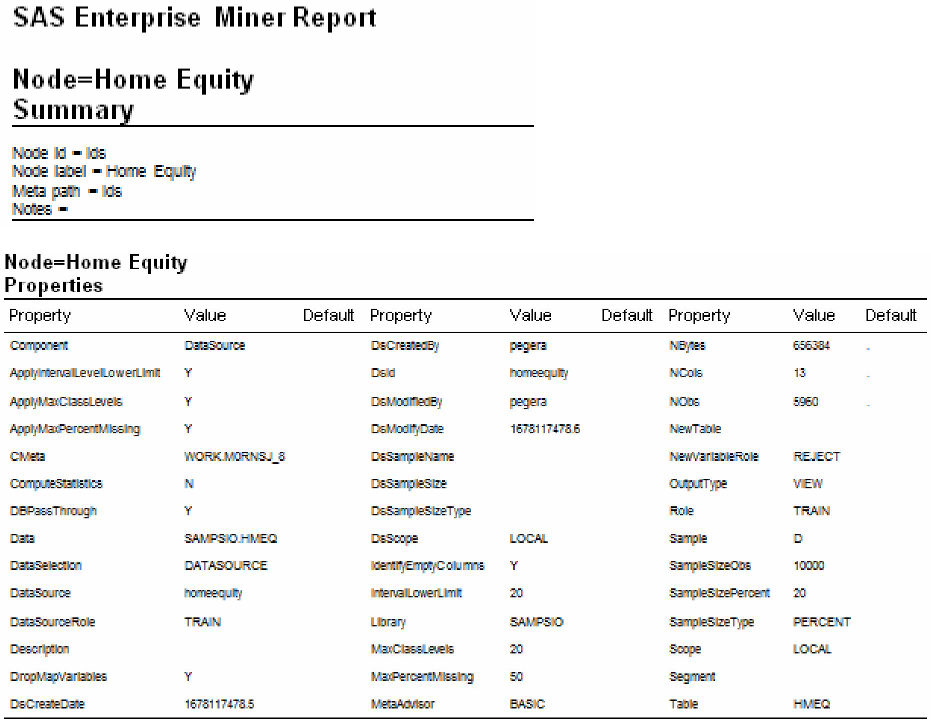
Node=Home Equity
Summary

Node Id = Ids
Node label = Home Equity
Meta path = Ids
Notes =

Node=Home Equity
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DsCreatedBy	pegera		NBytes	656384	.
ApplyIntervalLevelLowerLimit	Y		DsId	homeequity		NCols	13	.
ApplyMaxClassLevels	Y		DsModifiedBy	pegera		NObs	5960	.
ApplyMaxPercentMissing	Y		DsModifyDate	1678117478.6		NewTable		
CMeta	WORK.M0RNSJ_8		DsSampleName			NewVariableRole	REJECT	
ComputeStatistics	N		DsSampleSize			OutputType	VIEW	
DBPassThrough	Y		DsSampleSizeType			Role	TRAIN	
Data	SAMPSIO.HMEQ		DsScope	LOCAL		Sample	D	
DataSelection	DATASOURCE		IdentifyEmptyColumns	Y		SampleSizeObs	10000	
DataSource	homeequity		IntervalLowerLimit	20		SampleSizePercent	20	
DataSourceRole	TRAIN		Library	SAMPSIO		SampleSizeType	PERCENT	
Description			MaxClassLevels	20		Scope	LOCAL	
DropMapVariables	Y		MaxPercentMissing	50		Segment		
DsCreateDate	1678117478.5		MetaAdvisor	BASIC		Table	HMEQ	

- Journal



SAS Enterprise Miner Report

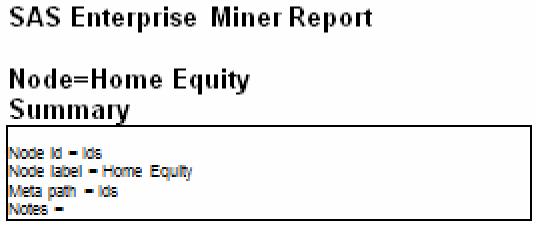
Node=Home Equity
Summary

Node Id = Ids
Node label = Home Equity
Meta path = Ids
Notes =

Node=Home Equity
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DsCreatedBy	pegera		NBytes	656384	.
ApplyIntervalLevelLowerLimit	Y		DsId	homeequity		NCols	13	.
ApplyMaxClassLevels	Y		DsModifiedBy	pegera		NObs	5960	.
ApplyMaxPercentMissing	Y		DsModifyDate	1678117478.6		NewTable		
CMeta	WORK.M0RNSJ_8		DsSampleName			NewVariableRole	REJECT	
ComputeStatistics	N		DsSampleSize			OutputType	VIEW	
DBPassThrough	Y		DsSampleSizeType			Role	TRAIN	
Data	SAMPSIO.HMEQ		DsScope	LOCAL		Sample	D	
DataSelection	DATASOURCE		IdentifyEmptyColumns	Y		SampleSizeObs	10000	
DataSource	homeequity		IntervalLowerLimit	20		SampleSizePercent	20	
DataSourceRole	TRAIN		Library	SAMPSIO		SampleSizeType	PERCENT	
Description			MaxClassLevels	20		Scope	LOCAL	
DropMapVariables	Y		MaxPercentMissing	50		Segment		
DsCreateDate	1678117478.5		MetaAdvisor	BASIC		Table	HMEQ	

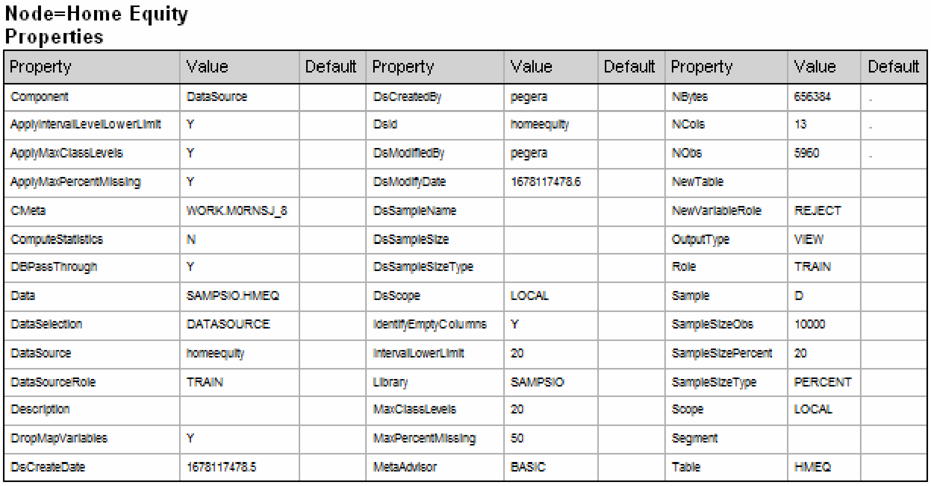
- Listing



SAS Enterprise Miner Report

Node=Home Equity
Summary

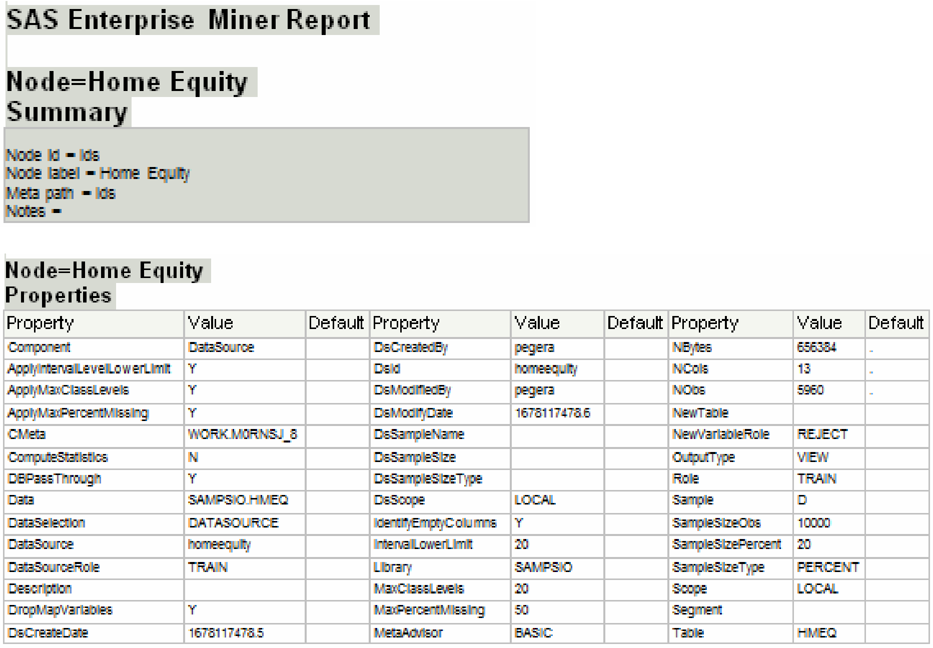
Node Id = Ids
Node label = Home Equity
Meta path = Ids
Notes =



Node=Home Equity
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DsCreatedBy	pegera		NBytes	656384	.
ApplyIntervalLevelLowerLimit	Y		DsId	homeequity		NCols	13	.
ApplyMaxClassLevels	Y		DsModifiedBy	pegera		NObs	5960	.
ApplyMaxPercentMissing	Y		DsModifyDate	1678117478.6		NewTable		
CMeta	WORK.M0RNSJ_8		DsSampleName			NewVariableRole	REJECT	
ComputeStatistics	N		DsSampleSize			OutputType	VIEW	
DBPassThrough	Y		DsSampleSizeType			Role	TRAIN	
Data	SAMPSIO.HMEQ		DsScope	LOCAL		Sample	D	
DataSelection	DATASOURCE		IdentifyEmptyColumns	Y		SampleSizeObs	10000	
DataSource	homeequity		IntervalLowerLimit	20		SampleSizePercent	20	
DataSourceRole	TRAIN		Library	SAMPSIO		SampleSizeType	PERCENT	
Description			MaxClassLevels	20		Scope	LOCAL	
DropMapVariables	Y		MaxPercentMissing	50		Segment		
DsCreateDate	1678117478.5		MetaAdvisor	BASIC		Table	HMEQ	

- Statistical



SAS Enterprise Miner Report

Node=Home Equity
Summary

Node Id = Ids
Node label = Home Equity
Meta path = Ids
Notes =

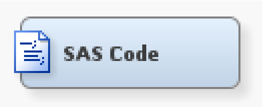
Node=Home Equity
Properties

Property	Value	Default	Property	Value	Default	Property	Value	Default
Component	DataSource		DsCreatedBy	pegera		NBytes	656384	.
ApplyIntervalLevelLowerLimit	Y		DsId	homeequity		NCols	13	.
ApplyMaxClassLevels	Y		DsModifiedBy	pegera		NObs	5960	.
ApplyMaxPercentMissing	Y		DsModifyDate	1678117478.6		NewTable		
CMeta	WORK.M0RNSJ_8		DsSampleName			NewVariableRole	REJECT	
ComputeStatistics	N		DsSampleSize			OutputType	VIEW	
DBPassThrough	Y		DsSampleSizeType			Role	TRAIN	
Data	SAMPSIO.HMEQ		DsScope	LOCAL		Sample	D	
DataSelection	DATASOURCE		IdentifyEmptyColumns	Y		SampleSizeObs	10000	
DataSource	homeequity		IntervalLowerLimit	20		SampleSizePercent	20	
DataSourceRole	TRAIN		Library	SAMPSIO		SampleSizeType	PERCENT	
Description			MaxClassLevels	20		Scope	LOCAL	
DropMapVariables	Y		MaxPercentMissing	50		Segment		
DsCreateDate	1678117478.5		MetaAdvisor	BASIC		Table	HMEQ	

Chapter 75

SAS Code Node

SAS Code Node



Overview of the SAS Code Node

The SAS Code node enables you to incorporate new or existing SAS code into process flow diagrams that were developed using SAS Enterprise Miner. The SAS Code node extends the functionality of SAS Enterprise Miner by making other SAS System procedures available for use in your data mining analysis. You can also write SAS DATA steps to create customized scoring code, conditionally process data, or manipulate existing data sets. The **SAS Code** node is also useful for building predictive models, formatting SAS output, defining table and plot views in the user interface, and for modifying variables metadata. The **SAS Code** node can be placed at any location within a SAS Enterprise Miner process flow diagram. By default, the **SAS Code** node does not require data. The exported data that is produced by a successful **SAS Code** node run can be used by subsequent nodes in a process flow diagram.

SAS Code Node Properties

SAS Code Node General Properties

The following general properties are associated with the **SAS Code** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type.
- **Imported Data** — Select the ... button to open a table of SAS data sets that are imported into the **SAS Code** node.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — Select the ... button to open a table of SAS data sets that are exported data by the **SAS Code** node.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

SAS Code Node Train Properties

The following train properties are associated with the **SAS Code** node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **SAS Code** node. Select the ... button to open a window that contains the variables table. You can set the Use and Report status for individual variables, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distributions. You can apply a filter based on the variable metadata column values so that only a subset of the variables is displayed in the table.
- **Code Editor** — Click the ... button to open the Code Editor. You can use the Code Editor to edit and submit code interactively at the same time that you view the SAS log and output listings. You can also run a process flow diagram path up to and including the **SAS Code** node and view the Results window without closing the programming interface. For more details, see the Code Editor section below.

- **Tool Type** — specifies the node type using the SAS Enterprise Miner SEMMA framework. Valid values are as follows:
  - **Sample**
  - **Explore**
  - **Modify**
  - **Model**
  - **Assess**
  - **Utility**

  The default setting for the Tool Type property is Utility. When the Tool Type is set to Model, SAS Enterprise Miner creates a target profile for the node if none exists. It also creates a report data model that is appropriate for a modeling node. Doing so enables SAS Enterprise Miner to automatically generate assessment results as long as certain variables are found in the scored data set (P_, I_, F_, R_ (depending on the target level)). See Predictive Modeling for more details about these variables and other essential information about modeling nodes.
- **Data Needed** — specifies whether the node needs at least one predecessor node. Valid values are **Yes** and **No**. The default setting for the Data Needed property is **No**.
- Rerun — specifies whether the node should rerun each time the process flow is executed, regardless of whether the node has run before or not. Valid values are **Yes** and **No**. The default setting for the Rerun property of the SAS Code node is **No**.
- **Use Priors** — specifies whether the posterior probability values are adjusted by the prior probability values. Valid values for the Use Priors property are **Yes** and **No**. The default setting for the Use Priors property is **Yes**.

SAS Code Node Score Properties

The following score properties are associated with the **SAS Code** node.

- **Advisor Type** — specifies the type of SAS Enterprise Miner input data advisor to be used to set the initial input variable measurement levels and roles. Valid values are as follows:
  - **Basic** — Any new variables created by the node will inherit Basic metadata attributes. These attributes include the following
    - character variables are assigned a Level of Nominal
    - numeric variables are assigned a Level of Interval
    - variables are assigned a Role of Input
  - **Advanced** — Variable distributions and variable attributes are used to determine the variable level and role attributes of newly created variables.

  The default setting for the Advisor Type property is Basic. You can also control the metadata programmatically by writing SAS code to the file CDELTA_TRAIN.sas. There is also a feature that permits a user to create a data set that predefines metadata for specific variable names. This data set must be named COLUMNMETA, and it must be stored in the EMMETA library.
- **Publish Code** — specifies the file that should be used when collecting the scoring code to be exported. Valid values are as follows:
  - **Flow** — Flow scoring code is used to score SAS data tables inside the process flow diagram. The scoring code is written to EMFLOWSCORE.sas.

    - **Publish** — Publish scoring code is used to publish the SAS Enterprise Miner model to a scoring system outside the process flow diagram. The scoring code is written to EMPUBLISHSCORE.sas.

  The default setting of the Publish Code property is Publish. It is possible to have scoring code that is used within the process flow (Flow code) and different code that is used to score external data (Publish code). For example, when generating Flow code for modeling nodes, the scoring code can reference the observed target variable, and you can generate residuals from a statistical model. Since Publish code is destined to be used to score external data where the target variable is unobserved, residuals from a statistical model cannot be generated.

- **Code Format** — specifies the format of the score code to be generated. Valid values are as follows:
    - **DATA step** — The score code contains only the DATA step statement that is accumulated when the flow score code is generated.
    - **Other** — The score code contains statements other than DATA step statements, such as PROC step statements.

  The default setting for the Code Format property is DATA step. It is necessary to make the distinction because nodes such as the **Ensemble** node and the **Score** node collect score code from every predecessor node in the process flow diagram. If all of the predecessor nodes generate only DATA step score code, then the score code from all of the nodes in the process flow diagram can be appended together. However, if PROC step statements are intermixed in the score code in any of the predecessor nodes, a different algorithm must be used.

SAS Code Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Code Editor Overview

You use the Code Editor to enter SAS code that executes when you run the node. The editor provides separate panes for Train, Score, and Report code. You can edit and submit code interactively in all three panes at the same time that you view the SAS log and output listings. You can also run the process flow diagram path up to and including the **SAS Code** node and view the Results window without closing the programming interface.

The Code Editor provides tables of macros and macro variables that you can use to integrate your SAS code with the SAS Enterprise Miner environment. You use the macro variables and the variables macros to reference information about the imported data sets, the target and input variables, the exported data sets, the files that store the scoring code, the decision metadata, and so on. You use the utility macros, which typically accept arguments, to manage data and format output. You can insert a macro variable, a variables macro, or a utility macro into your code without having to enter its name. You simply select an item from the macro variables list or macros table and drag it to the active code pane.

If an imported data set exists, you can access the variables table from the Code Editor. The variables table has the same functionality regardless of whether it is accessed from the Code Editor or the **SAS Code** node properties panel.

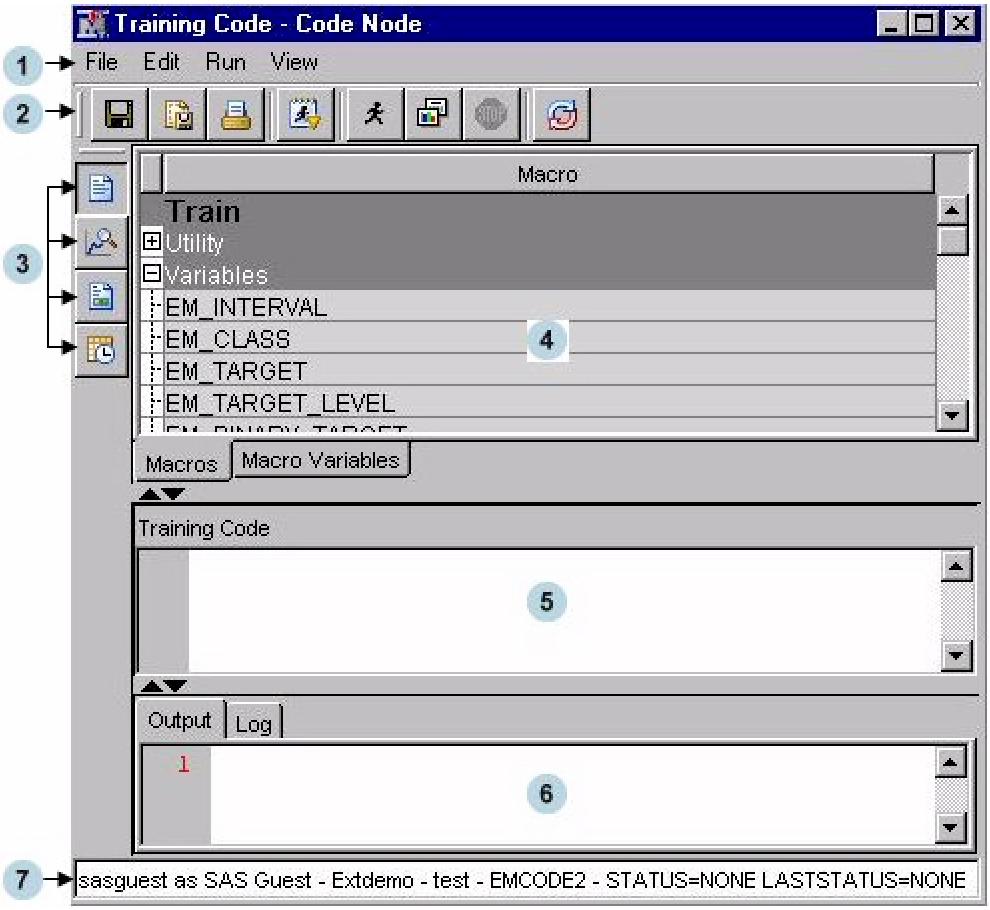
You can also access the **SAS Code** node properties panel from the Code Editor. You can specify values for any of the node properties in the Code Editor properties interface the same way you would in the **SAS Code** node properties panel.

Note: You cannot use the **SAS Code** node or SAS Enterprise Miner project start code to create an MS Excel library. If you want to use Excel data in SAS Enterprise Miner, you must use the **File Import** node to do so.

Code Editor User Interface

Introduction

The Code Editor consists of seven components. Some components serve multiple functions:



sasguest as SAS Guest - Extdemo - test - EMCODE2 - STATUS=NONE LASTSTATUS=NONE

1. “Menu” on page 1114
2. “Toolbar” on page 1115
3. “Content Selector Buttons” on page 1115
4. “Tables Pane” on page 1115
   - Macros table
   - Macro variables table
   - Variables table
5. “Code Pane” on page 1117
   - Training Code
   - Score Code
   - Report Code
6. “Results Pane” on page 1118
   - Output
   - Log
7. “Status Bar” on page 1118

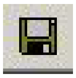
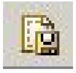
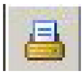
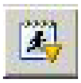
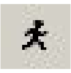
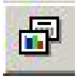
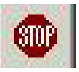
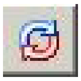
Menu

The Code Editor menu consists of the following items:

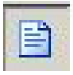
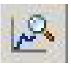
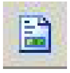
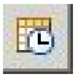
- **File**
  - **Save** — saves the contents in the current view of the code pane.
  - **Save As** — saves any combination of the code, output, or log.
  - **Save All** — saves the code, output, and log.
  - **Print** — prints the contents of the pane that currently has the focus.
  - **Exit** — closes the Code Editor window and returns to the SAS Enterprise Miner main workspace.
- **Edit**
  - **Cut** — deletes the selected item and copies it to the clipboard.
  - **Copy** — copies the selected item to the clipboard.
  - **Paste** — pastes a copied item from the clipboard.
  - **Select All** — selects all of the text from the code pane.
  - **Clear All** — clears all of the text from the current code pane.
  - **Find and Replace** — opens the Find/Replace dialog box, which enables you to search for and replace text in the code, output, and log.
- **Run**
  - **Run Code** — runs the code in the active code pane. This does not affect the status of the node or the process flow. It is simply a way to validate your code.
  - **Run Node** — runs the **SAS Code** node and any predecessor nodes in the process flow that have not been executed.
  - **Results** — opens the **SAS Code** node Results window.

  - **Stop Node** — interrupts a currently running process flow.
- **View**
  - **Training Code** — views the Training Code pane.
  - **Score Code** — views the Score Code pane.
  - **Report Code** — views the Report Code pane.
  - **Properties** — open the **SAS Code** node properties panel.

Toolbar

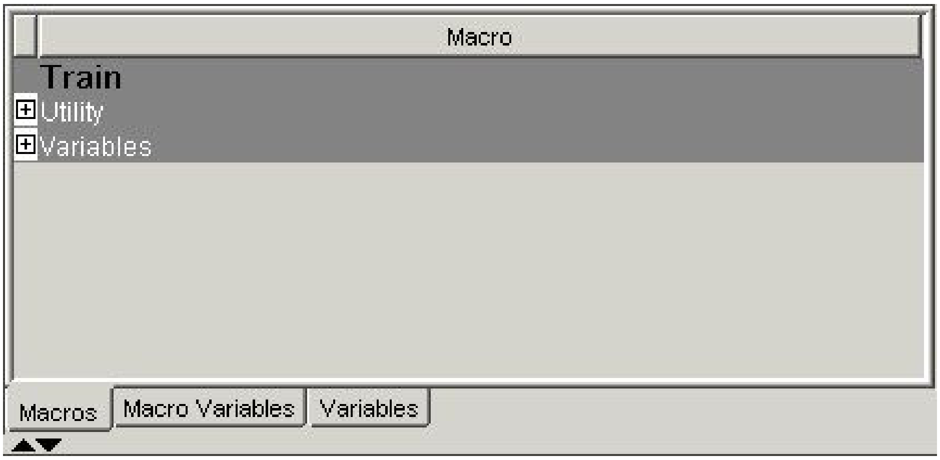
-  — saves the contents in the current view of the code pane
-  — saves the contents of the code pane, the output, and the SAS log
-  — prints the contents of the code pane, the output, or the SAS log
-  — runs the code in the active code pane
-  — runs the **SAS Code** node and any predecessor nodes in the process flow that have not been executed
-  — opens the **SAS Code** node Results window
-  — stops a currently running process flow diagram
-  — resets the workspace

Content Selector Buttons

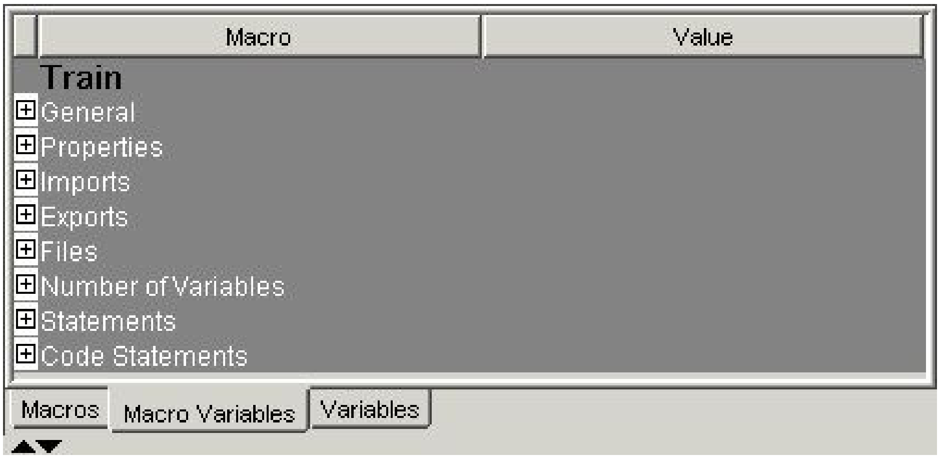
-  — displays the **SAS Code** node Training Code
-  — displays the **SAS Code** node Score Code
-  — displays the **SAS Code** node Report Code
-  — opens the property settings for the **SAS Code** node

Tables Pane

- **Macros** — Click the **Macros** tab to view a table of macros in the Tables pane. The macro variables are arranged in two groups: Utility and Variables. Click onthe plus or minus sign on the left of the group name to expand or collapse the list, respectively. You can insert a macro into your code without entering its name by selecting an item from the macros table and dragging it to the code pane.



- **Macro Variables** — Click the **Macro Variables** tab to view a table of macro variables in the Tables pane. You can use the split bar to adjust the width of the columns in the table. For many of the macro variables, you see the value to which it resolves in the Value column. But in some cases, the value cannot be displayed in the table because those macro variables are populated at run time.

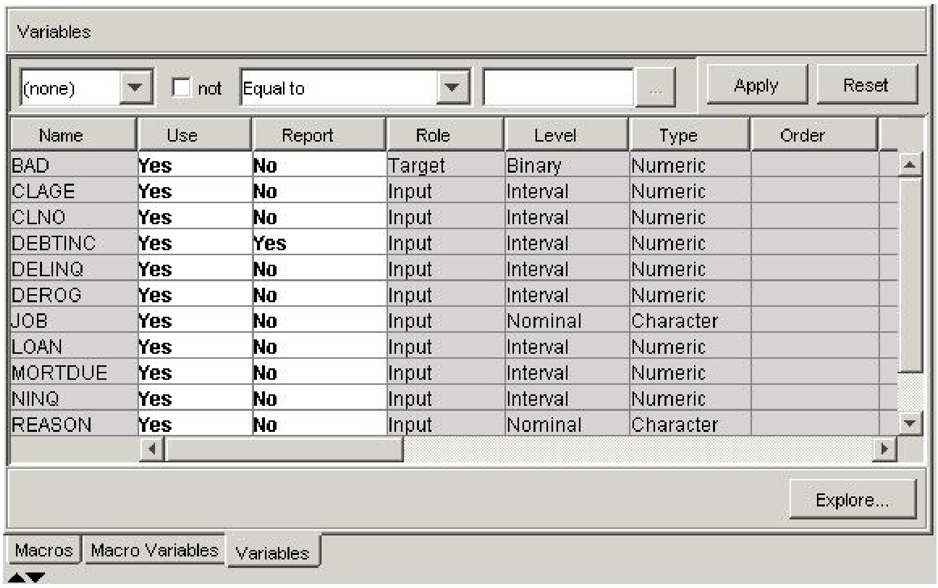


The macro variables are arranged in groups according to function:

- **General** — Use general macro variables to retrieve system information.
- **Properties** — Use properties macro variables to retrieve information about the nodes.
- **Imports** — Use imports macro variables to identify the SAS tables that are imported from predecessor nodes at run time.
- **Exports** — Use exports macro variables to identify the SAS tables that are exported to successor nodes at run time.
- **Files** — Use files macro variables to identify external files that are managed by SAS Enterprise Miner, such as log and output listings.
- **Number of Variables** — Use the number of variables macro variables for a given combination of the measurement levels and model roles.
- **Statements** — Use statements macro variables to identify SAS program statements that are frequently used by SAS Enterprise Miner, such as the decision statement in the modeling procedures.
- **Code Statements** — Use the Code Statements macro variable to identify the file containing the Code statement.

You can insert a macro variable into your code without entering its name by selecting an item from the macro variables table and dragging it to the code pane.

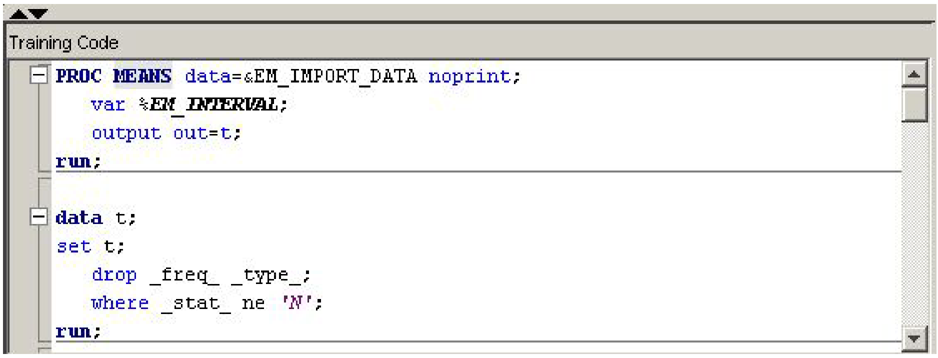
- **Variables** — Click the **Variables** tab to view the variables table in the Tables pane. The variables table has the same functionality regardless of whether it is accessed from the Code Editor or the **SAS Code** node properties panel.



Name	Use	Report	Role	Level	Type	Order
BAD	Yes	No	Target	Binary	Numeric	
CLAGE	Yes	No	Input	Interval	Numeric	
CLNO	Yes	No	Input	Interval	Numeric	
DEBTINC	Yes	Yes	Input	Interval	Numeric	
DELINQ	Yes	No	Input	Interval	Numeric	
DEROG	Yes	No	Input	Interval	Numeric	
JOB	Yes	No	Input	Nominal	Character	
LOAN	Yes	No	Input	Interval	Numeric	
MORTDUE	Yes	No	Input	Interval	Numeric	
NINQ	Yes	No	Input	Interval	Numeric	
REASON	Yes	No	Input	Nominal	Character	

Code Pane

The Code pane has three views: Training Code, Score Code, and Report Code.



```
PROC MEANS data=&EM_IMPORT_DATA noprint;
   var %EM_INTERVAL;
   output out=t;
run;

data t;
set t;
   drop _freq_ _type_;
   where _stat_ ne 'N';
run;
```

Click on the (Training), (Score), or (Report) icons on the toolbar to choose the pane in which you want to work.

The code from the three panes is executed sequentially when you select **Run** node ( ). Training code is executed first, followed by Score code, and then Report code.

If you select **Run Code** ( ), only the code in the visible code pane is executed. For more details, see the "Code Pane" on page 1117 section.

Use the ▲▼ controls to either expand (▲) or collapse (▼) the code pane.

Results Pane

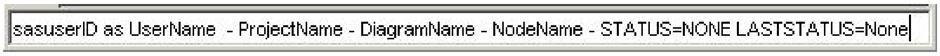
The Results pane has two tabs: Output and Log. Click the **Output** tab to view the output generated by your code or click the **Log** tab to view the SAS log that was generated by your code. If you run the node ( ), rather than just your code ( ), the output and log must be viewed from the SAS Code node's Results window ( ) and not from the Code Editor's Results pane.

Use the ▲▼ controls to either expand (▲) or collapse (▼) the Results pane.

Status Bar

The status bar displays the following:

- SAS User ID — the SAS User ID of the current SAS Enterprise Miner session owner.
- User name — the User name that is associated with the current SAS Enterprise Miner session owner.
- Project name — the name of the currently open SAS Enterprise Miner project.
- Diagram name — the name of the currently open SAS Enterprise Miner diagram.
- Node name — the name of the selected node in the current SAS Enterprise Miner diagram workspace.
- Current status — the current status of the selected node in the current SAS Enterprise Miner diagram workspace.
- Last status — the last known status of the selected node in the current SAS Enterprise Miner diagram workspace.



sasuserID as UserName - ProjectName - DiagramName - NodeName - STATUS=NONE LASTSTATUS=None

Code Editor Macros

Overview

The Macros table lists the SAS macros that are used to encode multiple values, such as a list of variables, and functions that are already programmed in SAS Enterprise Miner. The macro variables are arranged in two groups: Utility and Variables. Utility macros are used to manage data and format output and Variables macros are used to identify variable definitions at run time. The macros discussion below is organized as follows:

- "Utility Macros" on page 1119
  - %EM_REGISTER
  - %EM_REPORT
  - %EM_MODEL
  - %EM_DATA2CODE
  - %EM_DECDATA
  - %EM_CHECKMACRO
  - %EM_CHECKSETINIT

  - %EM_ODSLISTON
  - %EM_ODSLISTOFF
  - %EM_METACHANGE
  - %EM_GETNAME
  - %EM_CHECKERROR
  - %EM_PROPERTY
- "Variables Macros" on page 1124

Utility Macros

Use utility macros to manage data and format output.

The following utility macros are available:

- **%EM_REGISTER** — Use the %EM_REGISTER macro to register a unique file key. When you register a key, SAS Enterprise Miner generates a macro variable named &EM_USER_key. You then use &EM_USER_key in your code to associate a file with the key. Registering a file enables SAS Enterprise Miner to track the state of the file, avoid name conflicts, and ensure that the registered file is deleted when the node is deleted from a process flow diagram.

  %EM_REGISTER allows the following arguments:

  - ACTION = < TRAIN | SCORE | REPORT > — associates the registered CATALOG, DATA, FILE, or FOLDER with an action. If the registered object is modified, the associated action is triggered to execute whenever the node is run subsequently. The default value is TRAIN. This is an optional argument. The argument has little use in the **SAS Code** node but can be of significant value to extension node developers.
  - AUTODELETE = <Y|N> — Request the delete status of the file prior to the run. This argument is optional.
  - EXTENSION = <file-extension> — an optional parameter to identify non-standard file extensions (.sas or .txt, for example).
  - FOLDER = <folder-key> — the folder key where a registered file resides (optional).
  - KEY = <data-key> — an alias for a filename.
  - PROPERTY = <Y|N> — an optional argument that indicates that the file is a node property and that when the node or the process flow diagram is exported, the content of the registered file will also be exported with the rest of the properties.
  - TYPE = <CATALOG | DATA | FILE | FOLDER> — the type of file that is to be registered.

  For example, if you want to use the data set Class from the SASHELP library, register the key Class as follows:

  ```
  %em_register(key=Class, type=data);
  ```

  Later, in your code, you can use statements like the following:

  ```
  data &em_user_Class;
  set Sashelp.Class;
  ```

  References to &EM_USER_Class then resolve to the permanent data set Sashelp.Class.

- **%EM_REPORT** — Use the %EM_REPORT macro to specify the contents of a results window display that is created using a registered data set. The display contents, or view, can be a data table view or a plot view. Examples of plot types are histograms, bar charts, and line plots. The views (both tables and plots) appear in the results window of the **SAS Code** node and in any results package files (SPK files).

  %EM_REPORT allows the following arguments:

  - AUTODISPLAY = <Y | N> — specifies whether the report displays automatically when the results viewer is opened.
  - BLOCK = <group-name> — specifies the group that the report belongs to when the results viewer is opened. The default setting is CUSTOM.
  - COLOR = <variable-name> — specifies a variable that contains color value.
  - COMPARE = <Y | N> — specifies whether data in the generated report can be used to compare registered models. The default setting is N.
  - DESCRIPTION = <window-title-description> — specifies a text string or report description that appears in the window title.
  - DISCRETEX = <Y | N> — specifies whether the values on the x-axis is discrete when the VIEWTYPE is HISTOGRAM.
  - DISCRETEY = <Y | N> — specifies whether the values on the y-axis is discrete when the VIEWTYPE is HISTOGRAM.
  - EQUALIZECOLX = <Y | N> — specifies if the x-axis should be equalized (that is, use a shared common scale and tick marks) across columns of the lattice. The default setting is N.
  - EQUALIZECOLY = <Y | N> — specifies if the y-axis should be equalized across columns of the lattice. The default setting is N.
  - EQUALIZEROWX = <Y | N> — specifies if the x-axis should be equalized (that is, it should use a shared common scale and tick marks) across rows of the lattice. The default setting is N.
  - EQUALIZEROWY = <Y | N> — specifies if the y-axis should be equalized across rows of the lattice. The default setting is N.
  - FREQ = <frequency-variable-name> — specifies a frequency variable.
  - GROUP = <group-variable-name(s)> — specifies one or more grouping variables.
  - IDVALUE = <data-set-name> — specifies a data set. When a corresponding variable name is specified using the REPORTID argument, a report is generated for each value of the specified variable in the named data set. A report window is created for each unique value.
  - KEY = <data-key> (required) — specifies the data key. Because this is a required argument, you must assign the data key using %EM_REGISTER before using %EM_REPORT.
  - LATTICETYPE = <viewtype> — valid viewtypes are Data, Scatter, Lineplot, Bar, Histogram, Pie, Profileview, Gainsplot.
  - LATTICEX = <lattice-row-variable-name> — specifies variables to be used as rows in a lattice.
  - LATTICEY = <lattice-column-variable-name> — specifies variables to be used as columns in a lattice.

  - LOCALIZE = <Y | N> — specifies whether the description should be localized or used as-is. The default setting is N.
  - REPORTID = <variable-name> — specifies a variable name. When a corresponding data set name is specified using the IDVALUE argument, a report is generated for each value of the specified variable in the named data set. A report window is created for each unique value.
  - SLIDER = <Y | N> — specifies wether the threshold slider tool is visible for constellation plots.
  - SPK = <Y | N> — specifies whether to include the report and data in an SPK package. The default setting is Y.
  - SUBGROUP = <subgroup-variable-name(s)> — specifies one or more subgrouping variables.
  - TIPTEXT = <variable-name> — specifies a variable that contains tooltip text.
  - TOOLTIP = <variable-name> — specifies a variable containing tooltip text for the Constellation application.
  - VIEWS = <numeric-value> — assigns a numeric ID to the generated report.
  - VIEWTYPE = < plot-type > — specifies the type of plot that you want to display. Valid plot types include Data, Bar, Histogram, Lineplot, Pie, Profileview, Scatter, Gainsplot, Lattice, Dendrogram, and Constellation. Data is the default value.
  - WHERE = — specifies an explicit SQL WHERE clause.
  - X = <x-variable-name> — specifies the x-axis variable.
  - XREF = <numeric-value> — specifies a reference line on the x-axis.
  - Y = <y-variable-name> — specifies the y-axis variable.
  - Yn = <Yn-variable-name> — where n is an integer ranging from 1 to 16. Y1, Y2, ... , Y16 specify variables that are to be plotted and overlaid on the y-axis.
  - YREF = <numeric-name> — specifies a reference line on the y-axis.
  - Z = <z-variable-name> — specifies the z-axis variable of a 3-dimensional plot.

  Examples using %EM_REPORT are provided below.

- **%EM_MODEL** — The %EM_MODEL macro enables you to control the computations that are performed and the score code that is generated by the SAS Enterprise Miner environment for modeling nodes.

  The macro supports the following arguments:

  - TARGET = <target-variable-name> — name of the target (required).
  - ASSESS = <Y|N> — assess the target. The default is Y.
  - DECSCORECODE = <Y|N> — generate decision score code. The default is N.
  - FITSTATISTICS = <Y|N> — compute fit statistics. The default is N.
  - CLASSIFICATION = <Y|N> — generate score code to generate classification variables (I_, F_, U_) . The default is N.
  - RESIDUALS = <Y|N> — generate score code to compute residuals. The default is N.
  - PREDICTED = <Y|N> — indicate whether the node generates predicted values. The default is Y.

For example, suppose you have a binary target variable named BAD and your code only generates posterior variables. You can use the %EM_MODEL macro to indicate that you want SAS Enterprise Miner to generate fit statistics, assessment statistics, and to generate score code that computes classification, residual, and decision variables.

```
%em_model(
   target=BAD,
   assess=Y,
   decscorecode=Y,
   fitstatistics=Y,
   classification=Y,
   residuals=Y,
   predicted=Y);
```

Note: %EM_MODEL is available for use in your code, but it does not currently appear in the Code Editor's table of macros.

- **%EM_DATA2CODE** — The %EM_DATA2CODE macro converts a SAS data set to SAS program statements. For example, it can be used to embed the parameter estimates that PROC REG creates directly into scoring code. The resulting scoring code can be deployed without need for an EST data set. You must provide the code to use the parameter estimates to produce a model score.

  %EM_DATA2CODE accepts the following arguments:

  - APPEND= <Y | N> — specifies whether to append or overwrite code if the specified file already exists.
  - DATA= <source-data-name> — specifies the source data.
  - OUTDATA= <output-data-set-name> — specifies the name of the output data set that is created when the DATA step code runs.
  - OUTFILE= <output-data-step-code-filename> — specifies the name of the output file that contains the generated SAS DATA step code.

- **%EM_DECDATA** — The %EM_DECDATA macro uses information that you entered to create the decision data set that is used by SAS Enterprise Miner modeling procedures. %EM_DECDATA copies the information to the WORK library and assigns the proper type (profit, loss, or revenue) for modeling procedures.

  - DECDATA = <decision-data-set> — specifies the data set that contains the decision data set.
  - DECMETA = <decision-metadata — specifies the data set that contains decision metadata.
  - NODEID = <node-identifier> — specifies the unique node identifier.

- **%EM_CHECKMACRO** — Use the EM_CHECKMACRO macro to check for the existence of a macro variable. Assigning a value is optional.

  %EM_CHECKMACRO accepts the following arguments:

  - NAME = <macro-variable-name> — specifies the name of the macro variable for which you want to check.
  - GLOBAL = <Y | N> — specifies whether the named macro variable is a global macro variable.
  - VALUE = <variable-value> — specifies a value for the macro variable if it has not been previously defined.

- **%EM_CHECKSETINIT** — Use the %EM_CHECKSETINIT macro to validate and view your SAS product licensing information.

  %EM_CHECKSETINIT has the following required argument:

  - PRODUCTID = <product id number> — specifies the product identification number. If the product specified is not licensed, SAS Enterprise Miner issues an error and halts execution of the program.

- **%EM_ODSLISTON** — Use the %EM_ODSLISTON macro to turn on the SAS Output Delivery System (ODS) listing and to specify a name for the destination HTML file.

  %EM_ODSLISTON accepts the following arguments:

  - FILE = <destination-file> — specifies the name of an HTML output file that contains the generated ODS listing.

- **%EM_ODSLISTOFF** — Use the %EM_ODSLISTOFF utility macro to turn SAS ODS listing off. No argument is needed for this macro.

- **%EM_METACHANGE** — Use the %EM_METACHANGE macro to modify the columns metadata data set that is exported by a node. The macro should be called during either the TRAIN or SCORE actions.

  %EM_METACHANGE allows the following arguments:

  - NAME = <variable-name> — the name of the variable that you want to modify (required).

  - ROLE = <ASSESS | CENSOR | CLASSIFICATION | COST | CROSSID | DECISION | FREQ | ID | INPUT | KEY | LABEL | PREDICT | REFERRER | REJECTED | RESIDUAL | SEGMENT | SEQUENCE | TARGET | TEXT | TEXTLOC | TIMEID | TREATMENT | URL > — assign a new role to the variable (optional).

    Note: You cannot specify more than one frequency variable in a SAS Enterprise Miner data source.

  - LEVEL = <UNARY | BINARY | ORDINAL | NOMINAL | INTERVAL> — assign a new measurement level to the variable (optional).

  - ORDER = <ASC | DESC | FMTASC | FMTDESC> — new level ordering for a class variable (optional).

  - COMMENT = <string> — a string that can be attached to a variable (optional).

  - LOWERLIMIT = <number> — the lower limit of a numeric variable's valid range (optional).

  - UPPERLIMIT = <number> — the upper limit of a numeric variable's valid range.

  - DELETE = <Y | N> — indicates whether the variable should be removed from the metadata (optional).

  - KEY = <data-key> — specifies which data set will receive the delta code, or metadata changes. If KEY= is not specified, then the default setting for KEY= is CDELTA_TRAIN, the exported training data set. The generated code is stored in the CDELTA.TRAIN.sas file in the node folder.

    If there are multiple source of data and metadata feeding in the node, then a user can use other keys to modify the metadata. Other valid values for KEY= are CDELTA_VALIDATE (an exported validation data set), CDELTA_TEST (an exported test data set), CDELTA_SCORE (an exported score data set), or CDELTA_TRANSACTION (an exported transaction data set).

- **%EM_GETNAME** — Use %EM_GETNAME to retrieve the name of a file or data set that is registered to a given key. The macro initializes the EM_USER_key macro variable. This macro should be called in actions other than CREATE, rather than call the EM_REGISTER macro.

  %EM_GETNAME allows the following arguments:

  - KEY = <data-key> — the registered data key
  - TYPE = <CATALOG | DATA | FILE | FOLDER | GRAPH> — the type of file that is registered.
  - EXTENSION = <file-extension> — an optional parameter to identify nonstandard file extensions.
  - FOLDER = <folder-key> — the folder key where a registered file resides (optional).

- **%EM_CHECKERROR** — This macro checks the return code and initializes the &EMEXCEPTIONSTRING macro variable. %EM_CHECKERROR has no arguments.

- **%EM_PROPERTY** — Use %EM_PROPERTY in the CREATE action to initialize the &EM_PROPERTY_name macro variable for the specified property. The macro enables you to specify the initial value to which &EM_PROPERTY_name resolves. You can also associate the property with a specific action (TRAIN, SCORE, or REPORT).

  %EM_PROPERTY allows the following arguments:

  - NAME = <property name> — specify the name of the property that is to be initialized (required). This is case sensitive and must match the property name that is specified in the XML properties file.
  - VALUE = <initial value> — specify the initial value for the property (required). The value should match the initial attribute that is specified for the property in the XML properties file.
  - ACTION = <TRAIN | SCORE | REPORT> — specify the action that is associated with the property (optional).

- **%EM_DATASET_VERIFY** — Use the %EM_DATASET_VERIFY macro to verify that a data set exists, can be open, and has the correct variables before further processing. Errors are generated, printed, and returned, by default using the sashelp.dmine data set error messages. The **Text Rule Builder** node uses this macro.

  The following are examples of how to use the %EM_DATASET_VERIFY macro along with possible output from running the macro:

```
%em_dataset_verify(sashelp.dmine,vars=locale KEY Lineno text key);
        *no error;
%em_dataset_verify(sashelp.dminemine,vars=locale KEY Lineno text key);
       ----> The data set "sashelp.dminemine" does not exist.
%em_dataset_verify(sashelp.dmine,vars=locale target TARGET);
       ---->ERROR: The following columns were not found in the Table
                  "sashelp.dmine: target TARGET":
```

Variables Macros

Use the variables macros to identify variable definitions at run time. Variables appear in these macros only if the variable's Use or Report status is set to Yes.

- **%EM_INTERVAL** — resolves to the input variables that have an interval measurement level. Interval variables are continuous variables that contain values across a range.
- **%EM_CLASS** — resolves to the categorical input variables, including all inputs that have a binary, nominal, or ordinal measurement level.
- **%EM_TARGET** — resolves to the variables that have a model role of target. The target variable is the dependent or the response variable.
- **%EM_TARGET_LEVEL** — resolves to the measurement level of the target variable.
- **%EM_BINARY_TARGET** — resolves to the binary variables that have a model role of target.
- **%EM_ORDINAL_TARGET** — resolves to the ordinal variables that have a model role of ordinal.
- **%EM_NOMINAL_TARGET** — resolves to the nominal variables that have a model role of nominal.
- **%EM_INTERVAL_TARGET** — resolves to the interval variables that have a model role of target.
- **%EM_INPUT** — resolves to the variables that have a model role of input. The input variables are the independent or predictor variables.
- **%EM_BINARY_INPUT** — resolves to the binary variables that have a model role of input.
- **%EM_ORDINAL_INPUT** — resolves to the ordinal variables that have a model role of input.
- **%EM_NOMINAL_INPUT** — resolves to the nominal variables that have a model role of input.
- **%EM_INTERVAL_INPUT** — resolves to the interval variables that have a model role of input.
- **%EM_REJECTED** — resolves to the variables that have a model role of REJECTED and a USE value of Yes.
- **%EM_BINARY_REJECTED** — resolves to the binary variables that have a model role of rejected.
- **%EM_ORDINAL_REJECTED** — resolves to the ordinal variables that have a model role of rejected.
- **%EM_NOMINAL_REJECTED** — resolves to the nominal variables that have a model role of rejected.
- **%EM_INTERVAL_REJECTED** — resolves to the interval variables that have a model role of rejected.
- **%EM_ASSESS** — resolves to the variables that have a model role of assessment.
- **%EM_CENSOR** — resolves to the variables that have a model role of censor.
- **%EM_CLASSIFICATION** — resolves to the variables that have a model role of classification.
- **%EM_COST** — resolves to the variables that have a model role of cost.
- **%EM_CROSSID** — resolves to the variables that have a model role of Cross ID.
- **%EM_DECISION** — resolves to the variables that have a model role of decision.

- **%EM_FREQ** — resolves to the variables that have a model role of freq.
- **%EM_ID** — resolves to the variables that have a model role of ID.
- **%EM_LABEL** — resolves to the variables that have a model role of label.
- **%EM_PREDICT** — resolves to the variables that have a model role of prediction.
- **%EM_REFERRER** — resolves to the variables that have a model role of referrer.
- **%EM_REJECTS** — resolves to the variables that have a model role of REJECTED.
- **%EM_REPORT_VARS** — resolves to the variables that have a model role of report.
- **%EM_CLASS_REPORT** — resolves to the class variables that have a model role of report.
- **%EM_INTERVAL_REPORT** — resolves to the interval variables that have a model role of report.
- **%EM_RESIDUAL** — resolves to the variables that have a model role of residual.
- **%EM_SEGMENT** — resolves to the variables that have a model role of segment.
- **%EM_SEQUENCE** — resolves to the variables that have a model role of sequence.
- **%EM_TEXT** — resolves to the variables that have a model role of text.
- **%EM_TIMEID** — resolves to the variables that have a model role of Time ID.

Macro Variables

Overview

The Macro Variables table lists the macro variables that are used to encode single values such as the names of the input data sets. The macro variables are arranged in groups according to function:

- “General” on page 1126
- “Properties” on page 1127
- “Imports” on page 1128
- “Exports” on page 1129
- “Files” on page 1130
- “Number of Variables” on page 1131
- “Statements” on page 1132
- “Code Statements” on page 1133

General

Use general macro variables to retrieve system information.

- **&EM_USERID** — resolves to the user name.
- **&EM_METAHOST** — resolves to the host name of SAS Metadata Repository.
- **&EM_METAPORT** — resolves to the port number of SAS Metadata Repository.

- **&EM_LIB** — resolves to the numbered EMWS SAS library containing the data sets and SAS catalogs related to the current process flow diagram. This is the same as the value of the process flow diagram's ID property.
- **&EM_DSEP** — resolves to the operating system file delimiter (for example, backslash (\) for Windows and slash (/) for UNIX).
- **&EM_CODEBAR** — resolves to the macro variable that identifies a code separator.
- **&EM_VERSION** — resolves to the version of SAS Enterprise Miner.
- **&EM_TOOLTYPE** — resolves to the node type (Sample | Explore | Modify | Model | Assess |Utility).
- **&EM_NODEID** — resolves to the node ID.
- **&EM_NODEDIR** — resolves to the path to the node folder.
- **&EM_SCORECODEFORMAT** — resolves to the format of the score code (DATASTEP | OTHER).
- **&EM_PUBLISHCODE** — resolves to the Publish Code property (FLOW | PUBLISH).
- **&EM_META_ADVISOR** — resolves to the Advisor Type property (BASIC | ADVANCED). This is equivalent to &EM_PROPERTY_MetaAdvisor.
- **&EM_MININGFUNCTION** — resolves to a description of the function of the node.

Properties

Use properties macro variables to retrieve information about the nodes.

- **&EM_PROPERTY_ScoreCodeFormat** — resolves to the value of the Code Format property.
- **&EM_PROPERTY_MetaAdvisor** — resolves to the value of the Advisor Type property. This is equivalent to &EM_Meta_Advisor.
- **&EM_PROPERTY_ForceRun** — resolves to Y or N. When set to Y, the node and its successors will rerun even though no properties, variables or imports have changed.
- **&EM_PROPERTY_UsePriors** — resolves to the value of the Use Priors property.
- **&EM_PROPERTY_ToolType** — resolves to the value of the Tool Type property.
- **&EM_PROPERTY_DataNeeded** — resolves to the value of the Data Needed property.
- **&EM_PROPERTY_VariableSet** — resolves to the name of the catalog containing the VariableSet.
- **&EM_PROPERTY_PublishCode** — resolves to the value of the Publish Code property.
- **&EM_PROPERTY_NotesFile** — resolves to the name of the file containing the contents of the Notes Editor.
- **&EM_PROPERTY_Component** — resolves to the SAS Enterprise Miner node name.
- **&EM_PROPERTY_RunID** — resolves to the value of the Run ID property. Each time the node is run a new ID is generated.

Imports

Use imports macro variables to identify the SAS tables that are imported from predecessor nodes at run time.

- **&EM_IMPORT_DATA** — resolves to the name of the training data set.
- **&EM_IMPORT_DATA_CMETA** — resolves to the name of the column metadata data set that corresponds to the training data set.
- **&EM_IMPORT_VALIDATE** — resolves to the name of the validation data set.
- **&EM_IMPORT_VALIDATE_CMETA** — resolves to the name of the column metadata data set that corresponds to the validation data set.
- **&EM_IMPORT_TEST** — resolves to the name of the test data set.
- **&EM_IMPORT_TEST_CMETA** — resolves to the name of the column metadata data set that corresponds to the test data set.
- **&EM_IMPORT_SCORE** — resolves to the name of the score data set.
- **&EM_IMPORT_SCORE_CMETA** — resolves to the name of the column metadata data set that corresponds to the score data set.
- **&EM_IMPORT_TRANSACTION** — resolves to the name of the transaction data set.
- **&EM_IMPORT_TRANSACTION_CMETA** — resolves to the name of the column metadata data set that corresponds to the transaction data set.
- **&EM_IMPORT_DOCUMENT** — resolves to the name of the document data set.
- **&EM_IMPORT_DOCUMENT_CMETA** — resolves to the name of the column metadata data set that corresponds to the document data set.
- **&EM_IMPORT_RULES** — resolves to the name of the rules data set that is exported from a predecessor Association or Path Analysis node.
- **&EM_IMPORT_REPORTFIT** — resolves to the name of the fit statistics data set.
- **&EM_IMPORT_RANK** — resolves to the name of the rank data set.
- **&EM_IMPORT_SCOREDIST** — resolves to the name of the score distribution data set.
- **&EM_IMPORT_ESTIMATE** — resolves to the name of the parameter estimates data set.
- **&EM_IMPORT_TREE** — resolves to the name of the tree data set from a predecessor modeling node.
- **&EM_IMPORT_CLUSSTAT** — resolves to the name of the cluster statistics data set from a predecessor Cluster node.
- **&EM_IMPORT_CLUSMEAN** — resolves to the name of the cluster mean data set from a predecessor Cluster node.
- **&EM_IMPORT_VARMAP** — resolves to the name of the data set of variable mapping from a predecessor Cluster node.
- **&EM_METASOURCE_NODEID** — resolves to the node ID that is providing the variables metadata.
- **&EM_METASOURCE_CLASS** — resolves to the class of the node.
- **&EM_METASOURCE_CHANGED** — resolves to Y or N, indicating whether the source of the metadata has changed.

Exports

Use exports macro variables to identify the SAS tables that are exported to successor nodes at run time.

- **&EM_EXPORT_TRAIN** — resolves to the name of the export training data set.
- **&EM_TRAIN_SCORE** — resolves to Y or N, indicating whether SAS Enterprise Miner should score the training data set.
- **&EM_TRAIN_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the training column metadata data set.
- **&EM_EXPORT_TRAIN_CMETA** — resolves to the name of the column metadata data set that corresponds to the export training data set.
- **&EM_EXPORT_VALIDATE** — resolves to the name of the export validation data set.
- **&EM_VALIDATE_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output validation data set.
- **&EM_VALIDATE_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the validation column metadata data set.
- **&EM_EXPORT_VALIDATE_CMETA** — resolves to the name of the column metadata data set that corresponds to the export validation data set.
- **&EM_EXPORT_TEST** — resolves to the name of the export test data set.
- **&EM_TEST_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output test data set.
- **&EM_TEST_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the test column metadata data set.
- **&EM_EXPORT_TEST_CMETA** — resolves to the name of the column metadata data set that corresponds to the export test data set.
- **&EM_EXPORT_SCORE** — resolves to the name of the export score data set.
- **&EM_SCORE_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output score data set.
- **&EM_SCORE_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the score column metadata data set.
- **&EM_EXPORT_SCORE_CMETA** — resolves to the name of the column metadata data set that corresponds to the export score data set.
- **&EM_EXPORT_TRANSACTION** — resolves to the name of the export transaction data set.
- **&EM_TRANSACTION_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output transaction data set.
- **&EM_TRANSACTION_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the transaction column metadata data set.
- **&EM_EXPORT_TRANSACTION_CMETA** — resolves to the name of the column metadata data set that corresponds to the export transaction data set.
- **&EM_EXPORT_DOCUMENT** — resolves to the name of the export document data set.

- **&EM_DOCUMENT_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output document data set.
- **&EM_DOCUMENT_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the document column metadata data set.
- **&EM_EXPORT_DOCUMENT_CMETA** — resolves to the name of the column metadata data set that corresponds to the export document data set.

Files

Use files macro variables to identify external files that are managed by SAS Enterprise Miner, such as log and output listings. Not all nodes create or manage all external files.

- **&EM_DATA_IMPORTSET** — resolves to the name of the data set containing metadata for the imported data sets.
- **&EM_DATA_EXPORTSET** — resolves to the name of the data set containing metadata for the exported data sets.
- **&EM_DATA_VARIABLESET** — resolves to the data set containing metadata for the variables that are available for use with the node.
- **&EM_DATA_ESTIMATE** — resolves to the name of the parameter estimates data set.
- **&EM_DATA_EMTREE** — resolves to the name of the tree data set.
- **&EM_DATA_EMREPORTFIT** — resolves to the name of the fit statistics data set in columns format.
- **&EM_DATA_EMOUTFIT** — resolves to the name of the fit statistics data set.
- **&EM_DATA_EMCLASSIFICATION** — resolves to the name of the data set that contains classification statistics for categorical targets.
- **&EM_DATA_EMRESIDUAL** — resolves to the name of the data set that contains summary statistics for residuals for interval targets.
- **&EM_DATA_EMRANK** — resolves to the name of the data set that contains assessment statistics such as lift, cumulative lift, and profit.
- **&EM_DATA_EMSCOREDIST** — resolves to the name of the data set that contains assessment statistics such as mean, minimum, and maximum.
- **&EM_DATA_INTERACTION** — resolves to the name of the interaction data set.
- **&EM_DATA_EMTRAINVARIABLE** — resolves to the name of the training variable data set.
- **&EM_CATALOG_EMNODELABEL** — resolves to the name of the node catalog.
- **&EM_FILE_EMNOTES** — resolves to the name of the file containing your notes.
- **&EM_FILE_EMLOG** — resolves to the name of the SAS Enterprise Miner output log file.
- **&EM_FILE_EMOUTPUT** — resolves to the name of the SAS Enterprise Miner output data file.
- **&EM_FILE_EMTRAINCODE** — resolves to the name of the file that contains the training code.
- **&EM_FILE_EMFLOWSCORECODE** — resolves to the name of the file that contains the flow score code.

- **&EM_FILE_EMPUBLISHSCORECODE** — resolves to the name of the file that contains the publish score code.
- **&EM_FILE_EMPMML** — resolves to the name of the PMML file.
- **&EM_FILE_CDELTA_TRAIN** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the training data set that is exported by a node (if one exists).
- **&EM_FILE_CDELTA_TRANSACTION** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the transaction data set that is exported by a node (if one exists).
- **&EM_FILE_CDELTA_DOCUMENT** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the document data set that is exported by a node (if one exists).

Number of Variables

Use the number of variables macro variables for a given combination of Level and Role. These macro variables only count variables that have a Use or Report status of Yes.

- **&EM_NUM_VARS** — resolves to the number of variables.
- **&EM_NUM_INTERVAL** — resolves to the number of interval variables.
- **&EM_NUM_CLASS** — resolves to the number of class variables.
- **&EM_NUM_TARGET** — resolves to the number of target variables.
- **&EM_NUM_BINARY_TARGET** — resolves to the number of binary target variables.
- **&EM_NUM_ORDINAL_TARGET** — resolves to the number of ordinal target variables.
- **&EM_NUM_NOMINAL_TARGET** — resolves to the number of nominal target variables.
- **&EM_NUM_INTERVAL_TARGET** — resolves to the number of interval target variables.
- **&EM_NUM_BINARY_INPUT** — resolves to the number of binary input variables.
- **&EM_NUM_ORDINAL_INPUT** — resolves to the number of ordinal input variables.
- **&EM_NUM_NOMINAL_INPUT** — resolves to the number of nominal input variables.
- **&EM_NUM_INTERVAL_INPUT** — resolves to the number of interval input variables.
- **&EM_NUM_BINARY_REJECTED** — resolves to the number of rejected binary input variables.
- **&EM_NUM_ORDINAL_REJECTED** — resolves to the number of rejected ordinal input variables.
- **&EM_NUM_NOMINAL_REJECTED** — resolves to the number of rejected nominal input variables.
- **&EM_NUM_INTERVAL_REJECTED** — resolves to the number of rejected interval input variables.

- **&EM_NUM_ASSESS** — resolves to the number of variables that have the model role of Assess.
- **&EM_NUM_CENSOR** — resolves to the number of variables that have the model role of Censor.
- **&EM_NUM_CLASSIFICATION** — resolves to the number of variables that have the model role of Classification.
- **&EM_NUM_COST** — resolves to the number of variables that have the model role of Cost.
- **&EM_NUM_CROSSID** — resolves to the number of variables that have the model role of Cross ID.
- **&EM_NUM_DECISION** — resolves to the number of variables that have the model role of Decision.
- **&EM_NUM_FREQ** — resolves to the number of variables that have the model role of Freq.
- **&EM_NUM_ID** — resolves to the number of variables that have the model role of ID.
- **&EM_NUM_LABEL** — resolves to the number of variables that have the model role of Label.
- **&EM_NUM_PREDICT** — resolves to the number of variables that have the model role of Predict.
- **&EM_NUM_REFERRER** — resolves to the number of variables that have the model role of Referrer.
- **&EM_NUM_REJECTS** — resolves to the number of variables that have the model role of Rejected.
- **&EM_NUM_REPORT_VAR** — resolves to the number of variables that have the model role of Report.
- **&EM_NUM_CLASS_REPORT** — resolves to the number of class variables that have the model role of Report.
- **&EM_NUM_INTERVAL_REPORT** — resolves to the number of interval variables that have the model role of Report.
- **&EM_NUM_RESIDUAL** — resolves to the number of variables that have the model role of Residual.
- **&EM_NUM_SEGMENT** — resolves to the number of variables that have the model role of Segment.
- **&EM_NUM_SEQUENCE** — resolves to the number of variables that have the model role of Sequence.
- **&EM_NUM_TEXT** — resolves to the number of variables that have the model role of Text.
- **&EM_NUM_TIMEID** — resolves to the number of variables that have the model role of Time ID.

Statements

Statements macro variables resolve to values that refer to information about decision variables and decision information. These macro variables are empty when there is more than one target variable.

- **&EM_DEC_TARGET** — resolves to the name of the target variable.
- **&EM_DEC_LEVEL** — resolves to the event level.
- **&EM_DEC_ORDER** — resolves to the sorting order of the target levels (ASCENDING | DESCENDING).
- **&EM_DEC_FORMAT** — resolves to the format of the decision target variable.
- **&EM_DEC_DECMETA** — resolves to the decision metadata data set of the target variable.
- **&EM_DEC_DECDATA** — resolves to the decision data set of the target variable.
- **&EM_DEC_STATEMENT** — resolves to the decision statement.

Code Statements

Use the Code Statements macro variable to identify the file containing the CODE statement.

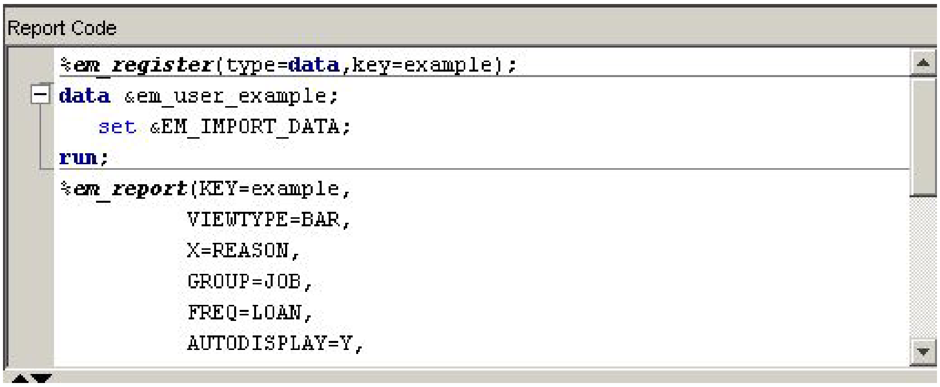
- **&EM_STATEMENT_RESCODE** — resolves to the file containing a CODE statement with a residuals option. In effect, this resolves to the file containing FLOW scoring code (&EM_FILE_EMFLOWSCORECODE).
- **&EM_STATEMENT_CODE** — resolves to the file for containing a CODE statement does not have a residuals option. In effect, this resolves to the file containing PUBLISH scoring code (&EM_FILE_EMPUBLISHSCORECODE).

Code Pane

The code pane is where you write new SAS code or where you import existing code from an external source. Any valid SAS language program statement is valid for use in the **SAS Code** node with the exception that you cannot issue statements that generate a SAS windowing environment. The SAS windowing environment from Base SAS is not compatible with SAS Enterprise Miner. For example, you cannot execute SAS/Lab from within a SAS Enterprise Miner **SAS Code** node.

The code pane has three views: Training Code, Score Code, and Report Code. You can use either the icons on the toolbar or the **View** menu to select the editor in which you want to work.

When you enter SAS code in the code pane, DATA steps and PROC steps are presented as collapsible and expandable blocks of code. The code pane itself can be expanded or contracted using the ▲▼ icons located at the bottom left-side of the pane.



```
%em_register(type=data,key=example);
data &em_user_example;
   set &EM_IMPORT_DATA;
run;
%em_report(KEY=example,
         VIEWTYPE=BAR,
         X=REASON,
         GROUP=JOB,
         FREQ=LOAN,
         AUTODISPLAY=Y,
```

You can drag macros and macro variables from their respective tables into the code pane. This speeds up the coding process and prevents spelling errors. You can import SAS code that is stored as a text file or a source entry in a SAS catalog. If your code is in an external text file, then follow this example:

```
filename fref "path-name\mycode.sas";
%inc fref;
filename fref;
```

If your code is in a catalog, follow this example:

```
filename fref catalog "libref.mycatalog.myentry.source";
%inc fref;
filename fref;
```

The code in the three views is executed sequentially when the node is run. Training code is executed first, followed by Score code, and finally, Report code. Suppose, for example, that you make changes to your Report code but do not change your Training and Score code. When you run your node from within the Code Editor, SAS Enterprise Miner does not have to rerun the Training and Score code; it just reruns the Report code. This can save considerable time if you have complex code or very large data sets.

The three views are designed to be used in the following manner:

- **Training Code** — Write code that passes over the input training or transaction data to produce some result in the Training Code pane. Here is an example:

  ```
  proc means data=&em_import_data;
  output out=m;
  run;
  ```

  You should also write dynamic scoring code in the training code pane. Scoring code is code that generates new variables or transforms existing variables. Dynamic scoring code, as opposed to static scoring code, is written so that no prior knowledge of the properties of any particular data set is assumed. That is, the code is not unique to a particular process flow diagram. For example, suppose that you begin your process flow diagram with a particular data source and it is followed by a **SAS Code** node that contains dynamic scoring code. If you changed the data source in the diagram, the dynamic scoring code should still execute properly. Dynamic scoring code can make use of SAS PROC statements and macros, whereas static scoring code cannot.

- **Score Code** — Write code that modifies the train, validate, test, or transaction data sets for the successor nodes. The Score view is, however, reserved for static scoring code. Static scoring code makes references to properties of a specific data set, such as variable names, so the code is unique for a particular process flow diagram. Here is an example:

  ```
  logage= log(age);
  ```

  If you write dynamic scoring code in the Score Code pane, it will not execute. Scoring code that is included in the Score Code pane must be in the form of pure DATA steps. SAS PROC statements and macros will not execute in the Score Code pane.

- **Report Code** — code that generates output that is displayed to the user. The output can be in the form of graphs, tables, or the output from SAS procedures. An example is a statements such as the following:

  ```
  proc print data=m;
  run;
  ```

Calls to the macro %EM_REPORT, which are illustrated in "Examples Using %EM_REPORT" on page 1153 , are the most common form of Report code.

You can execute your code in two modes.

- Run Code ( ) — Code is executed immediately in the current SAS session.

  Only the code in the active code pane is executed. The log and output appear in the Code Editor's Results pane. If a block of code is highlighted, only that code is executed. No pre-processing or post-processing occurs. Use this mode to test and debug blocks of code during development.

- Run Node ( ) — The code node and all predecessor nodes are executed in a separate SAS session, exactly as if the user has closed the editor and run the path. All normal pre-processing and post-processing occurs. Use the Results window to view the log, output, and other results generated by your code.

Most nodes generate permanent data sets and files. However, before you can reference a file in your code, you must first register a unique file key using the %EM_REGISTER macro and then associate a file with that key. When you register a key, SAS Enterprise Miner generates a macro variable named &EM_USER_key. You use that macro variable in your code to associate the file with the key. Registering a file enables SAS Enterprise Miner to track the state of the file and avoid name conflicts.

Use the %EM_GETNAME macro to reinitialize the macro variable &EM_USER_key when referring to a file's key in a code pane other than the one in which it was registered. Using Run Code causes the code in the active code pane to execute in a separate SAS session. If the key was registered in a different pane, &EM_USER_key will not be initialized. The registered information is stored on the server, so you don't have to register the key again, but you must reinitialize &EM_USER_key.

SAS Code Node Results

To view the **SAS Code** node Results window from within the Code Editor, click the icon. Alternatively, you can view the Results window from the main SAS Enterprise Miner workspace by right-clicking the **SAS Code** node in the diagram and selecting **Results**. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu in the Results window to view the following results:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **SAS Code** node properties configuration when the node was last run.
  - **Run Status** — displays the status of the **SAS Code** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — display a table of the variables in the training data set.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays (in Read-Only mode) any notes that were previously entered in the Notes editor.
- **SAS Results**

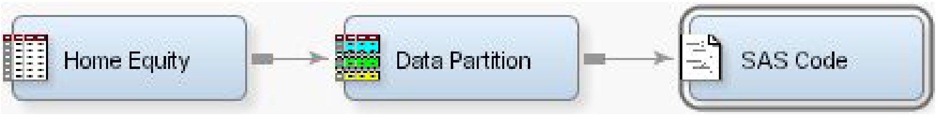
  - **Log** — the SAS log of the **SAS Code** node run.
  - **Output** — The **SAS Code** node output report, like all other nodes, includes Training Output, Score Output, and Report Output. The specific contents are determined by the results of the code that you write in the **SAS Code** node.
  - **Flow Code** — the SAS code used to produce the output that the **SAS Code** node passes on to the next node in the process flow diagram.
  - **Train Graphs** — displays graphs that are generated by SAS\GRAPH commands from within the Train code pane.
  - **Report Graph**s — displays graphs that are generated by SAS\GRAPH commands from within the Report code pane.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **SAS Code** node does not generate PMML.
- **Assessment** — appears only if the Tool Type property is set to MODEL. By default, it contains a submenu item for Fit Statistics. You can, however, generate other items by including the appropriate type code in the node.
- **Custom Reports** — appears as an item in the menu when you generate custom reports with %EM_REPORT. The title in the menu, by default, is Custom Reports, but that can be changed by specifying the BLOCK argument of the macro %EM_REPORT.
- **Table** — displays a table that contains the underlying data that is used to produce a chart.
- **Plot** — use the Graph wizard to modify an existing Results plot or create a Results plot of your own.

SAS Code Node Examples

Example 1A: Writing New SAS Code

Follow these steps to write SAS code to compare the distributions of interval variables in the training and validation data sets.

1. Define a data source for SAMPSIO.HMEQ. Ensure that the measurement level is **binary** for BAD, and **nominal** for JOB and REASON. Other variables have the level of **interval**.
2. Add an **Input Data** node by dragging \ the HMEQ data source into the diagram workspace.
3. Add a **Data Partition** node and connect it to the **Input Data** node.
4. Run the **Data Partition** node.
5. Add a **SAS Code** node and connect it to the **Data Partition** node. Your process flow diagram should look like the following:

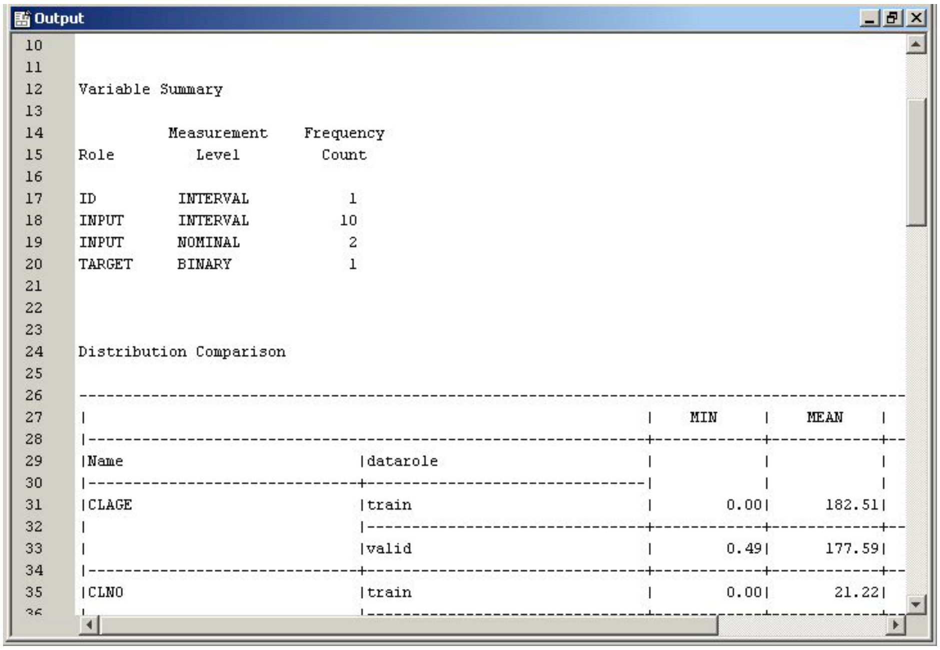


6. Select the **SAS Code** node and click the ellipsis icon ... that corresponds to the Code Editor property to open the editor.
7. Enter the following code in the Training Code pane.

```
/* perform PROC MEANS on interval variables in training data */
/* output the results to data set named t                    */

proc means data=&em_import_data noprint;
   var %em_interval;
   output out=t;
run;

/* drop unneeded variables and observations  */

data t;
set t;
   drop _freq_ _type_;
   where _stat_ ne 'N';
run;

/* transpose the data set  */

proc transpose data=t out=tt;
     id _stat_;
run;

/* add a variable to identify data partition */

data tt;
set tt;
   length datarole $8;
   datarole='train';
run;

/* perform PROC MEANS on interval variables in validation data */
/* output the results to data set named v                    */

proc means data=&em_import_validate noprint;
     var %em_interval;
     output out=v;
run;

/* drop unneeded variables and observations  */

data v;
set v;
   drop _freq_ _type_;
   where _stat_ ne 'N';
run;

/* transpose the data set  */

proc transpose data=v out=tv;
   id _stat_;
run;
```

```
/* add a variable to identify data partition */

data tv;
set tv;
   length datarole $8;
   datarole='valid';
run;

/* append the validation data results */
/* to the training data results       */

proc append base=tt data=tv;
run;

/* register the key Comp and      */
/* create a permanent data set so */
/* that the data set can be used  */
/* later in Report code           */

%em_register(key=Comp, type=data);

data &em_user_Comp;
  length _name_ $12;
  label _name_ = 'Name';
  set tt;
  cv=std/mean;
run;

/* tabulate the results */

proc tabulate data=&em_user_Comp;
   class _name_ datarole;
   var min mean max std cv;
   table _name_*datarole, min mean max std cv;
   keylabel sum=' ';
   title 'Distribution Comparison';
run;
```

8. Run the **SAS Code** node and view the results. In the SAS Code Results window, the Output window displays the tabulated comparison of the variables' distributions of the training and validation data sets.



```



Variable Summary

           Measurement    Frequency
Role          Level         Count

ID          INTERVAL          1
INPUT       INTERVAL         10
INPUT       NOMINAL           2
TARGET      BINARY            1



Distribution Comparison

-------------------------------------------------------------------------------------------
|                                                           |    MIN     |    MEAN    |
|-----------------------------------------------------------+------------+------------+--
|Name                         |datarole                     |            |            |
|-----------------------------+-----------------------------|            |            |
|CLAGE                        |train                        |        0.00|      182.51|
|                             |-----------------------------+------------+------------+--
|                             |valid                        |        0.49|      177.59|
|-----------------------------+-----------------------------+------------+------------+--
|CLNO                         |train                        |        0.00|       21.22|
|                             |-----------------------------+------------+------------+--
```

Example 1B: Adding Logical Evaluation

The SAS code in Example 1a generates error messages if no validation data set exists. You use conditional logic within a macro to make the program more robust. To do so, follow these steps.

1. Open the Code Editor.

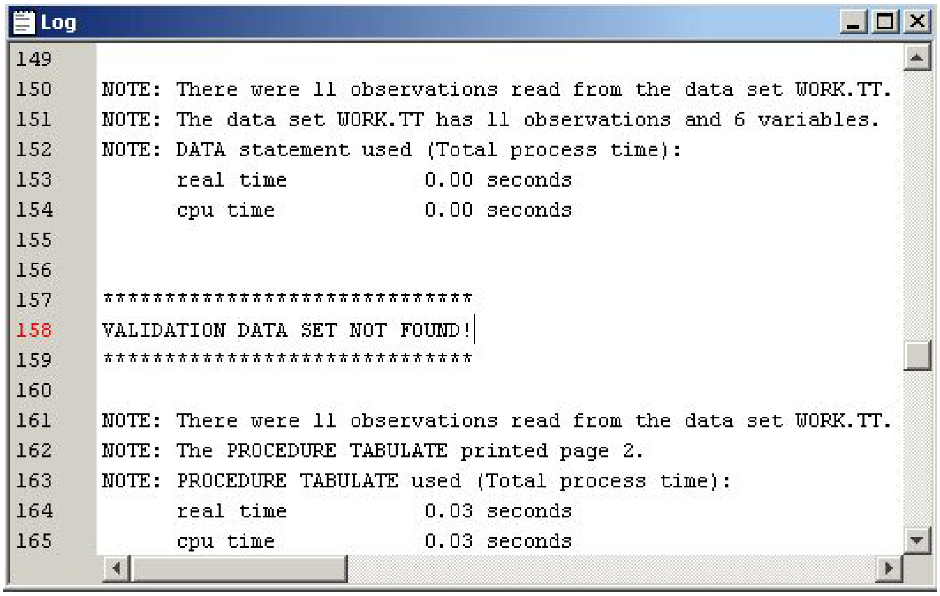
2. Add the following code shown in blue in the Training Code pane. The %EVAL function evaluates logical expressions and returns a value of either 1(for true) or 0 (for false). In this example, it checks whether a value has been assigned to the macro variable &EM_IMPORT_VALIDATE . If a validation data set exists, &EM_IMPORT_VALIDATE is assigned a value of the name of the validation data set, and the macro variable, &cv, is set to 1. The new code checks the existence of a validation data set before it calculates the values of minimum, mean, max, and standard deviation of variables. If no validation data set exists, it writes a note to the Log window.

```
%macro intcompare();
   %let cv=0;
   %if "em_import_validate" ne "" and
       (%sysfunc(exist(&em_import_validate)) or
        %sysfunc(exist(&em_import_validate, VIEW))) %then
        %let cv=1;

   proc means data=&em_import_data noprint;
      var %em_interval;
      output out=t;
   run;

   data t;
      set t;
      drop _freq_ _type_;
      where _stat_ ne 'N';
   run;

   proc transpose data=t out=tt;
      id _stat_;
   run;

   data tt;
      set tt;
      length datarole $8;
      datarole='train';
   run;

   %if &cv %then %do;

      proc means data=&em_import_validate noprint;
         var %em_interval;
         output out=v;
      run;

      data v;
         set v;
         drop _freq_ _type_;
         where _stat_ ne 'N';
      run;

      proc transpose data=v out=tv;
         id _stat_;
      run;

      data tv;
         set tv;
         length datarole $8;
         datarole='valid';
      run;

      proc append base=tt data=tv;
      run;

      %em_register(key=Comp, type=data);

      data &em_user_Comp;
         length _name_ $12;
         label _name_ = 'Name';
         set tt;
         cv=std/mean;
      run;

   %end;

   %else %do;

      %put &em_codebar;
      %put %str(VALIDATION DATA SET NOT FOUND!);
      %put &em_codebar;

   %end;

   proc tabulate data=&em_user_Comp;
      class _name_ datarole;
```

```
      var min mean max std cv;
      table _name_*datarole, min mean max std cv;
      keylabel sum=' ';
      title 'Distribution Comparison';
   run;

%mend intcompare;
%intcompare();
```

3. In the **Data Partition** node properties panel, change the Data Set Allocation property for the training and validation data sets to 70 and 0, respectively.

4. Run the **SAS Code** node and view the results. The Output window in the Results window displays statistics for the training data set only. Open the Log window within the Results window. The text “VALIDATION DATA SET NOT FOUND!” displays in the Log window.



```

NOTE: There were 11 observations read from the data set WORK.TT.
NOTE: The data set WORK.TT has 11 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


******************************
VALIDATION DATA SET NOT FOUND!
******************************


NOTE: There were 11 observations read from the data set WORK.TT.
NOTE: The PROCEDURE TABULATE printed page 2.
NOTE: PROCEDURE TABULATE used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds
```

Example 1C: Adding Report Elements

In parts 1a and 1b, the comparison of variables distributions is displayed in the Output window. In addition, you might want to include the tabulated comparison in a SAS table view and to create a plot of some of the statistics. To do so, follow the steps below.

1. In the **Data Partition** node properties panel, change the Data Set Allocation property for the training and validation data sets back to 40 and 30, respectively.

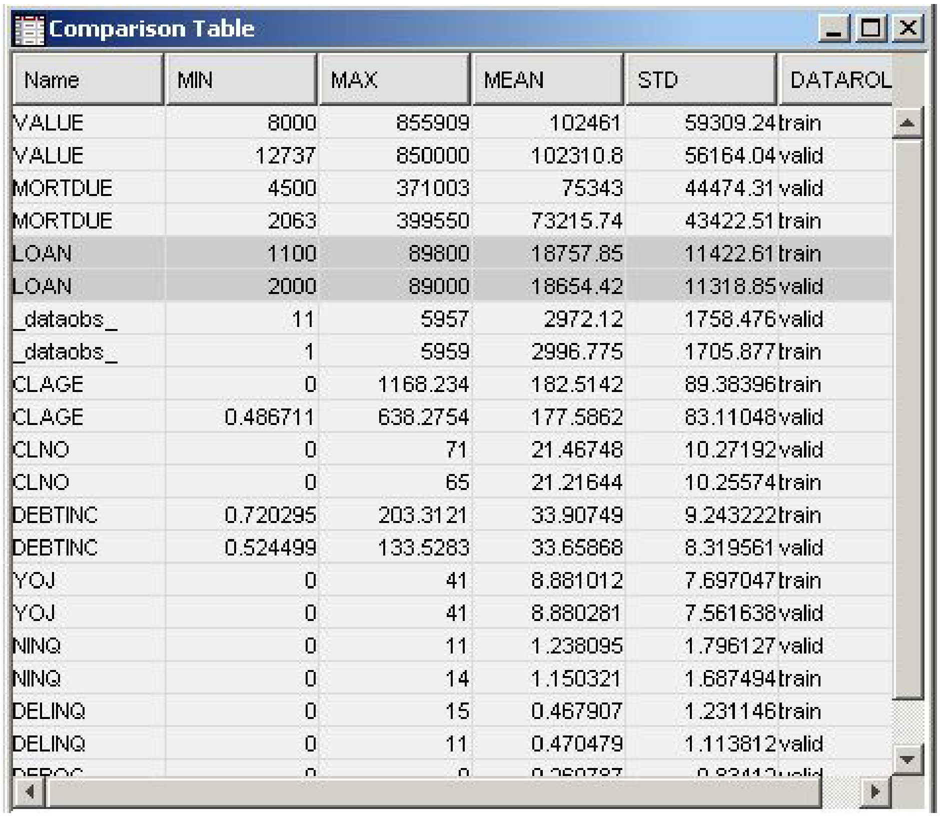
2. Open the Code Editor.

3. Enter the following code in the Report Code pane.

```
/* initialize the &em_user_Comp macro variable  */

%em_getname(key=Comp, type=data);

/*** Save Results with EM Name ***/

proc sort
data=&em_user_Comp
out=&em_user_Comp;
    by descending cv;
```

```
run;

/*** Add to EM Results ***/

%em_report(key=Comp,
       viewtype=Data,
       block=Compare,
       description=Comparison Table);

%em_report(key=Comp,
       viewtype=Bar,
       x=_name_,
       freq=cv,
       block=Compare,
       where= datarole eq 'train',
       autodisplay=Y,
       description=Training Data CV Plot);

%em_report(key=Comp,
       viewtype=Bar,
       x=_name_,
       freq=cv,
       block=Compare,
       where= datarole eq 'valid',
       autodisplay=Y,
       description=Validation Data CV Plot);

run;
```

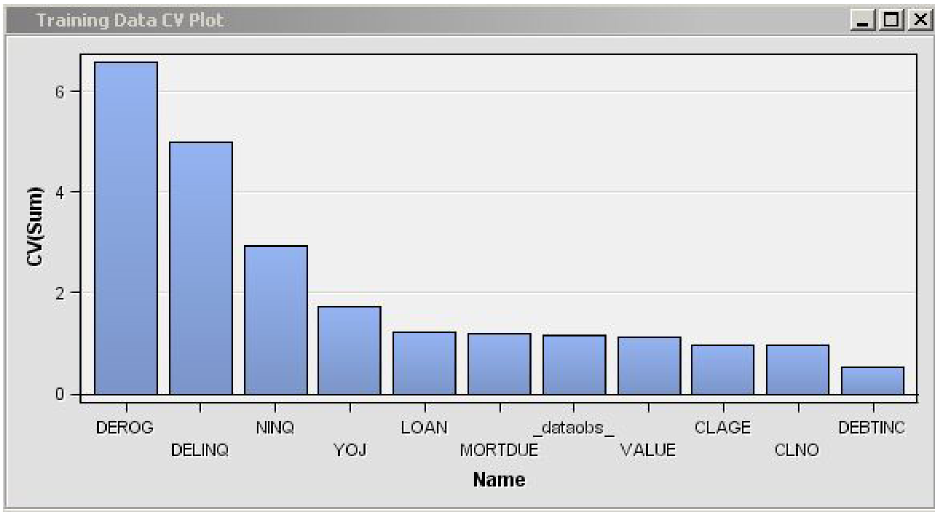
4. Run the **SAS Code** node and view results.
5. In the SAS Code Results window, select **View** ⇨ **Compare** ⇨ **Comparison Table** from the main menu. The following Comparison Table window appears. The table is sorted by the values of standard deviation in descending order.



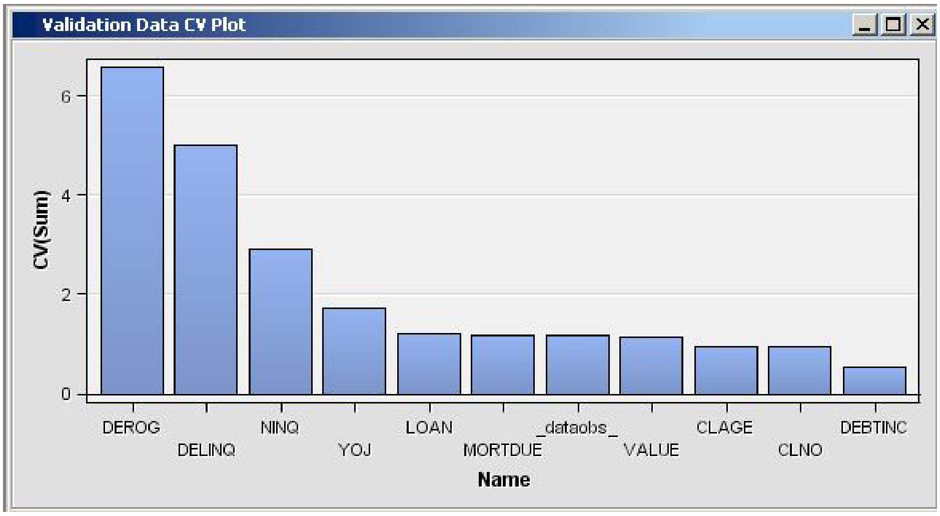
Comparison Table

Name	MIN	MAX	MEAN	STD	DATAROL
VALUE	8000	855909	102461	59309.24	train
VALUE	12737	850000	102310.8	56164.04	valid
MORTDUE	4500	371003	75343	44474.31	valid
MORTDUE	2063	399550	73215.74	43422.51	train
LOAN	1100	89800	18757.85	11422.61	train
LOAN	2000	89000	18654.42	11318.85	valid
dataobs	11	5957	2972.12	1758.476	valid
dataobs	1	5959	2996.775	1705.877	train
CLAGE	0	1168.234	182.5142	89.38396	train
CLAGE	0.486711	638.2754	177.5862	83.11048	valid
CLNO	0	71	21.46748	10.27192	valid
CLNO	0	65	21.21644	10.25574	train
DEBTINC	0.720295	203.3121	33.90749	9.243222	train
DEBTINC	0.524499	133.5283	33.65868	8.319561	valid
YOJ	0	41	8.881012	7.697047	train
YOJ	0	41	8.880281	7.561638	valid
NINQ	0	11	1.238095	1.796127	valid
NINQ	0	14	1.150321	1.687494	train
DELINQ	0	15	0.467907	1.231146	train
DELINQ	0	11	0.470479	1.113812	valid
DEROG	0	0	0.260787	0.83412	valid

6. In the Results window, select **View** ⇨ **Compare** ⇨ **Training Data CV Plot** ⇨ from the main menu. The following Training Data CV Plot window appears. The plot displays a bar chart of the coefficient of variation for each variable in the training data set.



In the Results window, select **View** ⇨ **Compare** ⇨ **Validation Data CV Plot** from the main menu. The following Validation Data CV Plot window appears. The plot displays a bar chart of the coefficient of variation for each variable in the training data set.



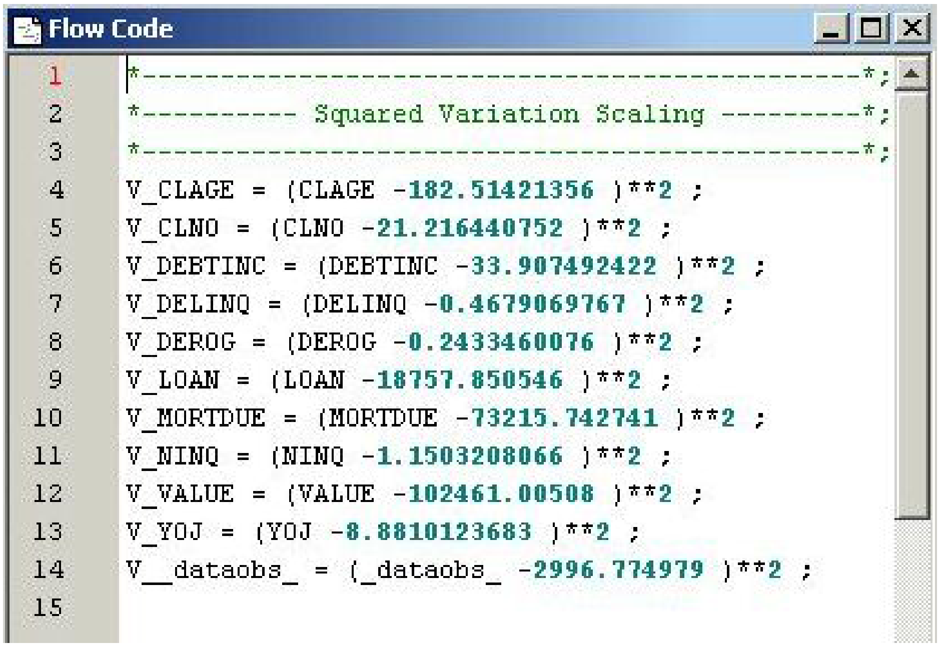
Example 1D: Adding Score Code

Suppose you want to generate scoring code to rescale the variables to their deviation from the mean. SAS Enterprise Miner recognizes two types of SAS scoring code, Flow scoring code and Publish scoring code. Flow scoring code is used to score SAS data tables inside the process flow diagram. Publish scoring code is used to publish the SAS Enterprise Miner model to a scoring system outside the process flow diagram. To generate both types of scoring code, follow the steps below.

1. Open the Code Editor.
2. Add the following code to your SAS program in the Training Code pane.

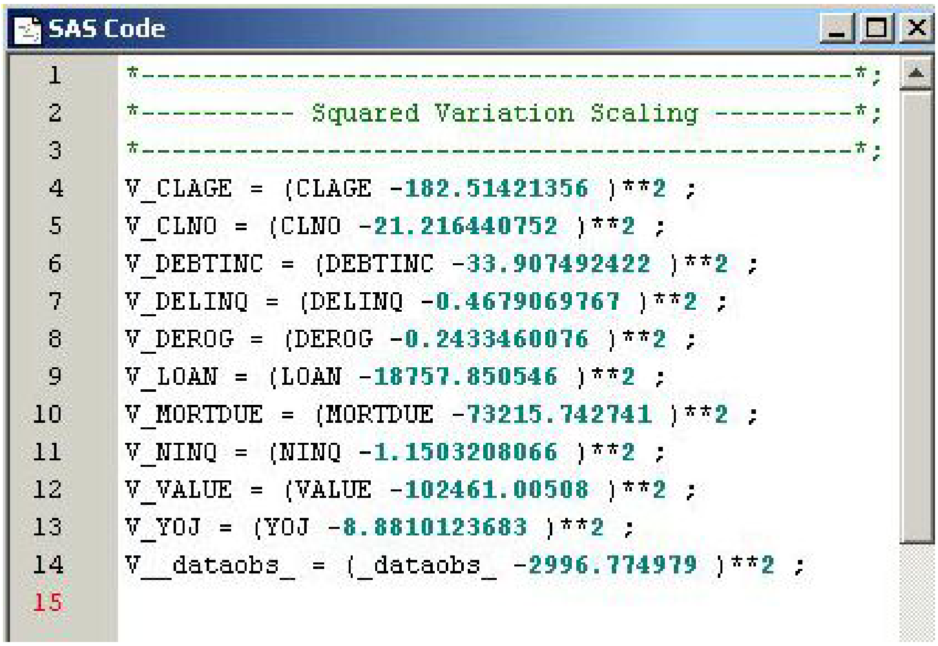
```
/* Add Score Code */

%macro scorecode(file);
data _null_;
   length var $32;
   filename X "&file";
   FILE X;
   set &em_user_Comp(where=(datarole eq 'train'));

if _N_ eq 1 then do;
   put '*---------------------------------------------*;';
   put '*---------- Squared Variation Scaling ---------*;';
   put '*---------------------------------------------*;';
end;

var=strip('V_' !! _name_);
put var '= (' _name_ '-' mean ')**2 ;' ;
run;

%mend scorecode;
%scorecode(&em_file_emflowscorecode);
%scorecode(&em_file_empublishscorecode);
```

3. Run the **SAS Code** node and open the Results window.
4. Select **View** ⇨ **SAS Results** ⇨ **Flow Code** from the main menu.



```
*------------------------------------------------*;
*---------- Squared Variation Scaling ---------*;
*------------------------------------------------*;
V_CLAGE = (CLAGE -182.51421356 )**2 ;
V_CLNO = (CLNO -21.216440752 )**2 ;
V_DEBTINC = (DEBTINC -33.907492422 )**2 ;
V_DELINQ = (DELINQ -0.4679069767 )**2 ;
V_DEROG = (DEROG -0.2433460076 )**2 ;
V_LOAN = (LOAN -18757.850546 )**2 ;
V_MORTDUE = (MORTDUE -73215.742741 )**2 ;
V_NINQ = (NINQ -1.1503208066 )**2 ;
V_VALUE = (VALUE -102461.00508 )**2 ;
V_YOJ = (YOJ -8.8810123683 )**2 ;
V__dataobs_ = (_dataobs_ -2996.774979 )**2 ;
```

5. To view the publish scoring code, select **View Scoring SAS Code** from the main menu.



```
*------------------------------------------------*;
*---------- Squared Variation Scaling ---------*;
*------------------------------------------------*;
V_CLAGE = (CLAGE -182.51421356 )**2 ;
V_CLNO = (CLNO -21.216440752 )**2 ;
V_DEBTINC = (DEBTINC -33.907492422 )**2 ;
V_DELINQ = (DELINQ -0.4679069767 )**2 ;
V_DEROG = (DEROG -0.2433460076 )**2 ;
V_LOAN = (LOAN -18757.850546 )**2 ;
V_MORTDUE = (MORTDUE -73215.742741 )**2 ;
V_NINQ = (NINQ -1.1503208066 )**2 ;
V_VALUE = (VALUE -102461.00508 )**2 ;
V_YOJ = (YOJ -8.8810123683 )**2 ;
V__dataobs_ = (_dataobs_ -2996.774979 )**2 ;
```

Example 1E: Modifying Variables Metadata

New variables have been added to the model, and the original variables need to be removed to avoid duplicating terms in the final model. The variables can be dropped from the incoming tables or they can be given a Role of REJECTED in the exported metadata. You follow these steps to generate SAS code to modify the exported metadata tables. SAS code is used to create rules that can have more than one condition. Even though the training, validation, and test data sets are processed in the flow, you need to modify only the metadata for the exported training data set. You modify the metadata for the validation and test data sets only when different variables are created on the validation or test data set.

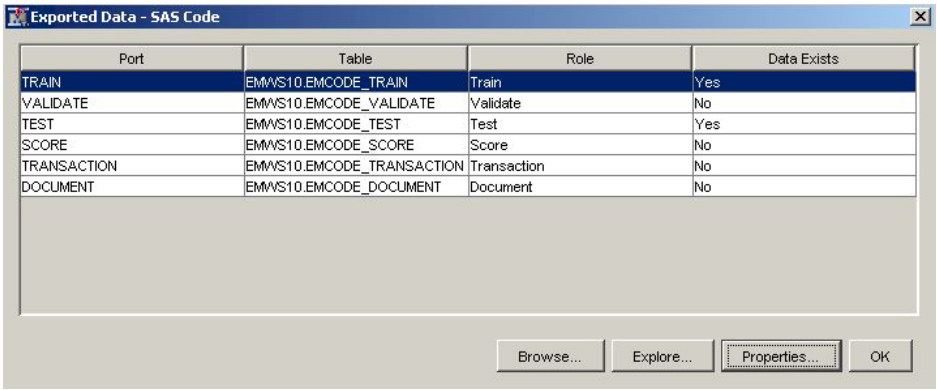
1. Open the Code Editor.

2. Add the following code to your SAS program in the Training Code pane.

```
/* Modify Exported Training Metadata */

data _null_;
length string $34;
filename X "&em_file_cdelta_train";
FILE X;
set &em_user_Comp(
   where=(datarole eq 'train'));

/* Reject Original Variable */

string = upcase('"'!!strip(_NAME_)!!'"');
put 'if upcase(NAME) eq ' string ' then role="REJECTED" ;' ;


/* Modify New Variables */

var=upcase(strip('V_' !! _name_ ));
string = '"'!!strip(var)!!'"';
put 'if upcase(NAME) eq ' string ' then do ;' ;
put '      role="INPUT" ;' ;
put '      level= "INTERVAL" ;' ;
put '      comment= "Squared Variation" ;' ;
put 'end ;' ;
run;
```

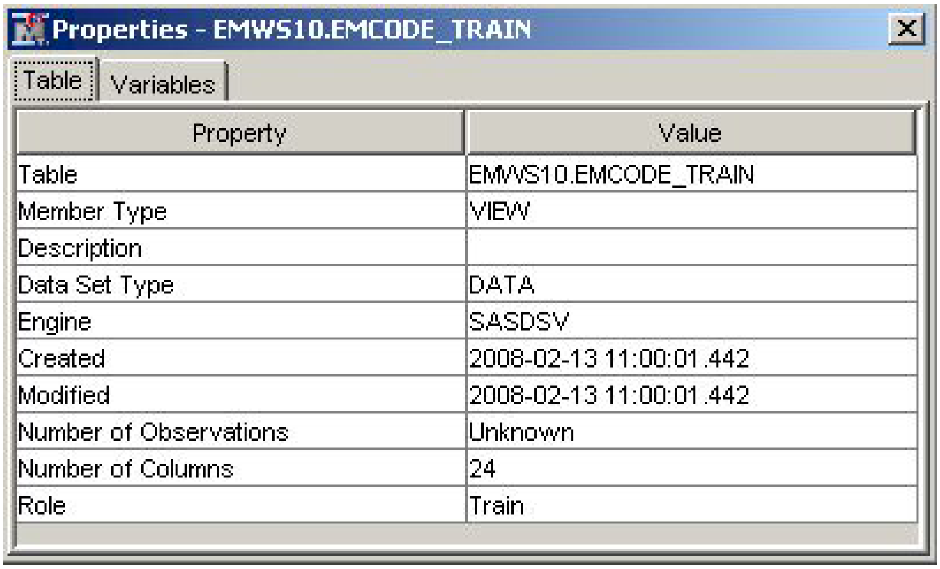
3. Run the **SAS Code** node and do not view the results. Close the Code Editor.

4. From the **SAS Code** node general properties, click the ... icon of the Exported Data property.



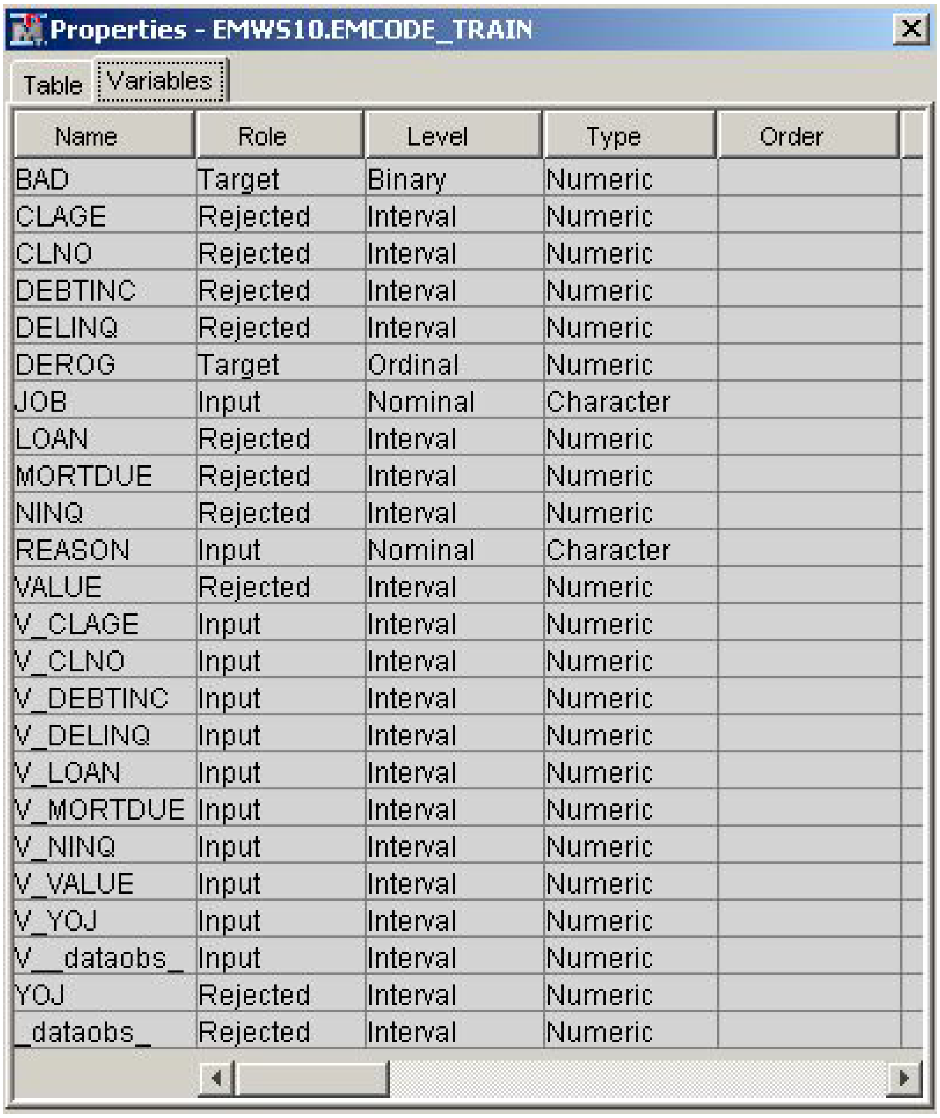
Port	Table	Role	Data Exists
TRAIN	EMWS10.EMCODE_TRAIN	Train	Yes
VALIDATE	EMWS10.EMCODE_VALIDATE	Validate	No
TEST	EMWS10.EMCODE_TEST	Test	Yes
SCORE	EMWS10.EMCODE_SCORE	Score	No
TRANSACTION	EMWS10.EMCODE_TRANSACTION	Transaction	No
DOCUMENT	EMWS10.EMCODE_DOCUMENT	Document	No

Select the Train data set from the Port column of the table. Click **Properties** at the bottom of the window.



Property	Value
Table	EMWS10.EMCODE_TRAIN
Member Type	VIEW
Description	
Data Set Type	DATA
Engine	SASDSV
Created	2008-02-13 11:00:01.442
Modified	2008-02-13 11:00:01.442
Number of Observations	Unknown
Number of Columns	24
Role	Train

Click the **Variables** tab.



Name	Role	Level	Type	Order
BAD	Target	Binary	Numeric	
CLAGE	Rejected	Interval	Numeric	
CLNO	Rejected	Interval	Numeric	
DEBTINC	Rejected	Interval	Numeric	
DELINQ	Rejected	Interval	Numeric	
DEROG	Target	Ordinal	Numeric	
JOB	Input	Nominal	Character	
LOAN	Rejected	Interval	Numeric	
MORTDUE	Rejected	Interval	Numeric	
NINQ	Rejected	Interval	Numeric	
REASON	Input	Nominal	Character	
VALUE	Rejected	Interval	Numeric	
V_CLAGE	Input	Interval	Numeric	
V_CLNO	Input	Interval	Numeric	
V_DEBTINC	Input	Interval	Numeric	
V_DELINQ	Input	Interval	Numeric	
V_LOAN	Input	Interval	Numeric	
V_MORTDUE	Input	Interval	Numeric	
V_NINQ	Input	Interval	Numeric	
V_VALUE	Input	Interval	Numeric	
V_YOJ	Input	Interval	Numeric	
V__dataobs_	Input	Interval	Numeric	
YOJ	Rejected	Interval	Numeric	
dataobs	Rejected	Interval	Numeric	

The original interval input variables now have a Role of REJECTED. The new variables (V_xxx) have a Role of INPUT.

Example 2: Writing SAS Code to Create Predictive Models

This example shows you additional features of the **SAS Code** node.

1. Define a data source for SAMPSIO.DMAGECR (German Credit) and set the binary variable GOOD_BAD as the target. Use the Advanced Advisor and select **Yes** when you are prompted to build models by using the values of the decisions.
2. Add an **Input Data** node by dragging the data source DMAGECR onto the diagram workspace.
3. Add a **SAS Code** node to the diagram workspace and connect it to the **Input Data** node.
4. Change the value of the Tool Type property to Model in the properties panel.
5. Select the **SAS Code** node and click the ... icon in the Code Editor property to open the Code Editor.
6. In the Training Code pane, enter the following code:

```
/* Register User Files */

%em_register(
   key=Fit,
   type=Data);

%em_register(
   key=Est,
   type=Data);

/* Training Regression Model */

/* Create a DMDB database */

%em_dmdb(out=1);

/* Fit logistic regression model  */
/* using macro %em_dmreg from the */
/* sashelp.emutil catalog         */

%em_dmreg(
   selection=Stepwise,
   outest=&em_user_Est,
   outselect=Work.Outselect);

/* Work.Outselect contains the names of REJECTED variables    */
/* &em_user_Est contains parameter estimates and t statistics */
/* for each of the stepwise models                            */


/* Modify Exported Metadata */

data _null_;
length string $34;
filename X "&em_file_cdelta_train";
FILE X;
```

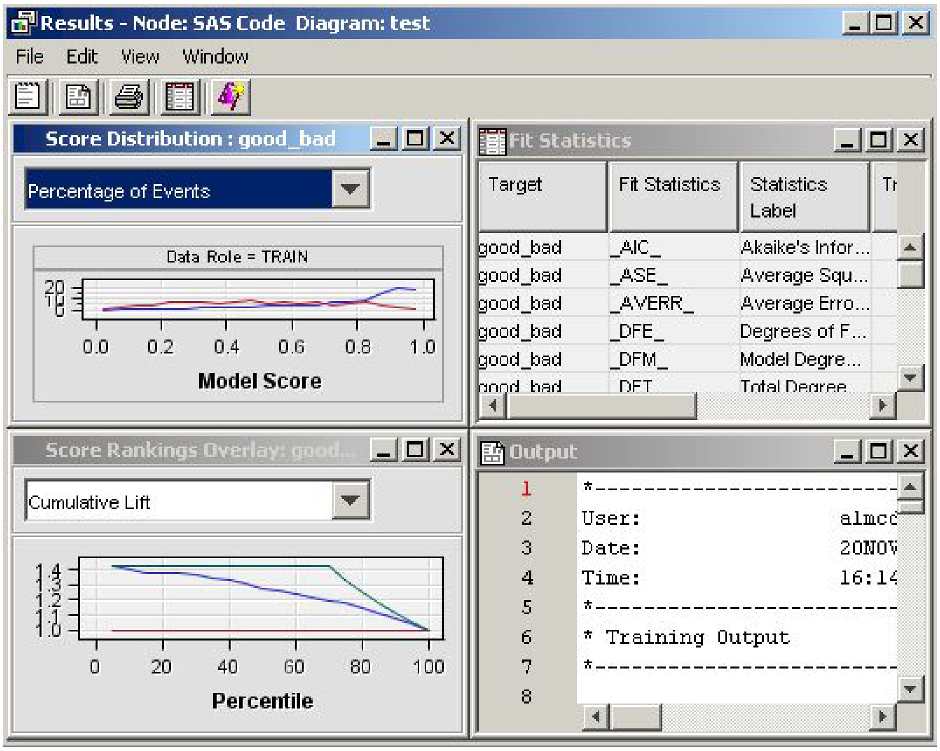
```
if _N_=1 then do;
   put "if ROLE in ('INPUT','REJECTED') then do;";
   put "if NAME in (";
   end;

set Work.Outselect end=eof;
string = '"'!!trim(left(TERM))!!'"';
put string;

if eof then do;
   put ') then role="INPUT";';
   put 'else role="REJECTED";';
   put 'end;';
end;

run;
```

7. In the Report Code editor, enter the following code:

```
/* Generate Graphs */

proc univariate data=&em_import_data noprint;
     class &em_dec_target;
     histogram %em_interval_input;
run;
```

SAS graphs are automatically copied from the WORK.GSEG catalog, and GIF files are created and stored in the node's REPORTGRAPH subfolder. For example, suppose your projects are stored in a folder named **C:\EMPROJECTS**. If your project name is SASCODE and your diagram ID is EMWS1, the GIF files will be stored in **C:\EMPROJECTS\SASCODE\WORKSPACES\EMWS1\EMCODE\REPORTGRAPH**.

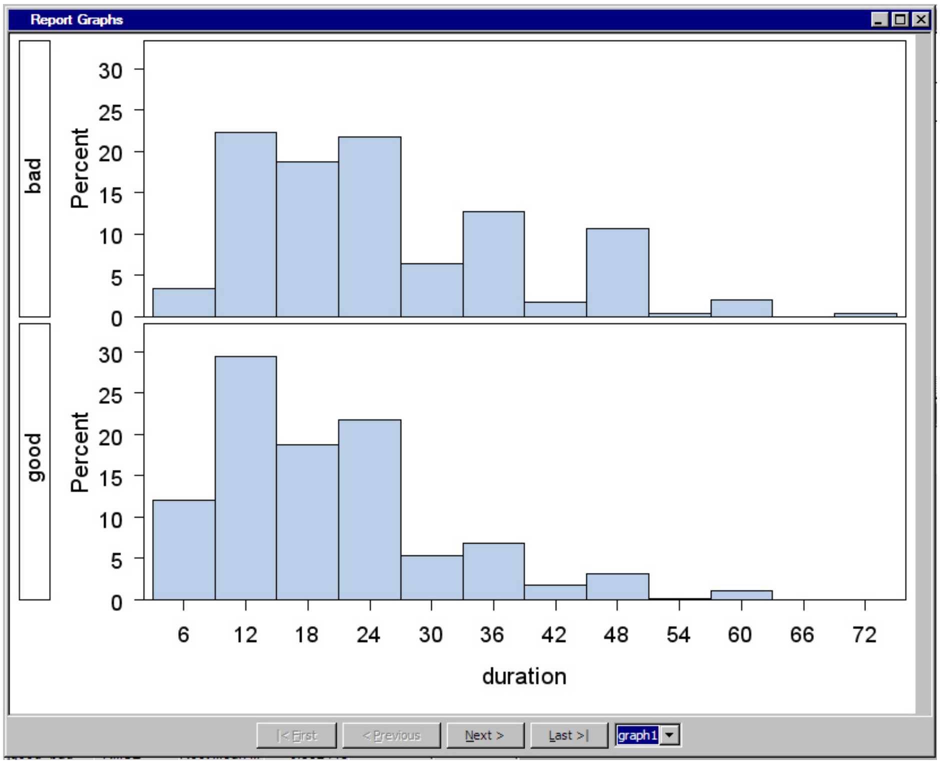
8. Run the **SAS Code** node and view the results. Open the Score Distribution chart. The following display shows an example of the SAS Code results window.



Target	Fit Statistics	Statistics Label	Tr
good_bad	_AIC_	Akaike's Infor...	
good_bad	_ASE_	Average Squ...	
good_bad	_AVERR_	Average Erro...	
good_bad	_DFE_	Degrees of F...	
good_bad	_DFM_	Model Degre...	
good_bad	_DFT_	Total Degree	

```
*---------------------------
User:                    almcc
Date:                    20NOV
Time:                    16:14
*---------------------------
* Training Output
*---------------------------
```

Standard results of a model node are displayed. The SAS Code is registered as a MODEL tool at the beginning of the SAS code. Therefore, fit statistics, and plots of score distribution and score rankings are automatically displayed. Select **View** ⇨ **SAS Results** ⇨ **Report Graphs** from the main menu. The Report Graphs window appears and displays the output from the PROC UNIVARIATE statement. The PROC UNIVARIATE statement produces histograms of each input interval input for both target levels.



9. Close the Results window. Add another **SAS Code** node to the diagram workspace and connect it to the **Input Data** node.

10. Change the value of the Tool Type property to Model in the Properties panel.

11. Open the Code Editor for the newly added **SAS Code** node and copy the following code in the Training Code editor. The code is similar to that in step 6, but uses PROC ARBOR to create a decision tree model. The PROC ARBOR step is encapsulated in the %EM_ARBOR macro.

```
/* Registering User Files */

%em_register(key=MODEL, type=DATA);
%em_register(key=IMPORTANCE, type=DATA);
%em_register(key=NODES, type=DATA);
%em_register(key=LEAFSTATS, type=DATA);

/* Training Decision Tree Model */

%em_arbor(
   criterion=probchisq,
   alpha=0.2,
   outmodel=&EM_USER_MODEL,
   outimport=&EM_USER_IMPORTANCE,
   outnodes=&EM_USER_NODES);
```

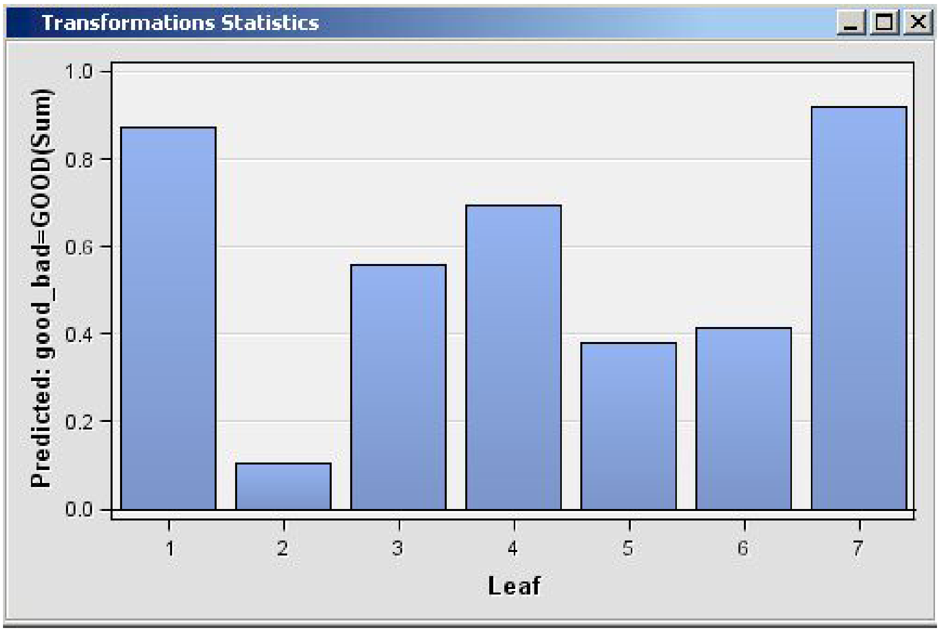
```
/**************************************************************************/
/* CRITERION    = criterion (VARIANCE, PROBF, ENTROPY, GINI, PROBCHISQ)   */
/* ALPHA        = alpha value; used with criterion = PROBCHISQ or PROBF   */
/*                (default=0.20)                                          */
/* OUTMODEL     = tree data set; encode info used in the INMODEL option   */
/* OUTIMPORT    = importance data set; contains variable importance       */
/* OUTNODES     = nodes data set; contains node information               */
/**************************************************************************/

/* Modifying Exported Metadata */

data _null_;
length string $200;
filename X "&EM_FILE_CDELTA_TRAIN";
file X;
set &EM_USER_IMPORTANCE
   end=eof;

if IMPORTANCE =0  then do;
    string = 'if NAME="'!!trim(left(name))!!'" then do;';
    put string;
    put 'ROLE="REJECTED";';
    string = 'COMMENT="'!!"&EM_NODEID"!!':
                  Rejected because of low importance value";';
    put string;
    put 'end;';
end;

else do;
   string = 'if NAME="'!!trim(left(name))!!'" then ROLE="INPUT";';
   put string;
end;

if ^eof then
   put 'else';
run;
```

12. Type the following code in the Report Code pane:

```
/* Generating Reports */

/* Initialize &EM_PRED with the name of the */
/* target=1 prediction variable             */

data _null_;
   set &em_dec_decmeta;
   where _TYPE_ eq "PREDICTED" AND LEVEL eq "GOOD";
   call symput("EM_PRED",VARIABLE);
run;

/* Reinitialize registered keys  */

%em_getname(key=LEAFSTATS, type=data);
%em_getname(key=NODES, type=data);
%em_getname(key=IMPORTANCE, type=data);
```

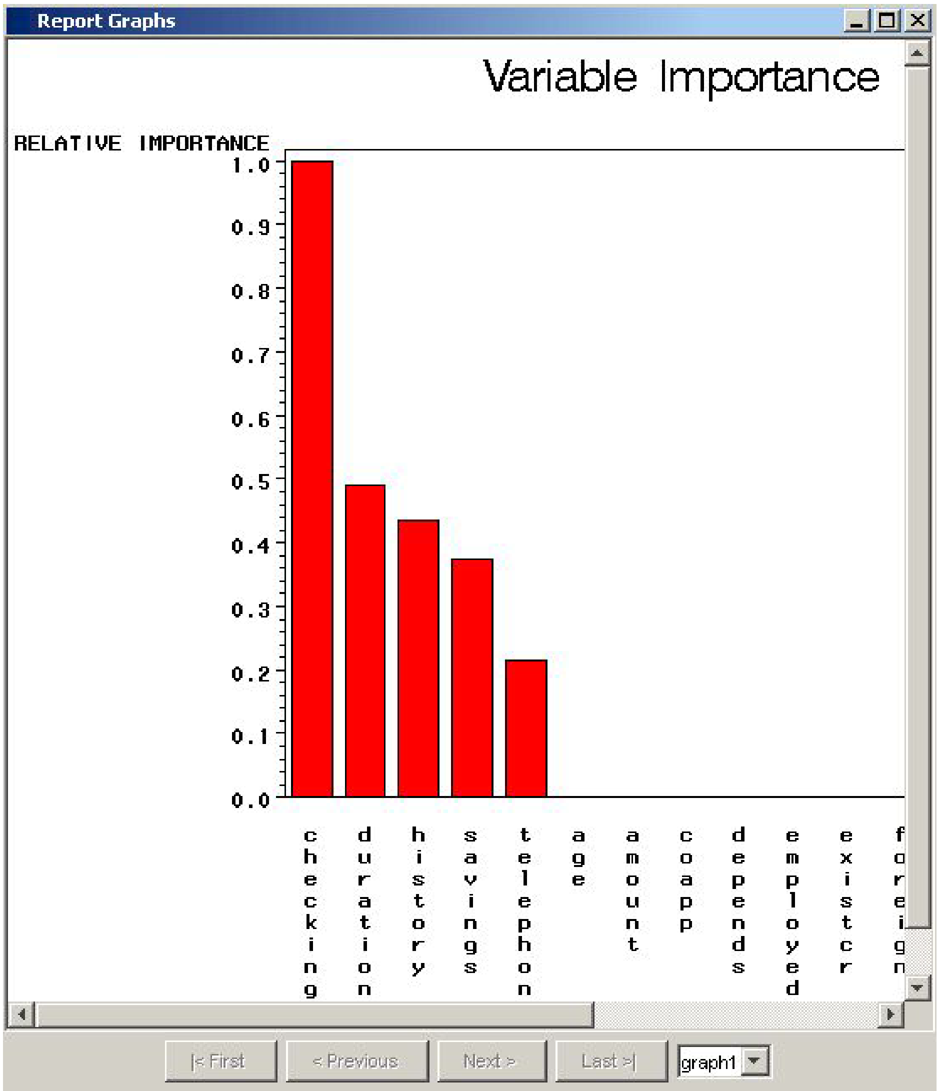
```
/* retrieve the predicted variables data set */

data &EM_USER_LEAFSTATS;
   set &EM_USER_NODES(
   keep=LEAF N NPRIORS P_: I_: U_:);
   where LEAF ne .;
   format LEAF 3.;
run;

/* plot the target prediction for each leaf */

%EM_REPORT(key=LEAFSTATS,
           description=STATISTICS,
           viewtype=BAR,
           freq=&EM_PRED,
           x=LEAF);

/* Generating Graphs */

%em_getname(key=IMPORTANCE, type=data);

/* Plot the Importance of the Individual Variables */

proc gchart data=&EM_USER_IMPORTANCE;
     vbar name/sumvar=importance discrete descending;
     title 'Variable Importance';
run;
title;
quit;
```

13. Run the **SAS Code** node and open the Results window. Select **View Custom Reports Transformation Statistics** from the main menu. The Transformation Statistics plot is displayed:



14. Select **View SAS Results Report Graphs** from the main menu. The Report Graphs window appears and displays the output from the PROC GCHART statement. The

PROC GCHART statement produces bar charts of the importance value of each input variable.

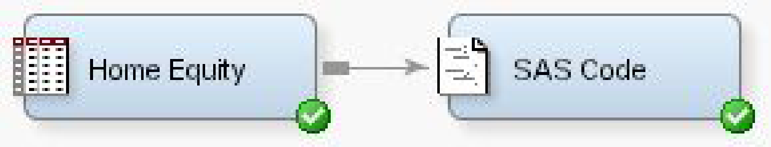


Examples Using %EM_REPORT

Bar Charts

This example demonstrates how to generate a simple bar chart and progressively add features.

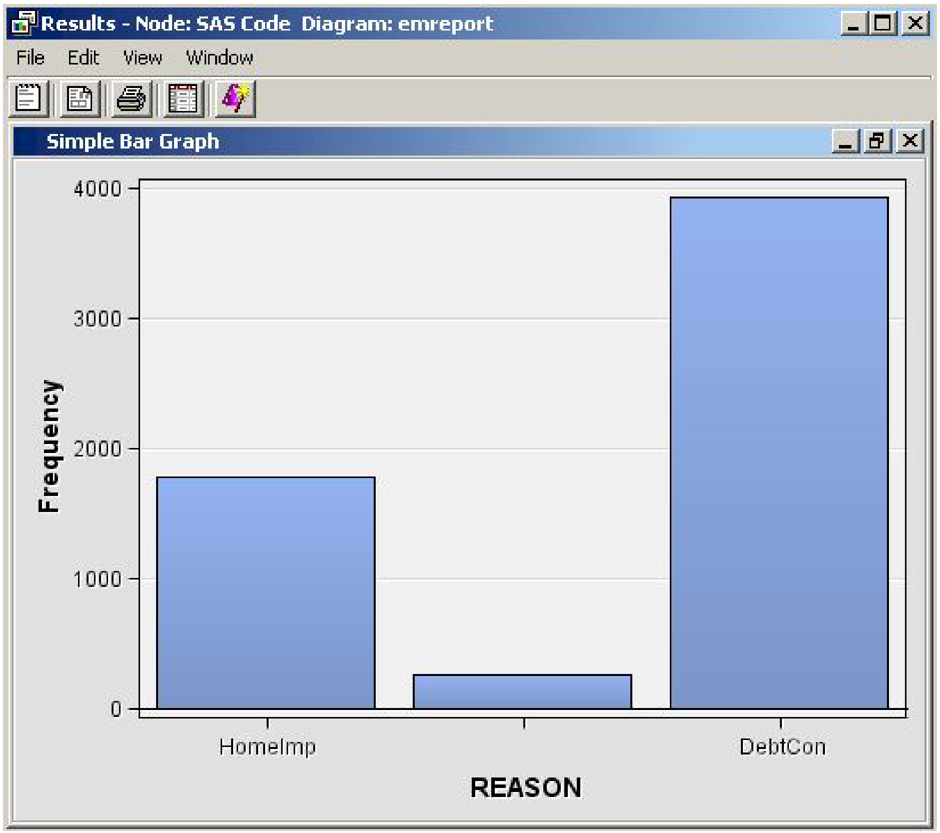
1. Create a new diagram.
2. Add an input data source to the diagram. Use the Home Equity data set from the SAMPSIO library.
3. Add a **SAS Code** node to the diagram and connect it to the **Home Equity** node.



4. Click the **SAS Code** node and open the Code Editor.
5. Enter the following code in the Report Code pane:

```
%em_register(type=Data,key=Example);
data &em_user_Example;
   set &em_import_Data;
run;
%em_report(
   key=Example,
   viewtype=Bar,
   x=Reason,
   autodisplay=Y,
   description=%bquote(Simple Bar Chart),
   block=%bquote(My Graphs));
```

6. Click **Run Node** ( ).
7. Click **Results** ( ). When the Results window appears, double-click the title bar of the bar chart pane and you should see the following:



Examining the code that was submitted, the first line is as follows:

```
%em_register(type=Data, key=Example);
```

The macro %EM_REGISTER registers the data key "Example". Here are the three lines:

```
data &em_user_Example;
   set &em_import_Data;
run;
```

These lines perform a SAS DATA step. By using the macro variable &em_user_Example for the data set name, the data set name is linked to the data key that was registered previously. So the general form of this macro variable is &em_user_<key>, where <key> is the argument that you supplied to %EM_REGISTER. The macro variable &EM_IMPORT_DATA used in the set statement resolves to the data set that is imported from the **Home Equity** data node that precedes the **SAS Code** node in the path. Finally, you analyze the arguments that were supplied to the macro %EM_REPORT:

```
%em_report(
   key=Example,
   viewtype=Bar,
   x=Reason,
   autodisplay=Y,
   description=%bquote(Simple Bar Chart),
   block=%bquote(My Graphs));
```

Six arguments were specified.

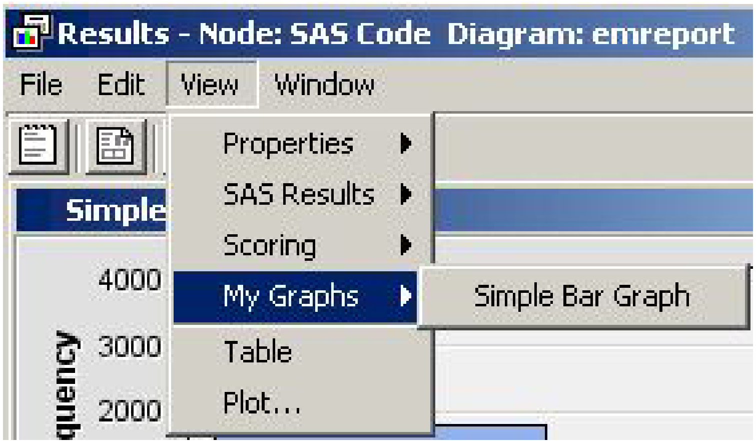
The 1st argument, KEY=Example, links the graph to the data set via the key that was registered previously using %EM_REGISTER. It is a required argument for %EM_REPORT.

The 2nd argument, VIEWTYPE=Bar, specifies that a bar chart is the desired type of graph.

The 3rd argument, X=Reason, specifies that the variable REASON is to populate the x-axis. By default, the y-axis is the frequency of the variable populating the x-axis, but as is discussed later in this section, this feature of the graph can be changed with the FREQ argument. The variable, Reason, records the reported purpose for the applicant's home equity loan.

The 4th argument, AUTODISPLAY=Y, specifies to automatically display the graph in the Results window. Without this option, you would have to use the Results window's **View** menu to display the graph.

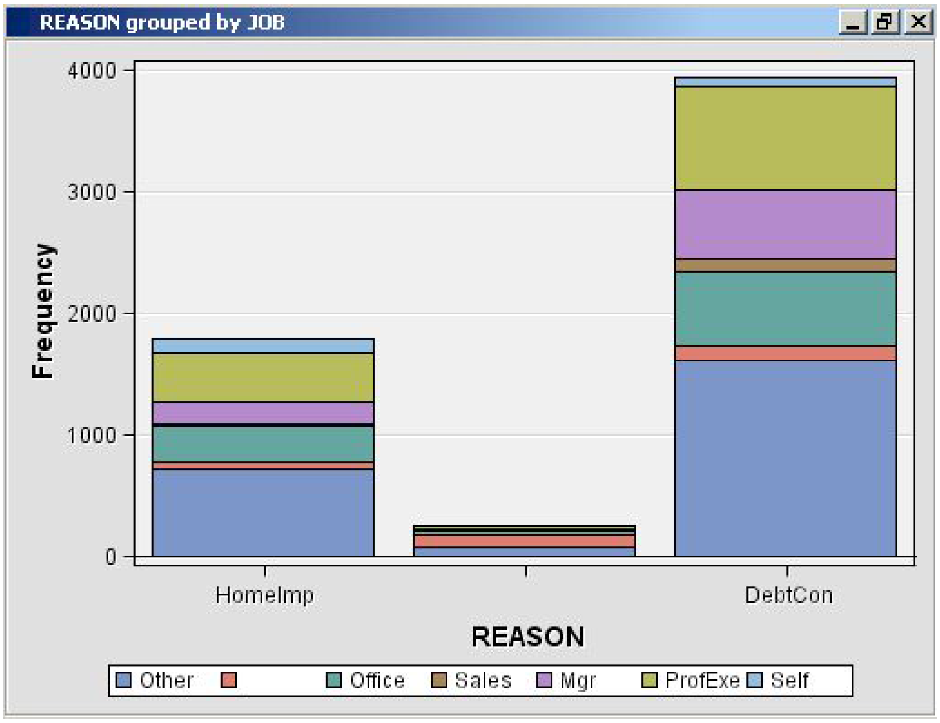
The 5th argument, DESCRIPTION=%bquote(Simple Bar Chart), specifies the text that is to appear in the title bar of the graph pane. The description is also used to populate a **View** submenu. By default, the **View** menu lists an item, known as a block, called Custom Reports. The description is listed in the block's submenu. By including the final option, BLOCK=%bquote(My Graphs), the block will be labeled "My Graphs" rather than "Custom Reports." The Description, "Simple Bar Chart" will appear as a menu item under My Graphs.



Our data set includes a variable, JOB, which records the profession of the loan applicant. Suppose you want to see how the frequencies for REASON are distributed across JOB. You can do this by specifying the GROUP option of %EM_REPORT. So, replace the call to %EM_REPORT in your code with the following:

```
%em_report(
   key=Example,
   viewtype=Bar,
   x=Reason,
   group=Job,
   autodisplay=Y,
   description=%bquote(REASON grouped by JOB),
   block=%bquote(My Graphs));
```

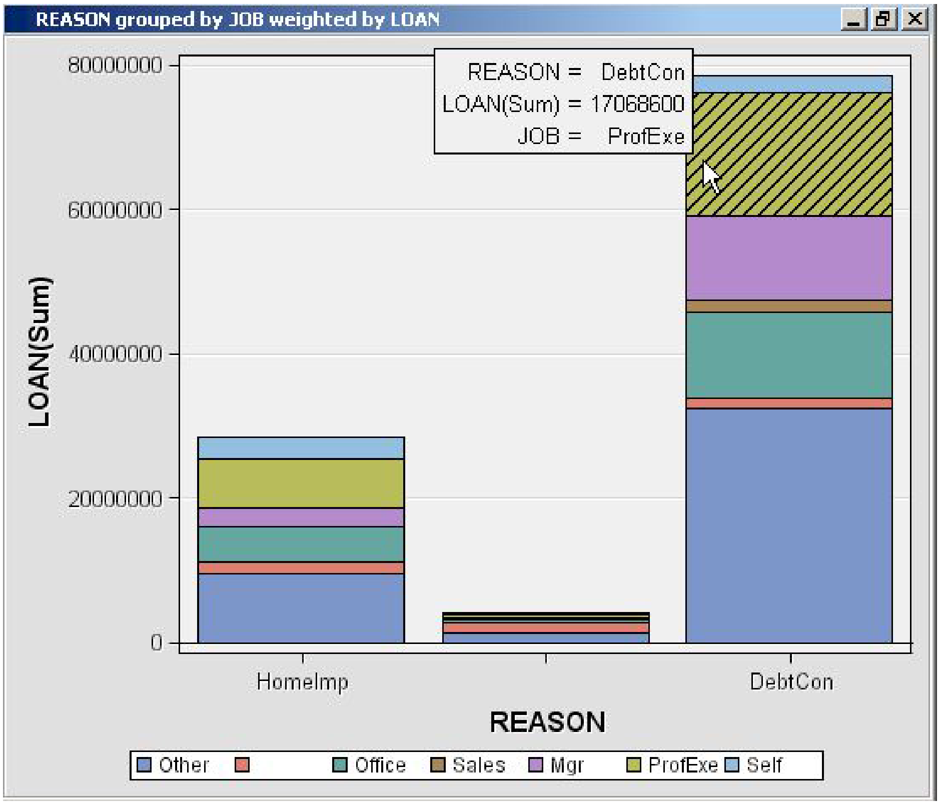
Save your modified code, click **Run Node** ( ), and then click **Results** ( ). You new graph should look like this:



There is a variable in our data set called LOAN that records the dollar amount of the requested loan. Suppose now that instead of displaying the number of loans by type, you want to display the dollar amounts, still grouping by JOB. To do this, add the FREQ argument to %EM_REPORT. Replace the call to %EM_REPORT with the following:

```
%em_report(
   key=example,
   viewtype=Bar,
   x=Reason,
   group=Job,
   freq=Loan,
   autodisplay=Y,
   description=%bquote(REASON grouped by JOB weighted by LOAN),
   block=%bquote(My Graphs));
```

Save your modified code, click **Run Node** ( ), and then click **Results** ( ). You new graph should look like this:



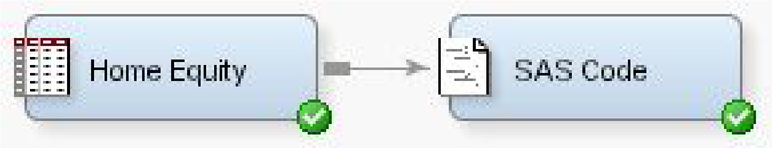
Multiple Bar Charts

The previous example, Bar Charts, demonstrated how to generate a single bar chart using %EM_REPORT. Specifically, when you used the Home Equity data set, a bar chart was generated for the variable REASON, grouped by JOB, and weighted by LOAN. This example extends that example by demonstrating how to generate a combo box that enables you to view different frames of a plot. The example starts where the previous example finished and adds two additional plots. A different weight variable is used for each frame of the plot. This is accomplished by including the VIEW argument of %EM_REPORT to specify an ID value in multiple calls to %EM_REPORT. The CHOICETEXT argument is also used. It enables you to attach text to each frame that is displayed by the combo box.

1. Create a new diagram

2. Add an input data source to the diagram. Use the Home Equity data set from the SAMPSIO library.

3. Add a **SAS Code** node to the diagram and connect it to the **Home Equity** node.



4. Click the **SAS Code** node and open the Code Editor.

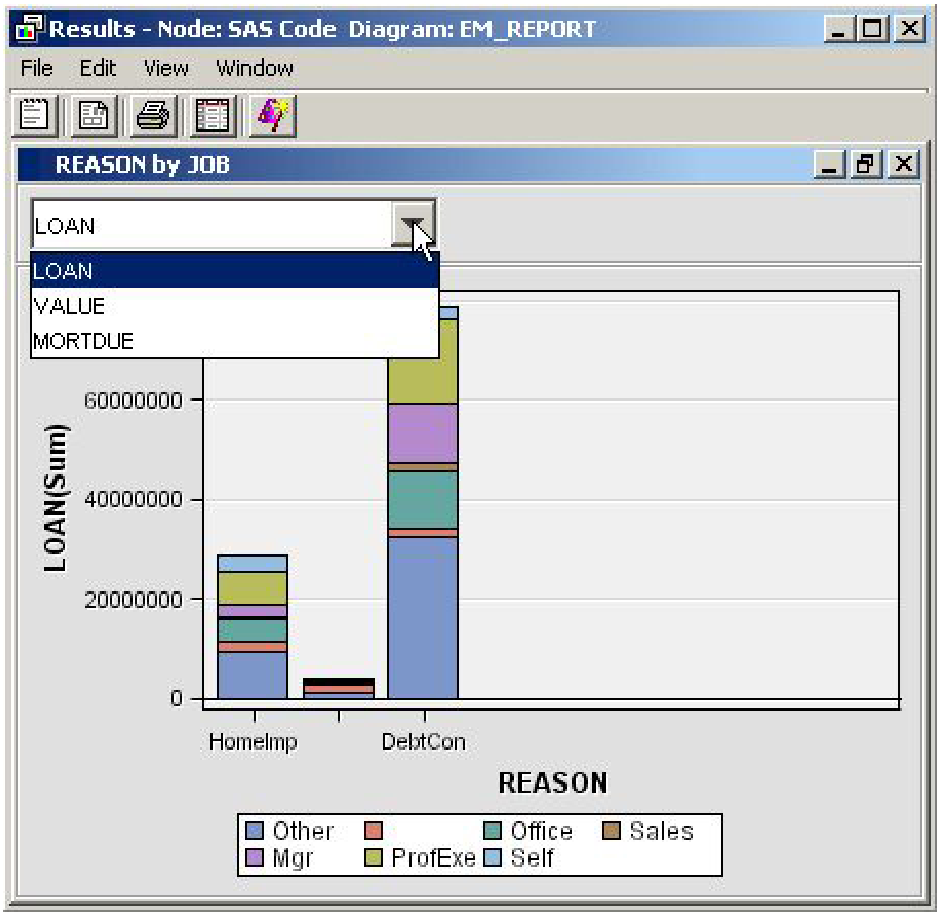
5. Enter the following code in the Report Code pane:

```
%em_register(type=Data, key=Example);
data &em_user_Example;
set &em_import_data;
run;

%em_report(
   key=Example,
   viewtype=Bar,
   view=1,
   x=Reason,
   group=Job,
   freq=Loan,
   choicetext=Loan,
   autodisplay=Y,
   description=%bquote(Reason by Job with Weights),
   block=%bquote(My Graphs));

%em_report(
   view=1,
   freq=Value,
   choicetext=Value);

%em_report(
   view=1,
   freq=Mortdue,
   choicetext=Mortdue);
```

   Each call to %EM_REPORT defines a different frame for the graph. There are three things that you should notice about the second and third calls to %EM_REPORT. The first is that you must specify the VIEW argument with the same ID number in all three calls to %EM_REPORT. This links the three calls. The second is that except for the VIEW argument, the only other arguments that you need to specify are the ones that have values that differ from the first call to %EM_REPORT. The third is that while arguments can have different values across the multiple calls to %EM_REPORT, you cannot specify different sets of arguments.

6. Click **Run Node** ( ).

7. Click **Results** ( ). When the Results window appears, double-click the title bar of the bar chart pane and you should see the following:



Click the drop-down arrow to choose a different frame to view.

There is no pre-defined limit on the number of frames that you can have. However, as the number of frames grows large, the utility of the combo box declines.

Multiple Y Plot

This example demonstrates how to use the macro %EM_REPORT to generate a line plot with two variables on the y-axis. The technique demonstrated previously, in the example “Multiple Bar Charts” on page 1157 , for generating multiple frames is also applied.

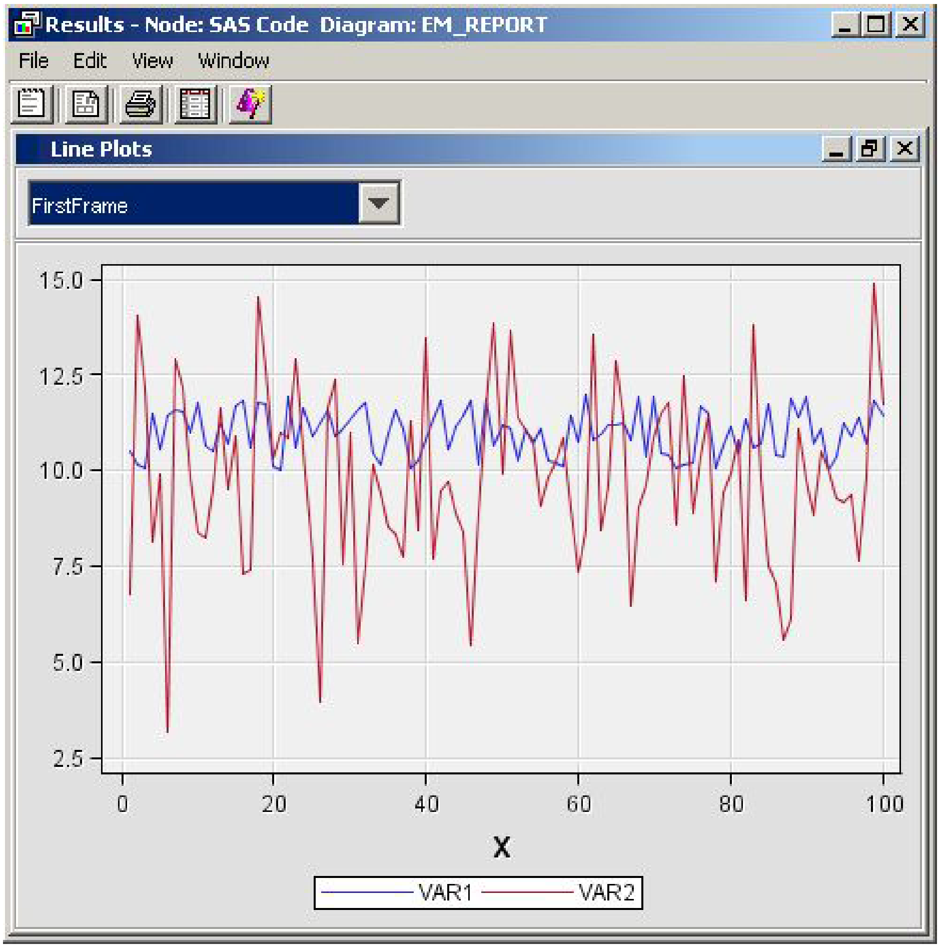
1. Create a new diagram.
2. Add a **SAS Code** node to the diagram. The data for the example is simulated.
3. Click the **SAS Code** node and open the Code Editor.
4. Enter the following code in the Report Code pane:

```
%em_register(type=Data, key=Sample);

/* Simulate the data */

data &em_user_Sample;
   do X=1 to 100;
      var1 = 10 + ranuni(1234)*2;
      var2 = 10 + rannor(1234)*2;
      var3 = 10 + rannor(1234)*2.5;
      output;
   end;
```

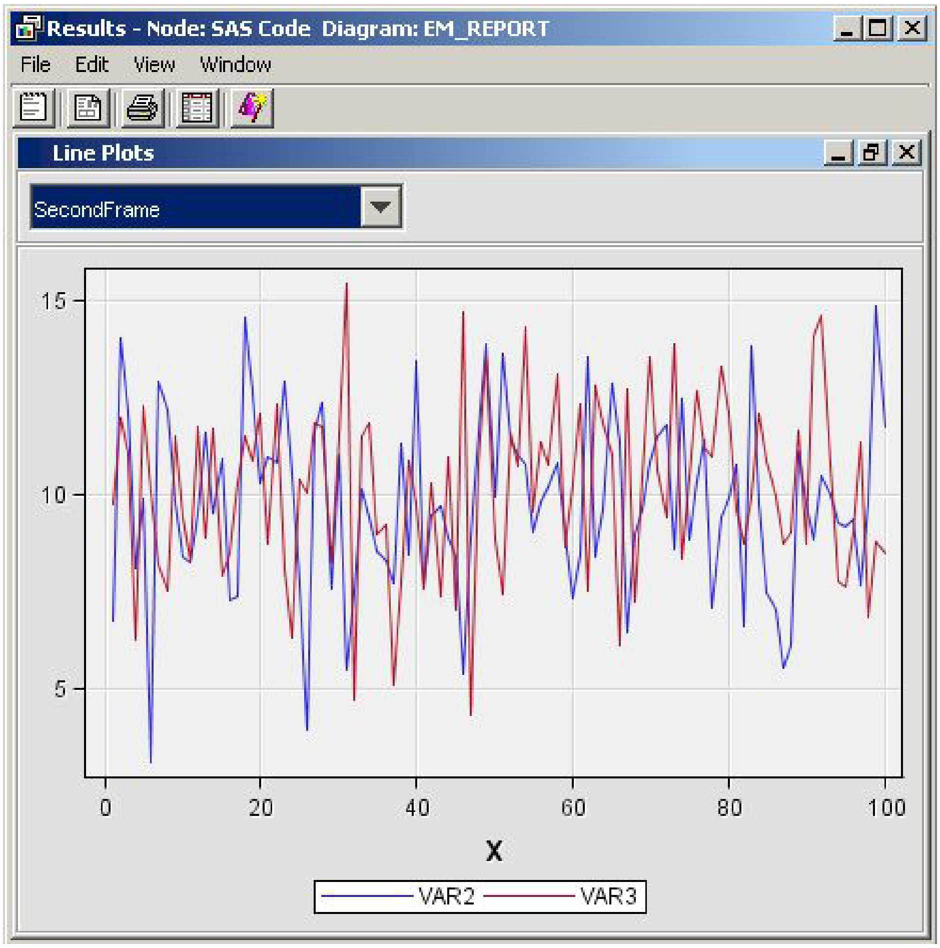
```
run ;

%em_report(
   key=Sample,
   viewtype=Lineplot,
   view=2,
   x=X,                    /* specify the x-axis variable     */
   y1=var1,                /* specify the 1st y-axis variable */
   y2=var2,                /* specify the 2nd y-axis variable */
   choicetext=FirstFrame,
   autodisplay=Y,
   description=%bquote(Line Plots),
   block=%bquote(My Graphs));

%em_report(
   view=2,
   y1=var2,
   y2=var3,
   choicetext=SecondFrame);
```

5. Click **Run Node** ( ).

6. Click **Results** ( ). When the Results window appears, double-click the title bar of the bar chart pane and you should see the following:



Click the drop-down arrow and select SecondFrame:

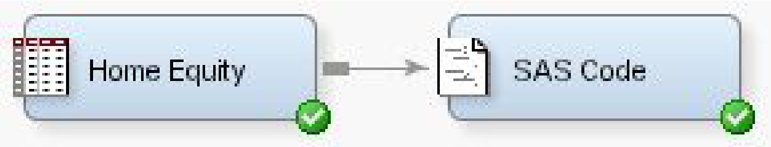


%EM_REPORT enables you to overlay up to 16 variables on the y-axis using the Y1=<variable name>, Y2=<variable name>, ... , Y16=<variable name> arguments.

Dendrogram

This example demonstrates how to use the macro %EM_REPORT to generate a dendrogram.

1. Create a new diagram.
2. Add an input data source to the diagram. Use the Home Equity data set from the SAMPSIO library.
3. Add a **SAS Code** node to the diagram and connect it to the **Home Equity** node.



4. Click the **SAS Code** node and open the Code Editor.
5. Enter the following code in the Report Code pane:

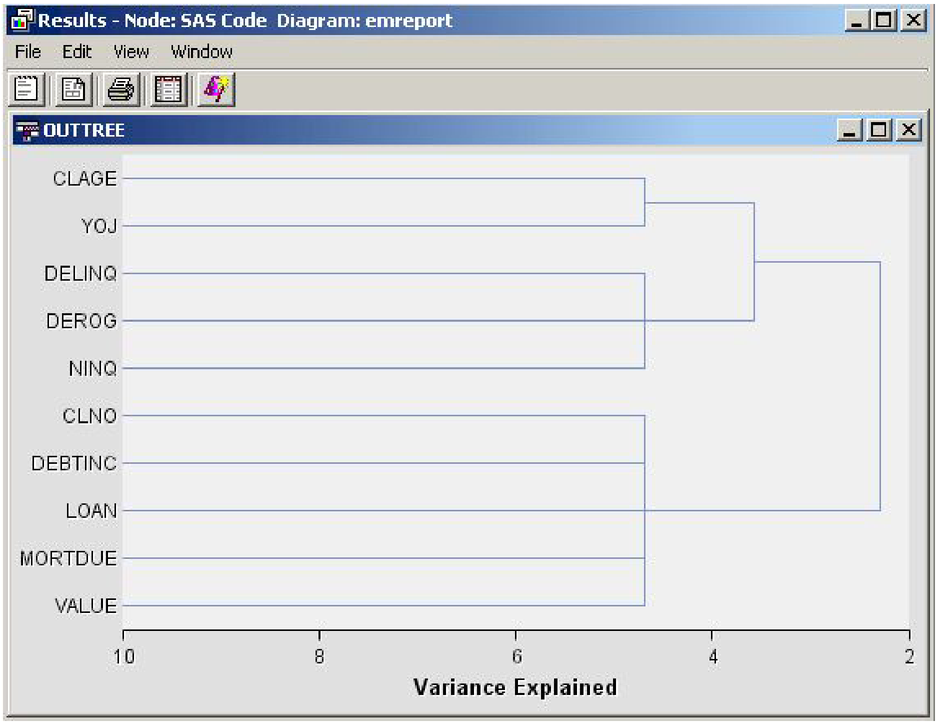
```
%em_register(key=Outtree, type=Data);
%em_getname(key=Outtree, type=Data);
proc varclus data = &em_import_data hi outtree=&em_user_Outtree;
```

```
var Clage Clno Debtinc Delinq Derog Loan Mortdue Ninq Value Yoj;
run;
%em_report(
   key=OUTTREE,
   viewtype=DENDROGRAM,
   autodisplay=Y,
   block=Dendrogram,
   name=_Name_,
   parent=_Parent_,
   height=_Varexp_);
```

Note: The macro %EM_GETNAME used in the example code above returns a filename an initializes the macro variable &EM_USER_KEY, where KEY is the data key that is defined in the call to %EM_REGISTER.

6. Click **Run Node** ( ).

7. Click **Results** ( ). When the Results window appears, close the output pane and double-click the title bar of the OUTTREE pane and you should see the following:



If you click **View** in the Results window, you will see the item Dendrogram. It will have a submenu item, OUTTREE.

Three Dimensional Components

This example demonstrates how to use %EM_REPORT to generate 3-dimensional scatter, bar, and surface plots.

1. Create a new diagram

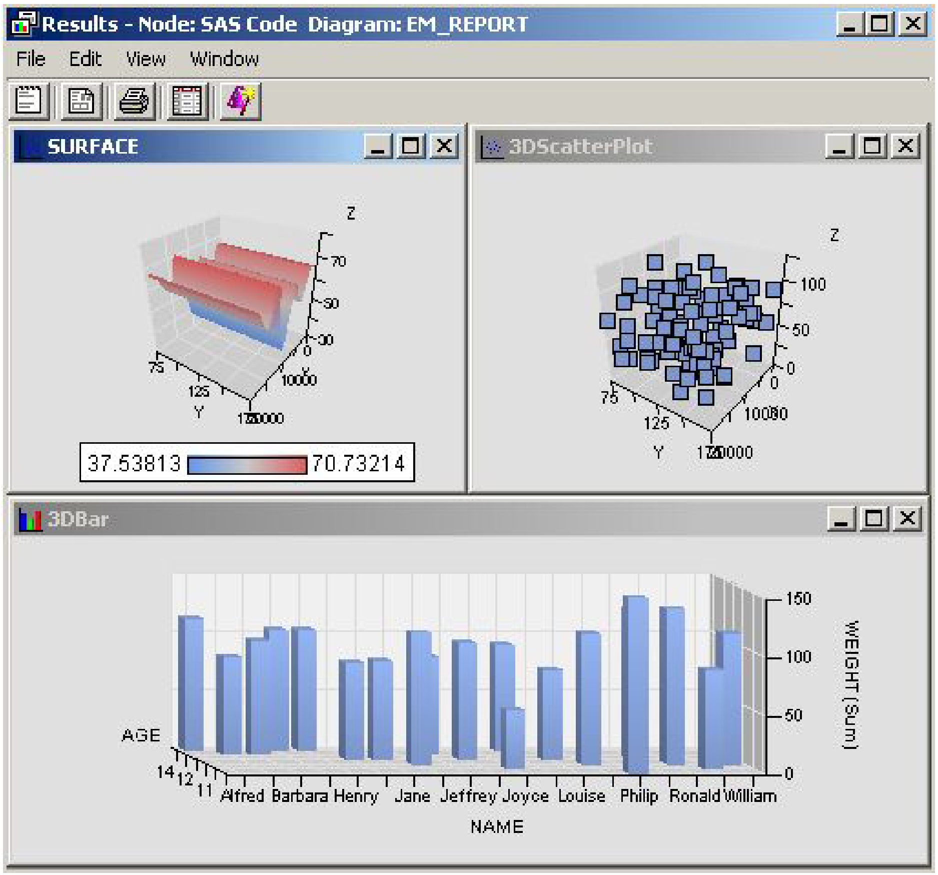
2. Add a **SAS Code** node to the diagram. The example uses simulated data and data that is available from the SASHELP library that is automatically included with your SAS installation.

3. Click the **SAS Code** node and open the Code Editor.
4. Enter the following code in the Report Code pane:

```
%em_register(key=Data, type=Data);

/* simulate data */

data One;
do i = 1 to 100;
   x= ranuni(0) * 100 * 200;
   y = ranuni(0) * 100 + 75;
   z = ranuni(0) * 100 + 10;
   output;
end;
run;

data &em_user_data;
  set Work.One;
run;

/* K-Dimensional Scatter Plot */

%em_report(
   key=Data,
   viewtype=ThreeDScatter,
   x=X,
   y=Y,
   z=Z,
   block=%bquote(My Graphs),
   description=%bquote(3DScatterPlot),
   autodisplay=Y);

/* K-Dimensional Surface Plot */

%em_report(
   key=Data,
   viewtype=Surface,
   x=X,
   y=Y,
   z=Z,
   block=%bquote(My Graphs),
   description=%bquote(Surface),
   autodisplay=Y);

%em_register(key=Class, type=Data);

data &em_user_Class;
  set Sashelp.Class;
run;

/* K-Dimensional Bar Chart */

%em_report(
   key=Class,
   viewtype=ThreeDBar,
   x=Name,
   y=Weight,
   series=Age,
   block=%bquote(My Graphs),
   description=%bquote(3DBar),
   autodisplay=Y);
```

5. Click **Run Node** ( ).

6. Click **Results** ( ).



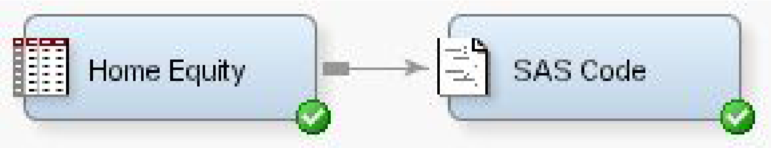
Simple Lattice of Plots

This example demonstrates how to use %EM_REPORT to generate a simple lattice of plots. A lattice of plots is a collection of plots displayed as a grid.

1. Create a new diagram.

2. Add an input data source to the diagram. Use the Home Equity data set from the SAMPSIO library.

3. Add a **SAS Code** node to the diagram and connect it to the **Home Equity** node.



4. In the properties panel of the **Home Equity** data source node, click the ... icon for the Variables property to open the variables table. Change the Role property of the variables JOB and REASON to Classification and click **OK**.

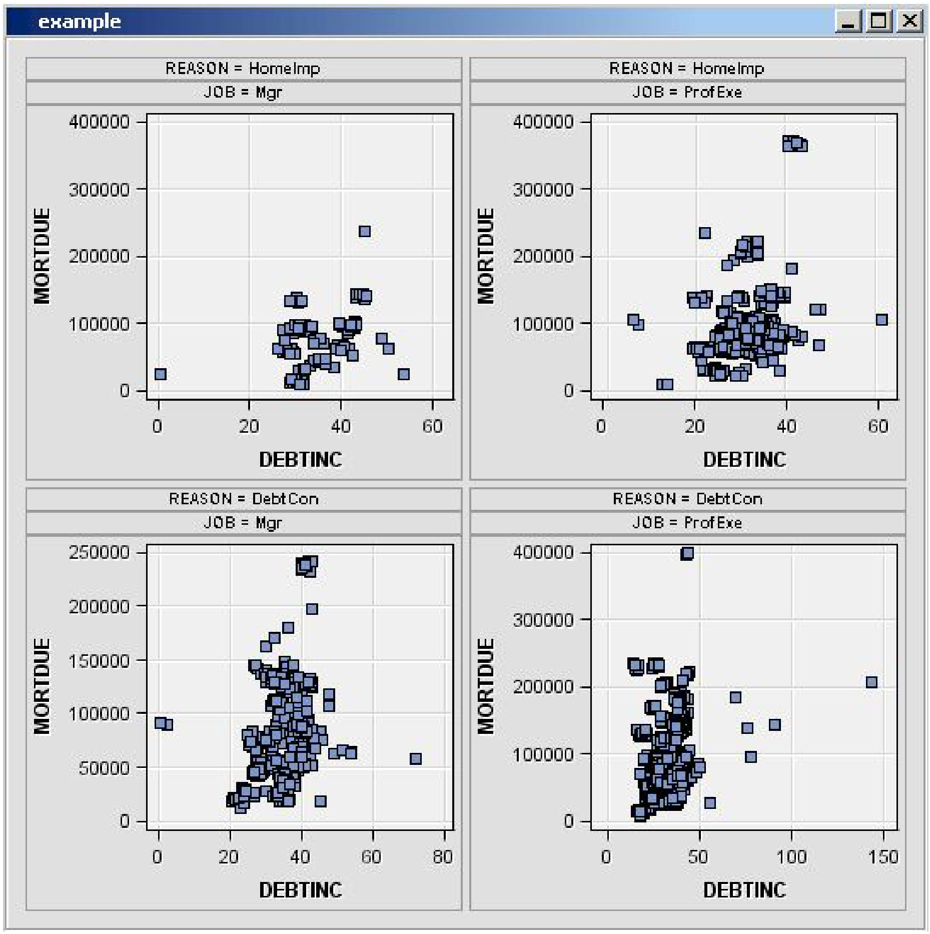
5. Click the **SAS Code** node and open the Code Editor.

6. Enter the following code in the Report Code pane:

```
%em_register(type=Data, key=Example);
data &em_user_Example;
   set &em_import_data;
   where (Job='ProfExe' or Job='Mgr') and
         (Reason = 'DebtCon' or Reason = 'HomeImp');
run;
%em_report(
   key=Example,
   viewtype=Lattice,
   latticetype=Scatter,
   x=Debtinc,
   y=Mortdue,
   latticex=Job,
   latticey=Reason);
```

7. Click **Run Node** ( ).

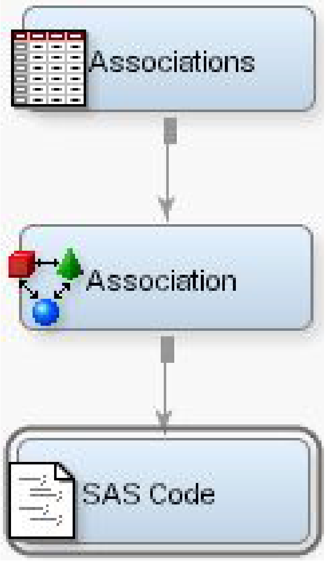
8. Click **Results** ( ). When the Results window appears, select **View** ⇨ **Custom Reports** ⇨ **example**.



Constellation Plot

This example demonstrates how to use %EM_REPORT to generate a Constellation plot.

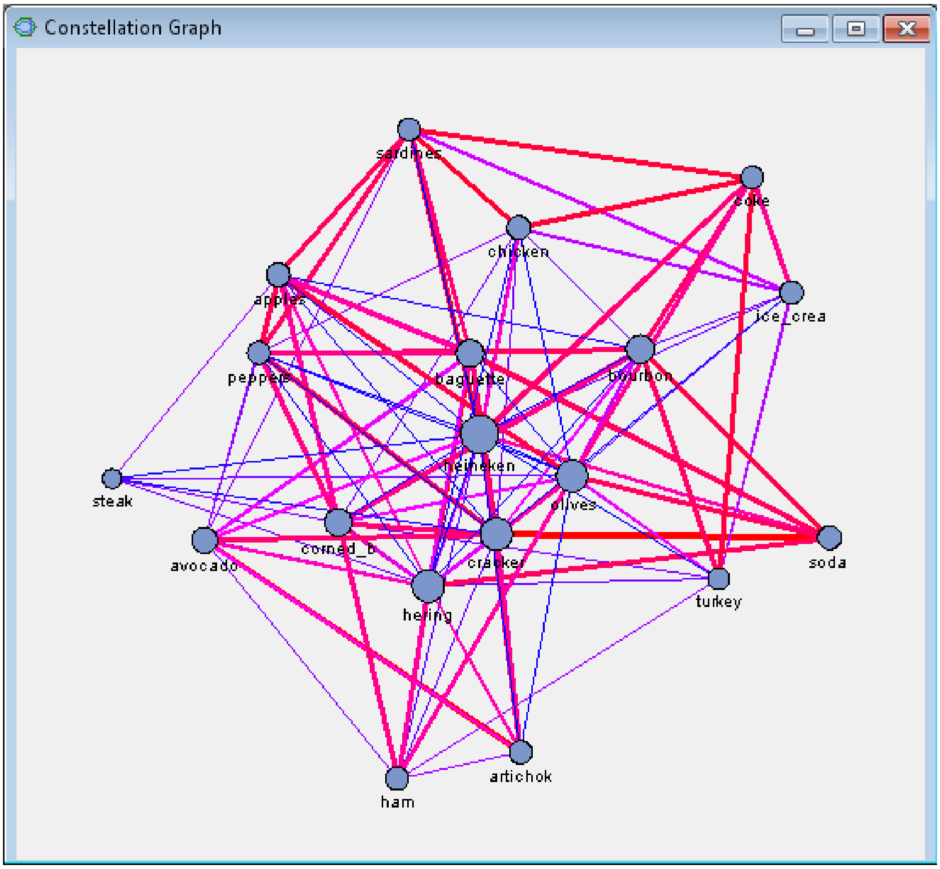
1. Create a new diagram.
2. Add an input data source to the diagram. Use the Associations data set from the SAMPSIO library.
3. Add an **Association** node to the diagram and connect it to the data source node.
4. Add a **SAS Code** node to the diagram and connect it to the **Association** node.



5. Click the **SAS Code** node and open the Code Editor.
6. Enter the following code in the **Report Code** pane:

```
%em_register(key=A, type=DATA);
%em_register(key=B, type=DATA);

data &em_user_a;
   set &em_lib..assoc_links;
run;

data &em_user_b;
   set &em_lib..assoc_nodes;
run;

%em_report(viewtype=Constellation,
           linkkey=A,
           nodekey=B,
           LINKFROM=FROM,
           LINKTO=TO,
           LINKID=linkid,
           LINKVALUE=CONF,
           nodeid=item,
           nodesize=count,
           nodetip=item);
```

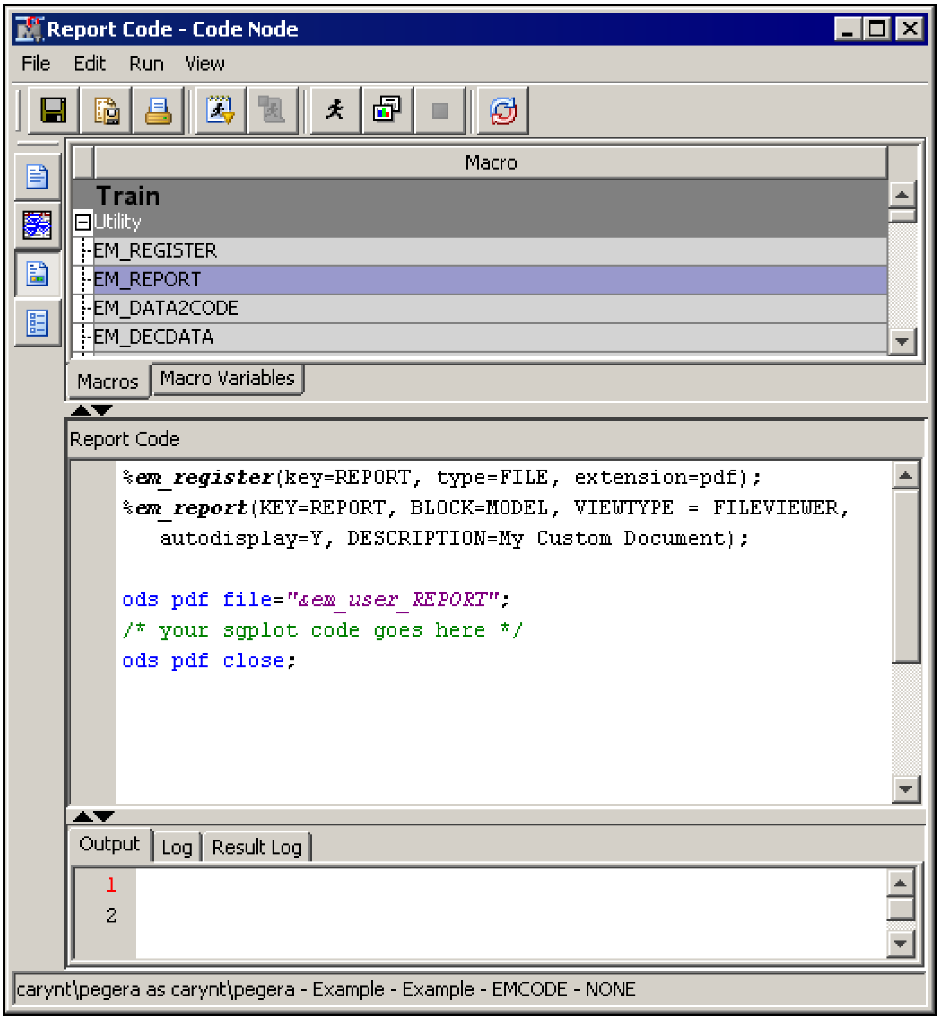
7. Click **Run Node** ( ).
8. Click **Results** ( ). When the Results window appears, select **View** ⇨ **Custom Reports** ⇨ **Constellation Graph**



Importing Statistical Graphics with %EM_REPORT

The **SAS Code** node can be used to display the results of PROC SGPLOT, or any other SG procedure. What follows here is a pseudo-example that demonstrates the structure required to import these graphics. The graphics are first exported as a PDF by PROC SGPLOT and then imported using %EM_REPORT. To begin, you need to enter the following code in the **Report Code** section of the Code Editor. This is also illustrated in the image below.

```
%em_register(key=REPORT, type=FILE, extension=pdf);
%em_report(KEY=REPORT, BLOCK=MODEL, VIEWTYPE = FILEVIEWER,
    autodisplay=Y, DESCRIPTION=My Custom Document);

ods pdf file="&em_user_REMPORT";
/* your sgplot code goes here */
ods pdf close;
```



```
%em_register(key=REPORT, type=FILE, extension=pdf);
%em_report(KEY=REPORT, BLOCK=MODEL, VIEWTYPE = FILEVIEWER,
   autodisplay=Y, DESCRIPTION=My Custom Document);

ods pdf file="&em_user_REPORT";
/* your sgplot code goes here */
ods pdf close;
```

caryntpegera as caryntpegera - Example - Example - EMCODE - NONE

Notice the commented portion `/* your sgplot code goes here */`. You need to insert your specific PROC SGPLOT code at that point in the code.

This code creates a window in the **SAS Code** node Results window that enables you to open a PDF file that contains the results of the SGPLOT procedure. The %EM_REGISTER macro creates a file reference for the ODS file that is produced and is stored in the project's workspace for this code node. For example, this could be `C:\Project\test\Workspaces\EMWS7\EMCODE\report.pdf`, which maps to the project, workspace, code node ID, and file reference for the PDF.

The %EM_REPORT macro uses that file reference to populate a window in the **SAS Code** Node results browser that displays a link to your report. The `ODS PDF` statement uses a macro variable created by the %EM_REGISTER macro called &EM_USER_REPORT to write the PDF file in the desired location. Here, the term `report` in this macro variable is the same as the KEY= value in the &EM_REGISTER macro invocation.

Chapter 76
Score Code Export Node

Score Code Export Node

Overview of the Score Code Export Node

The Score Code Export node is located on the Utilities tab of the Enterprise Miner workspace.

SAS Enterprise Miner creates SAS language score code for the purpose of scoring new data. Users run this code in production systems to make business decisions for each record of new data.

The Score Code Export node is an extension for SAS Enterprise Miner that exports files that are necessary for score code deployment. Extensions are programmable add-ins for the SAS Enterprise Miner environment.

The following files are exported by the Score Code Export node:

- the SAS scoring model program.
- an XML file containing scoring variables and other properties.
- an XML file containing descriptions of the final variables created by the scoring code. This file can be kept for decision-making processes.
- a ten-row sample of the scored data set showing typical cases of the input attributes, intermediate variables, and final output variables. This data set can be used to test and debug new scoring processes.
- a ten-row sample of the training data set showing the typical cases of the input attributes.

- a format catalog, if the scoring program contains user-defined formats.

For more information about the exported files, see “Score Code Node Output” on page 1175 .

SAS Enterprise Miner can register models directly in the SAS Metadata Server. Models registered in the SAS Metadata Server are used by SAS Data Integration Studio, SAS Enterprise Guide, and SAS Model Manager for creating, managing, and monitoring production and analytical scoring processes.

The Score Code Export node exports score code created by SAS Enterprise Miner into a format that can be used by the SAS Scoring Accelerator for Teradata. The exported files are stored in a directory, not the SAS Metadata Server.

The Score Code Export node does not replace the functionality of registering models in the SAS Metadata Server to be used by SAS Data Integration Studio, SAS Enterprise Guide, and SAS Model Manager.

Data Requirements of the Score Code Export Node

The Score Code Export node requires a process flow diagram that contains both a Score node and another node that creates score code. The Score node aggregates the score code for the entire process flow diagram and transfers it the Score Code Export node. Therefore, the Score node must directly precede the Score Code Export node.

Properties of the Score Code Export Node

Score Code Node General Properties

The following general properties are associated with the Score Code Export node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Score Code Export node that is added to a diagram will have a Node ID of CodeXpt. The second Score Code Export node added to a diagram will have a Node ID of CodeXpt2, and so on.

- **Imported Data** — accesses the Imported Data — Regression window. The Imported Data — Regression window contains a list of the ports that provide data sources to the Regression node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.

  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Exported Data** — accesses the Exported Data — Regression window. The Exported Data — Regression window contains a list of the output data ports that the Regression node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Score Code Node Train Properties

- **Rerun** — Use this property to force the node to run again. This property is useful if the macro variable controlling the target directory and folder name have changed.
- **Output Directory** — Enter a fully qualified name for the location of an output directory to contain the score code files. If no directory is entered, a default directory named Score is created in the SAS Enterprise Miner project directory. You can change the value of the default directory by setting the &EM_SCOREDIR=*directory* macro variable in the SAS Enterprise Miner project start code or server start code.
- **Folder Name** — Enter the name of the model that you are creating. The name is used to create a new subdirectory in the output directory that contains the exported score files. If no name is entered, a default name is generated as a combination of the &SYSUSERID automatic macro variable and an incremental index (for example, userID, userID_2, userID_3).

  You can replace the &SYSUSERID automatic macro variable with a custom name by setting the &EM_SCOREFOLDER=*score-folder-name* macro variable in the SAS Enterprise Miner project start code or server start code. An incremental index preceded by an underscore is added to *score-folder-name*.

Score Code Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Score Code Export Node Results

You can open the Results window of the Scorecard node by right-clicking the node and selecting **Results**. For general information about the Results window, see Using the Results Window.

Select **View** from the main menu to view the following results in the Scorecard Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Score Code Export node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Score Code Export node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Score Code Export node run.
  - **Output** — the SAS output of the Score Code Export node run. The Scorecard SAS output includes the following:
    - variables summary
    - output variables
    - files exported
  - **Flow Code** — the SAS code used to produce the output that the Score Code Export node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Predictive Model Markup Language (PMML) code that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled.
  - **Input Variables** — a read-only table of the input variables for the Score Code Export node.
  - **EM Output Variables** — a read-only table of the output variables for the Score Code Export node. This table indicates the role, create, type, label, and length of each output variable.
  - **Summary** — A table that indicates who ran the node, when it was run, and where the output files are located.
- **Table** — is unavailable for this node.
- **Plot** — opens the Graph Wizard for the table that you select.

Score Code Node Output

Score Code Export Node Output Files

The Score Code Export node writes the following output files, and possibly a formats catalog, to the location specified by the Output Directory property. These files are used as input to the %INDTD_PUBLISH_MODEL macro that creates the Teradata scoring functions.

- score.sas — SAS language score code that is created by Enterprise Miner. This code ban be used directly in a SAS program.
- score.xml — A description of the variables that are used and created by the scoring code. XML files are created by a machine process for the use of machine process. Do not edit the XML file.

  Note: The maximum number of input variables for a UDF function is 128.
- emoutput.xml — A description of the final variables that are created by the scoring code. This file can be kept for decision-making processes. These variables include the primary classification, prediction, probability, segment, profit, and loss variables that are created by a data mining process. The list does not include intermediate variables that are created by the analysis. For more information, see Fixed Variable Names.
- scoredata.sas7bdat — A ten-row sample of the score data set that shows typical cases of the input attributes, intermediate variables, and final output variables. Use this data set to test and debug new scoring processes.
- traindata.sas7bdat — A ten-row sample table of the training data set that shows typical cases of the input attributes, intermediate variables, and final output variables.
- Formats Catalog — If the training data contains SAS user-defined formats, the Score Code Export node creates a formats catalog. The catalog contains the user-defined formats in the form of a look-up table. This file has an extension of .sas7bcat.

Score Code Export Node Output Variables

The Score Code Export node creates score code with both intermediate variables and output variables. Intermediate variables include imputed values of missing variables, transformation variables, and encoding variables. Output variables include predicted values and probabilities. Any of the intermediate or output variables can be used in a scoring process.

The number of input parameters on a scoring function has a direct impact on performance. The more parameters there are, the more time it takes to score a row. A recommended best practice is to ensure that only variables that are involved in a model score evaluation are exported by the Score Code Export node.

The most important output variables use a prefix and follow the naming conventions outlined in the following table:

Role	Type	Prefix	Key	Suffix	Example
Prediction	N	P_	Target Variable Name		P_amount

Role	Type	Prefix	Key	Suffix	Example
Probability	N	P_	Target Variable Name	Predicted event value	P_ purchaseYES P_ purchaseNO
Classification	$	I_	Target Variable Name		I_purchase
Expected Profit	N	EP_	Target Variable Name		EP_ conversion
Expected Loss	N	EL_	Target Variable Name		EL_ conversion
Return on Investment	N	ROI_	Target Variable Name		ROI_ conversion
Decision	$	D_	Target Variable Name		D_ conversion
Decision Tree Leaf	N	_NODE_			_NODE_
Cluster Number or SOM cell ID	N	_SEGMENT_			_SEGMENT_

Fixed Variable Names

The Score node maps the output variable names to fixed variable names. This mapping is appropriate in the most common case that there is only one prediction target or one classification target. The table below will help make sense of other cases.

Using the fixed variable names enables scoring users to build processes that can be reused for different models without changing the code that processes the outputs. These fixed names are listed in the **emoutput.xml** file and are described in the table below. Most scoring processes return one or more of these variables.

Role	Type	Fixed Name	Description
Prediction	N	EM_PREDICTION	The prediction value for an interval target variable.
Probability	N	EM_PROBABILITY	The probability of the predicted classification, which can be any one of the target variable values.

Role	Type	Fixed Name	Description
Probability	N	EM_ EVENTPROBABILITY	The probability of the target event. By default, this is the first value in descending order, but you can alter the ordering scheme. This is often the event of interest.
Classification	$	EM_ CLASSIFICATION	The predicted target class value.
Expected Profit	N	EM_PROFIT	Based on the selected decision.
Expected Loss	N	EM_LOSS	Based on the selected decision.
Return on Investment	N	EM_ROI	Based on the selected decision.
Decision	$	EM_DECISION	Optimal decision based on a function of probability, cost, and profit or loss weights.
Decision Tree Leaf, Cluster number, or SOM cell ID	N	EM_SEGMENT	Analytical customer segmentation.

SAS Enterprise Miner Tools Production of Score Code

The table below shows the types of score code that is created by each node in SAS Enterprise Miner. In addition, users can develop extension nodes, which can create either SAS DATA step or SAS program score code. However, this code is not converted to PMML, C, or Java.

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata UDF
Sample						
Input Data	*	*	*	*	*	*
Sample	*	*	*	*	*	*
Partition	*	*	*	*	*	*
Append	N	Y	N	N	N	N

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata UDF
Merge	N	Y	N	N	N	N
Time Series	N	Y	N	N	N	N
Filter	Y When the user keeps the created filter variable.	*	N	Y	Y	Y
Explore						
Association	N	Y	Y	N	N	N
Cluster	Y	N	Y	Y	Y	Y
DMDB	*	*	*	*	*	*
Graph Explore	*	*	*	*	*	*
Market Basket	N	N	N	N	N	N
Multiplot	*	*	*	*	*	*
Path	N	Y	Y	N	N	N
SOM	Y	N	N	Y	Y	Y
Stat Explore	*	*	*	*	*	*
Text Miner	N	N	N	N	N	N
Variable Clustering	Y	N	N	Y	Y	Y
Variable Selection	Y	N	N	Y	Y	Y
Modify						
Drop	*	*	*	*	*	*
Impute	Y	N	Y	Y	Y	Y
Interactive Binning	Y	N	N	Y	Y	Y

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata UDF
Replacement	Y	N	N	Y	Y	Y
Principle Components	Y	N	N	Y	Y	Y
Rules Builder	Y	N	N	Y	Y	Y
Transform Variables	Y	N	N	Y	Y	Y
Model						
Autoneural	Y	N	Y	Y	Y	Y
Decision Tree	Y	N	Y	Y	Y	Y
Dmine Regression	Y	N	Y	Y	Y	Y
Dmine Neural	Y	N	N	Y	Y	Y
Ensemble	Y	N	N	Y	Y	Y
Gradient Boosting	Y	N	N	Y	Y	Y
MBR	N	Y	N	N	N	N
Model Import	*	*	*	*	*	*
Neural Network	Y	N	Y	Y	Y	Y
Partial Least Squares	Y	N	N	Y	Y	Y
Regression (linear, logistic, and multinomial)	Y	N	Y	Y	Y	Y
Rule Induction	Y	N	N	Y	Y	Y

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata UDF
SVM — Linear Kernel	Y	N	Y	Y	Y	Y
SVM — Nonlinear Kernel	N	Y	N	N	N	N
Two Stage	Y	N	N	Y	Y	Y
Assess						
Cutoff	Y	N	N	Y	Y	Y
Decisions	Y	N	N	Y	Y	Y
Model Compariso n	Y	N	N	Y	Y	Y
Score	Y	N	N	Y	Y	Y
Sgement Profile	*	*	*	*	*	*
Utility						
Control Point	*	*	*	*	*	*
Start Groups	Y	N	N	Y	Y	Y
End Groups	Y	N	N	Y	Y	Y
Metadata	*	*	*	*	*	*
Reporter	*	*	*	*	*	*
SAS Code The user can enter either SAS DATA step code or SAS program code.	Y	Y	N	N	N	N
Credit Scoring						
Credit Exchange	*	*	*	*	*	*

Node	SAS DATA Step	SAS Program	PMML	C	Java	Teradata UDF
Interactive Grouping	Y	N	N	Y	Y	Y
Score Card	Y	N	N	Y	Y	Y
Reject Inference	Y	N	N	Y	Y	Y
* The node does not produce this type of score code.						

Chapter 77

Start Groups Node

Start Groups Node

Overview of the Start Groups Node

The **Start Groups** node is located on the **Utility** tab of the SAS Enterprise Miner tools bar. The **Start Groups** node is a descendant of the SAS Enterprise Miner 4.3 **Group Processing** node. The **Start Groups** node is useful when your data can be segmented or grouped, and you want to process the grouped data in different ways. The **Start Groups** node uses BY-group processing as a method to process observations from one or more data sources that are grouped or ordered by values of one or more common variables. BY variables identify the variable or variables by which the data source is indexed, and BY statements process data and order output according to the BY group values.

You can use the SAS Enterprise Miner **Start Groups** node to do the following tasks:

- define group variables such as GENDER or JOB, in order to obtain separate analyses for each level of group variable
- analyze more than one target variable in the same process flow
- specify index looping, or how many times the flow that follows the node should loop
- resample the data set and use unweighted sampling to create bagging models
- resample the training data set and use reweighted sampling to create boosting models

Start Groups Node and Missing Data Set Values

The **Start Groups** node processes observations that have missing values as valid group values. If the input data set that you want to perform group processing on contains a significant number of observations with missing values, it might be beneficial to use the Replacement or Impute nodes to replace or impute missing variable values before submitting the data set to the **Start Groups** node.

Using the Start Groups Node

The **Start Groups** node must be used in a process flow diagram that has one or more data sources. If you import more than one data set, the data sets must be compatible with each other. For example, if you import a training data set into the **Start Groups** node that uses a group variable named REGION, then the group variable REGION must also be in your score data set. When you import more than one data set (for example, a training, a validation, and a score data set), then group processing is performed on each data set when you run the process flow.

The **Start Groups** node requires at least one target variable to run. The **Start Groups** node can process more than one target variable. The **Start Groups** node processes all variables that have a role of target as targets.

In a process flow diagram, the group processing portion of the process flow diagram is defined by the **Start Groups** node, the **End Groups** node, and the data mining node tools that you place between the **Start Groups** node and the **End Groups** node. The **Start Groups** node is followed by one or more successor nodes that perform some data mining operation. The **End Groups** node defines the scope of the group processing operations. The **Start Groups** node will not run without an accompanying **End Groups** node. The **End Groups** node also accumulates data mining results for group processing reporting.

You cannot perform group processing on Transaction data sets in SAS Enterprise Miner.

You can connect any node to the **Start Groups** node as a successor, but only modeling nodes and the **Score** node accumulate results during individual loop iterations. If you want to accumulate the results of each loop iteration, you can use a successor **SAS Code** node to capture and record the intermediate values.

It is possible to have more than one **Start Groups** and **End Groups** node pair in a process flow, as long as each group processing operation (within a **Start Groups** and **End Groups** node pair) is performed serially. In other words, each group processing operation must complete before the subsequent group processing operation begins. SAS Enterprise Miner cannot run more than one group processing operation at a time. As a result, you cannot nest a group processing operation within a group processing operation. The restriction also means that you also should not design process flow diagrams that perform group processing operations in competing parallel branches.

Start Groups Node and the Ensemble Node

When using the **Start Groups** node to perform bagging or boosting in SAS Enterprise Miner 4.3, it was necessary to place an **Ensemble** node at the end of the group processing portion of the process flow diagram in order to produce the final bagging or boosting model or models.

In SAS Enterprise Miner 12.3, the **Start Groups** node does not require an **Ensemble** node to perform bagging or boosting. The **End Groups** node that terminates the group processing portion of all SAS Enterprise Miner 12.3 group processing diagrams also contains the code that is required to produce the final bagging or boosting model or models.

Start Groups Node Looping

The **Start Groups** node processes data using looping modes that cause the group processing portion of the process flow diagram to repeat a number of times. The looping modes for the **Start Groups** node are as follows:

- **Index** — the index mode setting specifies the number of times to loop through the group processing portion of the process flow diagram. No additional sampling or model averaging is performed.
- **Bagging** — the bootstrap aggregation, or bagging mode creates unweighted samples of the active training data set for bagging. The bagging method uses random sampling with replacement to create the samples. Unweighted resampling does not apply extra weight to the cases that were misclassified in the previous samples. When you specify the Bagging mode, you also need to use the Bagging section of the Train properties panel to specify your settings for the Type, Observations, Percentage, and Random Seed properties.

  Unweighted resampling for bagging is the most straightforward method of resampling. The Bagging method uses random sampling with replacement to create the n sample replicates. You set the number of samples (and, in turn, models) that you want to create in the Index Count property in the General section of the Train properties.

  You can specify the sample size as a percentage of the total observations in the data set, or as an absolute number of observations to be sampled. The default percentage is set to 100% of the observations in the input data set. The default number of observations is set to the total number of observations in the input data set. The actual sample size is an approximate percentage or number. For example, a 10% random sample of 100 observations might contain 9, 10, or 11 observations.

  If the data set contains a frequency variable, then the frequency variable is used to determine the percentage of observations sampled instead of the number of observations. For example, assume that you have the following data set:

Obs	Freq	X1
1	5	a
2	3	b
3	2	c

  In this case, the percentage of observations sampled is based on the total frequency count (5+3+2 =10) instead of the total number of observations in the data set (3).

  By default, the random seed that is used to generate the initial sample is 12345. To create a new random seed, enter a new seed value in the property field. The seed values used to generate your samples are saved by the node, so you can replicate the samples in another run of the process flow. You must use the same initial seed value to replicate samples from one run to another. To automatically use a different seed value each time, set the seed to 0. When the seed value is set to 0, the computer clock creates a random seed at run time to initialize the seed stream. Using this method, your samples are always different when you rerun the flow.

In bagging mode, after the group processing portion of the process flow diagram repeats the specified number of times, the final model is created by averaging the probabilities that were generated in each model iteration. Each iteration creates a new sample of the data that introduces variability. Bagging is most often performed with decision tree models.

- **Boosting** — Reweighted resampling was developed as a way of boosting the performance of a weak learning algorithm. Use the boosting mode if you want the sampling method to reweight each training observation. The weights in the resampling are increased for the observations that are most often misclassified in the previous models. Therefore, the distribution of the observation weights is based on the model performance of the previous samples.
- **Target** — the target mode loops for each target variable that is identified in the training data. The target looping mode creates similar models for a number of targets, such as modeling a competitive cross-sell scenario.
- **Stratify** — Use the Stratify mode to perform standard group processing. When you use the Stratify mode, the **Start Groups** node loops through each level of group variable when you run the process flow diagram. When you select Stratify, the Minimum Group Size and Target Group properties are enabled.
- **Cross-Validation** — the cross validation looping mode is a modification of the stratified mode. You use the cross validation mode when you want to export the complement of the groups specified, as opposed to the groups themselves. For example, assume you have two group variables, GENDER [M, F] and REGION [N, S, E, W]. Using Stratify-mode looping to perform standard group processing, the groups passed would be as follows:

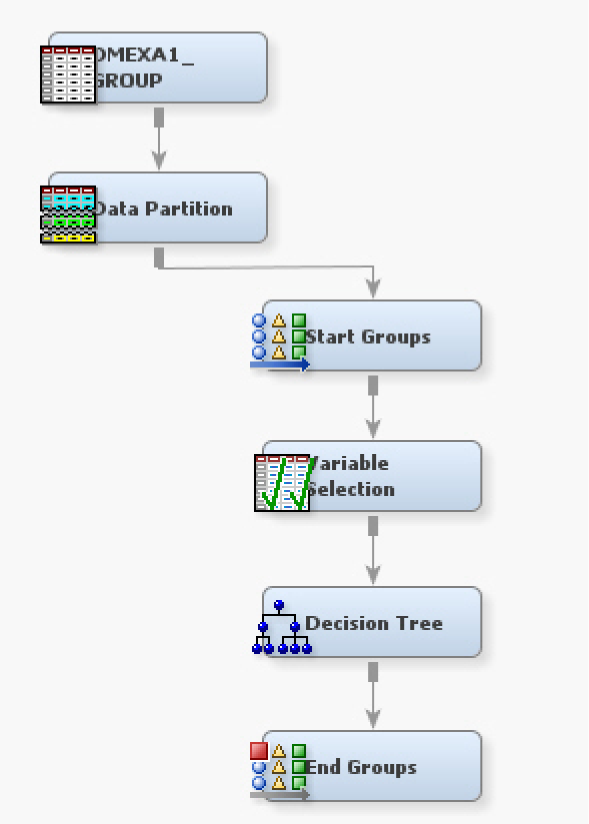
  - Loop 1 — Gender M and Region N
  - Loop 2 — Gender F and Region N
  - Loop 3 — Gender M and Region S
  - . . .
  - Loop 8 — Gender F and Region W

  When using Cross-Validation mode looping, all the data except the group is passed on. Using Cross-Validation mode looping with our example group variables GENDER [M, F] and REGION [N, S, E, W] we get the following:

  - Loop 1 — not(Gender M and Region N), that is, everything that is not M and N
  - Loop 2 — not(Gender F and Region N), that is, everything that is not F and N
  - Loop 3 — not(Gender M and Region S), that is, everything that is not M and S
  - . . .
  - Loop 8 — not(Gender F and Region W), that is, everything that is not F and W
- **No Grouping** — no looping is performed. You might want to suppress the **Start Groups** node from looping to reduce the run time during preliminary model building.

Start Groups Node and Bag Bite Sampling

You can use the index looping mode of the **Start Groups** node to perform Bag Bite sampling. To perform Bag Bite sampling, you typically write customized SAS code to create the n samples of the input data set that are required for subsequent modeling.



Each time the flow loops, the **Start Groups** node passes the input data to the **Variable Selection** node. The data then passes to the **Decision Tree** node for modeling. The **End Groups** node signifies the end of the Group Processing portion of the process flow diagram, and calculates the posterior probabilities (for class targets) or the predicted values (for interval targets) from the individual tree models to form the final classifier.

Start Groups Node Properties

Start Groups Node General Properties

The following general properties are associated with the **Start Groups** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Start Groups** node that is added to a diagram will have a Node ID of Grp. The second **Start Groups** node added to a diagram will have a Node ID of Grp2, and so on.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Start Groups window. The Imported Data — Start Groups window contains a list of the ports that provide data sources to the **Start Groups** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Start Groups window. The Exported Data — Start Groups window contains a list of the output data ports that the **Start Groups** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Start Groups Node Train Properties

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **Start Groups** node. Select the ... button to open a window that contains the variables table. You can set the Use, Report, and Grouping Role values for a variable In the variables table, you can also view the columns metadata and open an Explore window to view a variable's sampling information, observation values, or view a plot of variable distribution.
- **Rerun** — Use the Rerun property to specify whether the portion of the process flow diagram between the **Start Groups** node and the **End Groups** node should rerun each time the process flow is executed, regardless of whether the node or the process flow diagram has run before or not. Setting the Rerun value to **Yes** enables group processing loops to be performed even if there are configuration errors in subsequent nodes downstream in the process flow diagram. Valid values are **Yes** and **No**. The default setting for the Rerun property is **No**.

Start Groups Node Train Properties: General

- **Mode** — specifies the looping mode that you want the **Start Groups** node to use.
  - **Index** — the index mode setting specifies the number of times to loop through the group processing portion of the process flow diagram. No additional sampling or model averaging is performed.
  - **Bagging** — Unweighted resampling for bagging is the most straightforward method of resampling. Unweighted resampling uses random sampling with replacement to create the n sample replicates. You set the number of samples (and in turn, models) that you want to create in the Index Count property in the General section of the Train properties. Unweighted resampling does not apply extra weight to the cases that were misclassified in the previous samples. When

you specify the Bagging mode, you should specify settings for the Type, Percentage, and Random Seed properties in the Bagging properties section.

  - **Boosting**— the boosting mode performs weighted resampling to create boosting models. Boosting models are a modification of bagging models. Boosting models use a frequency variable that has a value proportional to the model residual. Rows that are incorrectly sampled are given higher frequency values, so the next model will consider them more significantly.
  - **Target** — the target mode loops for each target variable that is identified in the training data. The target looping mode creates similar models for a number of targets, such as modeling a competitive cross-sell scenario.
  - **Stratify** — Use the Stratify mode to perform standard group processing. When you use the Stratify mode, the **Start Groups** node loops through each level of group variable when you run the process flow diagram. When you select the Stratify mode, the Minimum Group Size and Target Group properties are enabled.
  - **Cross-Validation** — the cross validation looping mode is a modification of the stratified mode. You use the cross validation mode when you want to export the complement of the groups specified, as opposed to the groups themselves.
  - **No Grouping** — no looping is performed. You might want to suppress the node from looping to reduce the run time during preliminary model building.
- **Target Group** — Use the Target Group property to specify whether to process the target as a separate group. The default setting for the Target Group property is **No**.
- **Index Count** — When the Mode property is set to Index, Bagging, or Boosting, use the Index Count property to specify the number of loops to be performed. The Index Count property accepts positive integers as inputs.
- **Minimum Group Size** — When the Mode property is set to Stratify or Cross-Validation, use the Minimum Group Size to define the minimum number of observations that must be in a group for looping to apply to that group. By default, if a group level has fewer than 10 observations, then that group level is not processed. You can change the Minimum Group Size by entering a different value in the input field. The Minimum Group Size property accepts positive integers as inputs.

Start Groups Node Train Properties: Bagging

The properties in the Bagging group are available only when the Mode property is set to Bagging. Otherwise, the properties in the Bagging group are dimmed and unavailable.

- **Type** — Use the type property to specify the method that you want to use to determine the sample size.
  - **Percentage** — a percentage of the population. When the Type property is set to Percentage, a value must be entered in the Percentage property that specifies the desired proportion of the population to extract as a sample.
  - **Number of Observations** — a discrete number of observations. When the Type property is set to Number of Observations, a value must be entered in the Observations property that specifies the number of observations to extract as a sample.
- **Observations** — Specifies the number of observations to include in the sample when the Type property is set to Number of Observations.
- **Percentage** — Specifies the proportion of observations to include in the sample when the Type property is set to Percentage.

- **Random Seed** — Specifies the random seed value used to randomly sample observations from the input data.

Start Groups Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Start Groups Node Results

Open the Results window by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

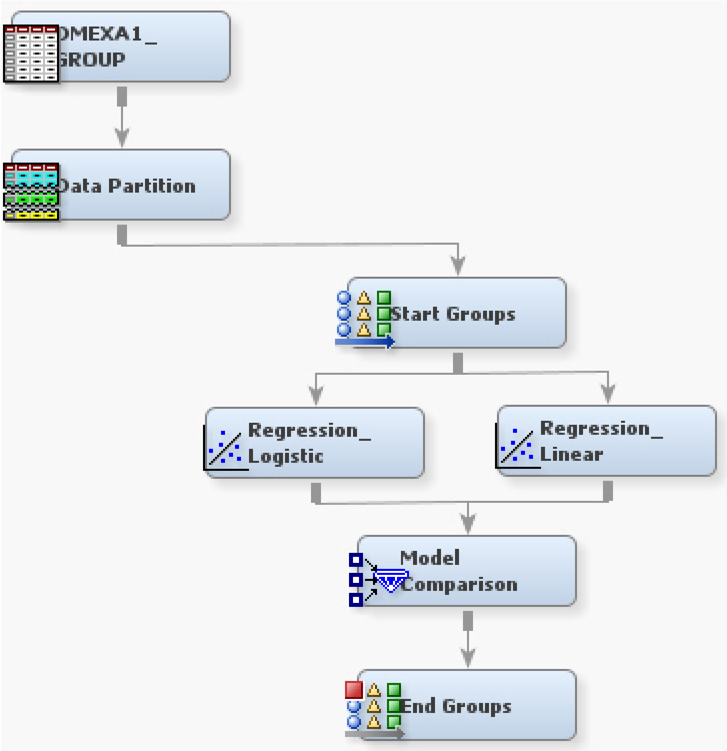
Select **View** from the main menu to view the following results in the Group Processing Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the configuration information in the **Start Groups** node properties panel. The information was captured when the node was last run.
  - **Run Status** — indicates the status of the **Start Groups** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a read-only table of variable meta information about the data set that was submitted to the **Start Groups** node. The table includes columns to see the variable name, the variable role, the variable level, the model used, and the group role used (Name, Role, Level, Use, Group Role).
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays any user-created notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **Start Groups** node run.
  - **Output** — the SAS output of the **Start Groups** node run. Typical output includes information such as a Variable Summary by Role, Level and Count, R-Squares, Chosen Effects, and Analysis of Variance (ANOVA) tables for the target variable, Estimating Logistic and Cutoff Classification tables, Node Split History, Split Effect Summary, and a Summary of Variable Selection. The Variable Clustering example contains additional information about the contents of the SAS Output window.

  - **Flow Code** — the SAS code used to produce the output that the **Start Groups** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the **Start Groups** node does not generate SAS Code. The SAS Code menu item is dimmed and unavailable in the Group Processing Results window.
  - **PMML Code** — the **Start Groups** node does not generate PMML code.
- **Group Processing**
  - **Summary** — a table that contains the Group Index, Group, and Frequency Count.

Start Groups Node Example

As a marketing analyst at a catalog company, you would like to build a logistic regression model for a binary target that indicates whether a purchase is made. You would also like to run a linear regression on an interval target that indicates the purchase amount. Separate models are required for males and females.



1. Add the SAMPSIO.DMEXA1 data set as the input data source. Assign PURCHASE and AMOUNT roles as target variables.
2. Add a **Data Partition** node to the diagram workspace. Use 70% training, 30% validation, and the default random seed value.
3. Add a **Start Groups** node to the diagram workspace. Set the Mode property to Stratify. Use the Start Groups Variables property window to assign the input variable GENDER and the Grouping Role of **Stratification**.

4. Add a **Regression** node to the diagram. Set the Regression Type property to **Logistic Regression**.
5. Add a second **Regression** node to the diagram. Set the Regression Type property to **Linear Regression**.
6. Add a **Model Comparison** node.
7. Add an **End Groups** node.
8. Run the process flow diagram.

Part 16

Node Reference: Applications

Chapter 78

Incremental Response Node

Incremental Response Node

Overview

The Incremental Response node models the incremental impact of a treatment (such as a marketing action or incentive) in order to optimize customer targeting for maximum return on investment. The Incremental Response node can determine the likelihood that a customer purchases a product, uses a coupon, or predicts the incremental revenue realized during a promotional period. The Incremental Response node is located in the **Applications** tab of the Enterprise Miner Nodes toolbar.

To better understand incremental response modeling, suppose that potential customers are divided into the following groups:

- **Persuadables**: These are customers who respond only when they are targeted. Persuadables can also be thought of as true responders.
- **Sure Things**: These are customers who will respond anyway. Marketing outreaches have no measurable impact on this group.
- **Lost Causes**: These are customers who will not respond whether they are targeted or not. Marketing outreaches have no measurable impact on this group.
- **Sleeping Dogs**, or **Do Not Disturb**: These are customers who are *less* likely to respond if they are targeted.

Conventional response models target marketing actions on people who are most likely to buy. However, the cost of this approach is wasted on the **Sure Things** group of customers, who would buy anyway. The only potential customer group that provides true incremental responses is the **Persuadables** group. In order to maximize incremental sales at minimum cost, ideally only the **Persuadables** group should be targeted by a marketing action.

In order to identify the incremental impact associated with a specific direct marketing action, Incremental Response uses control groups to measure the difference in response rate between the treated group and the control group.

In cases where there is a binary outcome, such as purchase / no purchase, incremental response modeling considers the probability of purchase for each customer under two scenarios—treated and non-treated.

In cases where there are two target variables, such as a binary variable for purchase / no purchase, and an interval variable that represents the amount of revenue in the event that purchase occurs, incremental response modeling considers the difference in the amount of expected sales for each customer under treated and non-treated scenarios. This approach can be called an incremental sales model.

Before building the predictive model, the Incremental Response node enables you to perform predictive variable selection in order to identify the variables most likely to maximize the incremental response rate. Variable selection is important in incremental response models because it can improve model stability as well as predictive accuracy. The Incremental Response node uses the Net Information Value (NIV) technique to perform variable selection. NIV measures the differential in information values between the control group and the treatment group for each variable.

Input Data Requirements for the Incremental Response Node

The Incremental Response node requires a binary treatment variable. Since the Incremental Response node measures response differentials between control and treatment groups, the tool must be able to distinguish between data that represents control groups and data that represents treatment groups. The binary treatment variable (0 for control group, 1 for treatment group) indicates which group an observation belongs to.

The Incremental Response node also requires a binary target variable called the response variable. The response variable indicates the outcome of the marketing action on the customer. For example, a purchase response variable would have a value of 0 for no purchase and 1 for purchase.

The Incremental Response node enables you to define an optional interval target variable called the outcome variable. For example, if you wanted to construct an incremental sales model, you would require a binary response variable (purchase / no purchase) and an interval outcome variable (purchase amount).

Missing Values in Incremental Response Node Input Data

The Incremental Response node excludes observations that have missing values from its analytical analysis. If your input data contains significant missing values, consider using the Impute node to replace the missing values in your input data with imputed values.

Incremental Response Node Properties

Incremental Response Node General Properties

The following general properties are associated with the Incremental Response Node:

- **Node ID** — displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Incremental Response node added to a diagram will have a Node ID of IR. The second Incremental Response node added to a diagram will have a Node ID of IR2, and so on.
- **Imported Data** — provides access to the Imported Data — Incremental Response window. The Imported Data — Incremental Response window contains a list of the ports that provide data sources to the Incremental Response node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — provides access to the Exported Data — Incremental Response window. The Exported Data — Incremental Response window contains a list of the output data ports that the Incremental Response node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Incremental Response Node Train Properties

The following Train properties are associated with the Incremental Response Node:

- **Variables** — Use the Variables window to view variable information. Select the ellipsis button to the right of the Variable property to open a variables table. The Use column in the variables table can be used to change the Use status for individual variables in certain models.

  You can view the columns metadata. You can also open an Explore window to view a variable’s sampling information, observation values, or a variable distribution plot. By default, columns for variable Name, Use, Role, and Level are displayed.

You can modify these values, and add additional columns using the following options:

  - **Apply** — Changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the **Apply** button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Report, Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Prescreen Variables** — Set the Prescreen Variables property to Yes if you want to choose a subset of predictive variables before building the model.
- **Rank Percentage Cutoff** — Specifies the percentage of the variables to select if the Prescreen Variables property is set to Yes. Ranks all predictive variables according to the net information value, and selects variables that have higher net information values. The default setting for the Rank Percentage Cutoff property is 50%.

Incremental Response Node Train Properties: Model Selection

- **Combined Model** — set the **Combined Model** property to **Yes** if you want to include the treatment variable as one predictor in the model, instead of running separate models for treatment group and control group. The default setting for the **Combined Model** property is **No**.

  The Combined Model is based on Lo's "The True Life Model — A Novel Data Mining Approach to Response Modeling in Database Marketing" (2002).

- **Selection Method** — specifies the method for selecting effects for the selection process. The default value for the **Selection Method** property is **None**, which specifies no model selection and the complete model is used.
  - **Forward** — starts with no effects in the model and adds effects until the Entry Significance Level is met.
  - **Stepwise** — similar to the Forward method, except that the effects already in the model do not necessarily remain in the model.
  - **Backward** — starts with all effects in the model and incrementally removes effects.
- **Selection Criterion** — chooses from the list of models at each step of the selection process the model that yields the best value for the specified criterion. The default value of the **Selection Criterion** property is **None**, which chooses the model at the final step of the selection process.
  - **AIC** — chooses the model that has the smallest Akaike Information Criterion value.
  - **SBC** — chooses the model that has the smallest Schwarz's Bayesian Criterion value.
  - **Validation Error** — chooses the model that has the smallest misclassification rate or error rate for the validation data set. The misclassification rate is used for binary targets, and the error rate is used for interval targets. The error rate is measured using the sum of squared errors statistic.

- **Cross-Validation Error** — chooses the model that has the smallest misclassification rate or error rate for the cross validation of the training data set. The misclassification rate is used for binary targets, and the error rate is used for interval targets. The error rate is measured using the sum of squared errors statistic.

- **Significance Level for Entry** — specifies the significance level to be used as a threshold criterion for adding input variables. The default value for the Significance Level for Entry property is 0.0, meaning that variables that have a p-value that is less than or equal to 0.05 are added as inputs.

- **Stay Signficance Level** — specifies the significance level to be used as a threshold criterion for removal of input variables. The default value for the Significance Level for Entry property is 0.0, meaning that variables that have a p-value that is less than or equal to 0.05 are removed as inputs.

- **Suppress Intercept** — set the Suppress Intercept property to Yes if you want to suppress the intercept term in the model. The default setting for the Suppress Intercept property is No.

- **Two-Way Interactions** — set the Two-Way Interactions property to Yes if you want to include all two-way interaction terms for class variables and all of the second order polynomial terms of interval variables that have a status of Use. The default setting for the Two-Way Interactions property is No.

Incremental Response Node Train Properties: Revenue Calculation

- **Use Constant Revenue** — set the Use Constant Revenue property to Yes if you want to specify a fixed revenue for each response. This option overrides the expected revenue for individual customers. The expected revenue for individual customers is the estimated outcome from the model. The default setting for the Use Constant Revenue property is No.

- **Revenue Per Response** — specifies the fixed revenue for each response if the Use Constant Revenue property is set to Yes. The default setting for the Revenue Per Response property is 10.

- **Cost** — specifies the cost of the direct marketing action for each contact. The default value of the Cost property is 0. A customer is considered as profitable in the incremental response model only if the incremental revenue is greater than the cost.

Incremental Response Node Report Properties

- **Number of Bins** — specifies the number of bins, *n*. The width of each bin is 100% / *n*.

Incremental Response Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.

- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.

- **Last Error** — displays the error message from the last run.

- **Last Status** — displays the last reported status of the node.

- **Last Run Time** — displays the time at which the node was last run.

- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.

- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Incremental Response Node Results

You can open the Results browser of the Incremental Response node by right-clicking the node and selecting **Results**. For general information about the Results browser, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Incremental Response — Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Incremental Response node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Incremental Response node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Incremental Response node run.
  - **Output** — the SAS output of the Incremental Response node run. The Incremental Response SAS output includes the following:
  - **Flow Code** — the SAS code used to produce the output that the Incremental Response node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Incremental Response node does not generate PMML code.
- **Models**
  - **Selected Variables by NIV** — a table of the selected variables and their ranks in the original variable set according to the net information values.
  - **Response Outcome Summary Table** — a table of summary statistics of the data, such as number of observations, number of responses, response rate, average outcome and total outcome.
  - **Response Outcome Summary** — displays bar charts of the statistics in the Response Outcome Summary Table for both treatment and control groups.

    - **Parameter Estimates for Response Model** — a plot of parameter estimates for the treatment and control groups. Parameter estimates are provided for each input variable that is in the response model.

    - **Parameter Estimates for Outcome Model** — a plot of parameter estimates for the treatment and control groups. Parameter estimates are provided for each input variable that is in the outcome model. The default outcome model is a two-stage model that uses the inverse Mills ratio. The combined model uses separate regressions for the binary and interval target. The interval target is zero unless there is a response.

- **Revenue Plots**

    - **Average Revenue** — a plot that displays the average revenue within the percentile for treatment and control group. The horizontal axis is the percentile based on the rank order of the predicted incremental revenue, The predicted incremental revenue is the difference between the treatment group and the control group.

    - **Incremental Revenue** — a plot that displays the average incremental revenue within the percentile. The increment is the difference of the expected revenue between the treatment group and the control group. The cost per contact is subtracted if it is specified in the Revenue Calculation options.

- **Response Plots**

    - **Treatment Response Model Diagnostics** — a plot that shows the response rate within the percentile for both the predicted treatment and observed treatment.

    - **Control Response Model Diagnostics** — a plot that shows the response rate within the percentile for both the predicted control and observed control.

    - **Cumulative Incremental Response Diagnostics**

        - **Cumulative Observed Increment Plot** — a plot that displays the cumulative incremental response rate for observed data.

        - **Predicted Observed Increment Plot** — a plot that displays the cumulative incremental response rate for predicted data.

    - **Cumulative Increment Response** — a plot that displays the cumulative incremental response rate for both predicted and observed data.

- **Outcome Plots**

    - **Treatment Outcome Model Diagnostics** — a plot that shows the average outcome within the percentile for both the predicted treatment and observed treatment.

    - **Control Outcome Model Diagnostics** — a plot that shows the average outcome within the percentile for both the predicted control and observed control.

    - **Cumulative Incremental Outcome Diagnostics**

        - **Cumulative Observed Increment Plot** — a plot that displays the cumulative incremental outcome for observed data.

        - **Predicted Observed Increment Plot** — a plot that displays the cumulative incremental outcome for predicted data.

    - **Cumulative Increment Response** — a plot that displays the cumulative incremental outcome for both predicted and observed data.

- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard for the selected table. You can modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Incremental Response Node Output Variables

The score code of the Incremental Response node produces the following output variables which can be used for business decisions based on the incremental response model:

- EM_P_TREATMENT_RESPONSE: The predicted response probability from the treatment model.
- EM_P_CONTROL_RESPONSE: The predicted response probability from the control model.
- EM_P_INCREMENT_RESPONSE: The incremental predicted response rate.
- EM_P_ADJ_INCREMENT_RESPONSE: The adjusted incremental predicted response rate, which makes the incremental predicted response rate positive.
- EM_P_ABS_INCREMENT_RESPONSE: The absolute value of the incremental predicted response rate, when an interval outcome target is used with the response variable.
- EM_P_TREATMENT_OUTCOME: The predicted value of the outcome variable from the treatment model.
- EM_P_CONTROL_OUTCOME: The predicted value of the outcome variable from the control model.
- EM_P_INCREMENT_OUTCOME: The predicted value of the incremental outcome.

When an interval outcome target or constant revenue is used with the response target variable, the following output variables are produced:

- EM_REV_TREATMENT: The estimated revenue for the treatment model.
- EM_REV_CONTROL: The estimated revenue for the control model.
- EM_REV_INCREMENT: The estimated incremental revenue.

Incremental Response Node Example

Overview

The following example uses incremental response modeling to examine opportunities for incremental sales as well as to enhance the profitability and effectiveness of a direct marketing campaign. A direct marketing team can use the modeling results to identify and target a reduced number of customers whose purchasing behavior can be changed positively by a treatment such as a sales incentive or a marketing promotion. The model measures the incremental sales that are generated by the promotion as well as examines the incremental customer responses. Incremental customer responses are examined and ordered from the customers who are most positively influenced by a treatment to those who are most negatively influenced by a treatment. The model also provides the estimated incremental revenue that could be realized because of the marketing action.

Create and Configure the Incremental Response Node Data Source

This example uses a data set from the SAMPSIO library called DMRETAIL. DMRETAIL is a training data set of 9,000 observations that documents the purchasing behaviors of customers. Each observation in the data set represents the purchasing history of a customer, including purchases made during the promotional incentive period.

Not all customers receive the promotional marketing treatment. The 9,000 customers are divided evenly into two treatment groups, as indicated by the binary treatment variable value in each observation. Half (4,500) of the customers in the data set are in treatment group 0. Treatment group 0 receives no marketing incentive promotion, but their purchase behaviors are recorded over the period of time that exactly corresponds with the duration of the marketing incentive promotion. The remaining 4,500 customers are in treatment group 1, which receives a treatment in the form of a promotional marketing incentive. Partitioning the data set into equal treatment and non-treatment populations allows the behavioral effect of the marketing promotion on customers to be statistically analyzed and measured.

The DMRETAIL table contains two variables that will be used as targets. A binary target variable Response indicates whether a purchase takes place during the promotional treatment period. An interval target variable Sales indicates the amount of the purchase, if a customer buys. The model results will indicate a customer’s likelihood to make a purchase during the period of interest, as well as the expected amount of the purchase.

The Incremental Response node does not require two target variables. Only a binary target variable for the purchase / no purchase response is required. When you create an Incremental Response node model using only a binary response target variable, the model can only predict the likelihood of a customer to make a purchase during the period of interest.

Use the following steps to convert the SAS table SAMPSIO.DMRETAIL from a data set into an Enterprise Miner data source:

1. In the Enterprise Miner Project Navigator, right-click the Data Sources folder and select Create Data Source. This opens the Data Source Wizard. The default selection for metadata source is SAS Table. Accept the default setting by clicking **Next**.
2. In the field for **Select a SAS table**, type SAMPSIO.DMRETAIL and then click **Next**.
3. The Table Properties display provides summary information about the SAMPSIO.DMRETAIL table. Select **Next** to advance to the **Metadata Advisor Options** portion of the Data Source Wizard. In the **Metadata Advisor Options**, select the **Advanced** radio button and click **Next**.
4. In the **Column Metadata** display of the Advanced Advisor, make the following variable role assignments:
   - Set the role of the variable **Promotion** to **Treatment**. Ensure that the Level for **Promotion** is binary.
   - Set the role of the variable **Response** to **Target**. Ensure that the Level for **Response** is binary.
   - Set the role of the variable **Sales** to **Target**. Ensure that the Level for **Sales** is interval.

- Select **Next**.

5. Accept the default settings for the remainder of the windows in the Data Source Wizard. Select **Next** to advance through each window until you reach the last Data Source Wizard window, and select **Finish**.

6. The SAMPSIO.DMRETAIL data source should appear in your **Data Source** folder in the Enterprise Miner Project Navigator.

Create and Configure the Incremental Response Process Flow Diagram

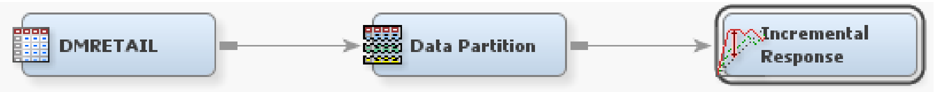
Right-click the Diagrams folder in the Project Navigator and select **Create Diagram**. You can name the diagram whatever you like. Drag the DMRETAIL data source that you just created onto your newly created diagram workspace.

Next, from the **Sample** tab of the nodes toolbar, drag a Data Partition node onto the workspace, and connect the DMRETAIL data source to the Data Partition node. Select the Data Partition node, and then use the Properties Panel to partition the DMRETAIL data into 60% training data and 40% validation data. This example does not use a holdout test data partition.

Drag an Incremental Response node from the **Applications** tab of the nodes toolbar onto the diagram workspace. Connect the Data Partition node to the Incremental Response node.

The default property settings of the Incremental Response node are sufficient for this example. You do not need to make any changes.

Your process flow diagram should resemble the one shown below:

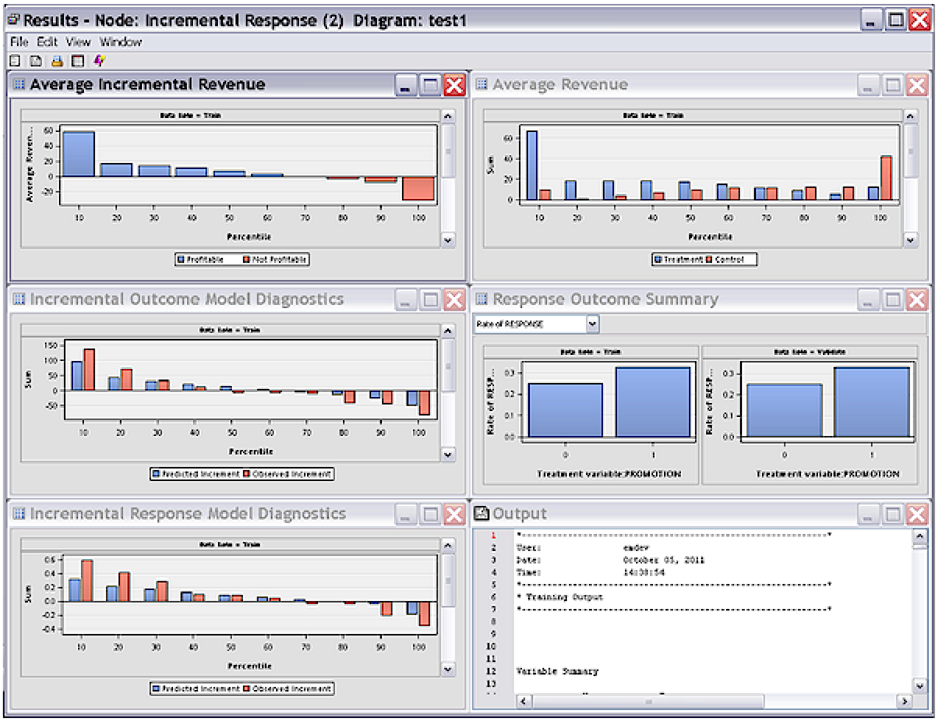


Run the Incremental Response Process Flow Diagram

Right-click the Incremental Response node in the diagram workspace and select **Run**. After the process flow diagram finishes running, click **Yes** to open the Incremental Response node Results browser.

Examine the Incremental Response Model Results

The Incremental Response node Results browser opens to display the Results plots and tables.

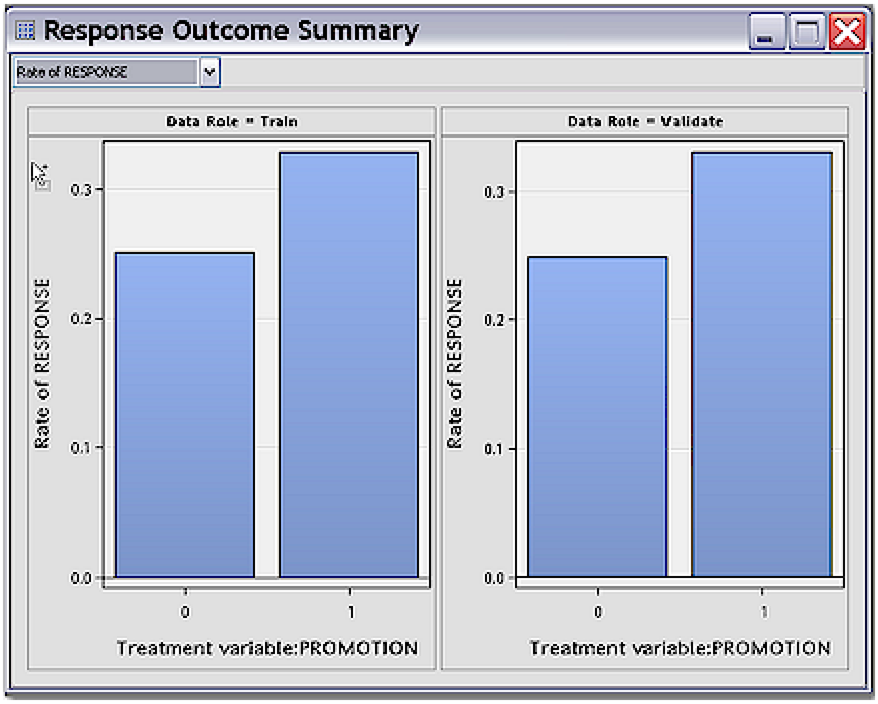


```
*------------------------------------------------------------*
User:               emdev
Date:               October 05, 2011
Time:               14:38:54
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*




Variable Summary
```

They include Response Outcome Summary plot, Average Revenue plot, Average Incremental Revenue plot, Parameter Estimates for Response Model, Parameter Estimates for Outcome Model, Treatment Response Model Diagnostics, Control Response Model Diagnostics, Incremental Response Model Diagnostics, Cumulative Incremental Response Diagnostics, Treatment Outcome Model Diagnostics, Control Outcome Model Diagnostics, Incremental Outcome Model Diagnostics, Cumulative Incremental Outcome Diagnostics, and Selected Variables Table by NIV (Net Information Value).

The default display contains the Average Revenue plot, Response Outcome Summary plot, Average Incremental Revenue plot, Incremental Outcome Model Diagnostics plot, and Incremental Response Model Diagnostics plot. To access other plots and tables, click the **View** tab and see Model section, Revenue Plots section, Response Plots section, or Outcome Plots section.

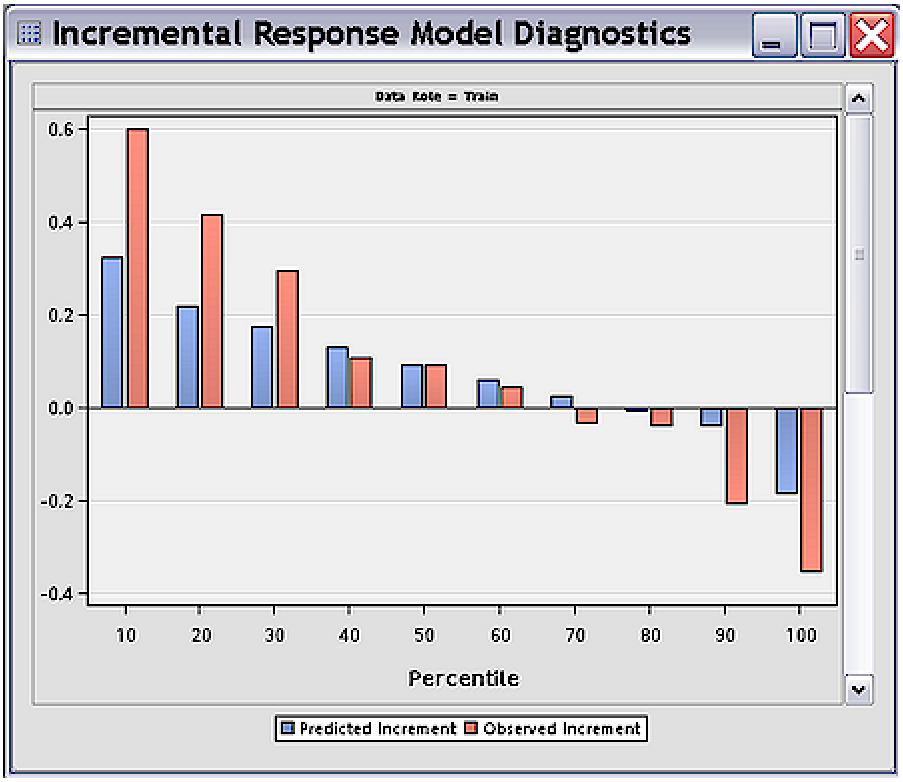
First, examine the Response Outcome Summary plot. By default, it opens to the Rate of Response selection.



This plot shows the rate of response (purchases) across the control and treatment groups. The responses across the training and validation data appear to be uniform, indicating little variation in the characteristics of the two partitioned data sets. The y-axis represents percent response.

The control group, which received no treatment, displays a 25.08% response. Approximately 1 in 4 customers made a purchase without any marketing treatment. The treatment group indicates a 32.91% response, or approximately 1 in 3 customers made a purchase in the group that received a marketing incentive promotion. The customers in the treatment group are not homogenous: as in the control group, some customers would purchase whether or not they received a marketing treatment, while other customers in the treatment group presumably purchased because of the marketing incentive that they received. Comparing the response rates of the control group to the treatment group, it would appear that approximately 7.83% (32.91% — 25.08%) of the customers in the treatment group were “true” responders to the marketing promotion. This 7.83% represents the average incremental response rate.

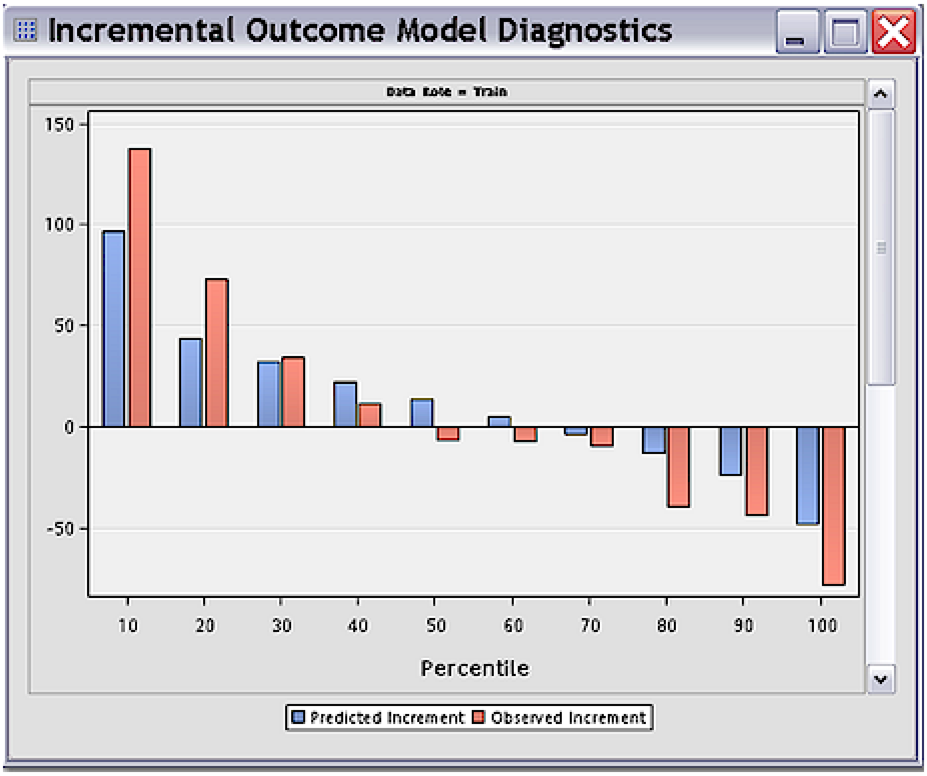
Next, examine the Incremental Response Model Diagnostics plot. The Incremental Response Model Diagnostics plot displays both the predicted and observed incremental response rate. The increment is calculated by subtracting the control response rate from the treatment response rate.



This plot displays incremental response by deciles, with the strongest response decile on the left (the top 10% of respondents) to the weakest response decile on the right (the weakest 10% of respondents). The predicted incremental response rate by percentile in blue displays a declining pattern. The observed increment response rate in red displays a similar trend. The top 10% of customers have the observed incremental response rate of 59.77% and the expected incremental response rate 32.29%, which are much higher than the average incremental rate of 7.83% indicated in the Response Outcome summary statistics mentioned above.

The results demonstrate that the model has the potential to pick up a significant portion of the customers that are most likely to be positively influenced by the promotion. It provides a guideline for targeting the Persuadables, helps justify the expense of the marketing promotion, and indicates the ability to minimize the risk of negative responses to the promotion.

Next, examine the Incremental Outcome Model Diagnostics plot. The Incremental Outcome Model Diagnostics plot displays both the predicted and observed incremental outcomes. The increment is calculated by subtracting the control outcome from the treatment outcome.

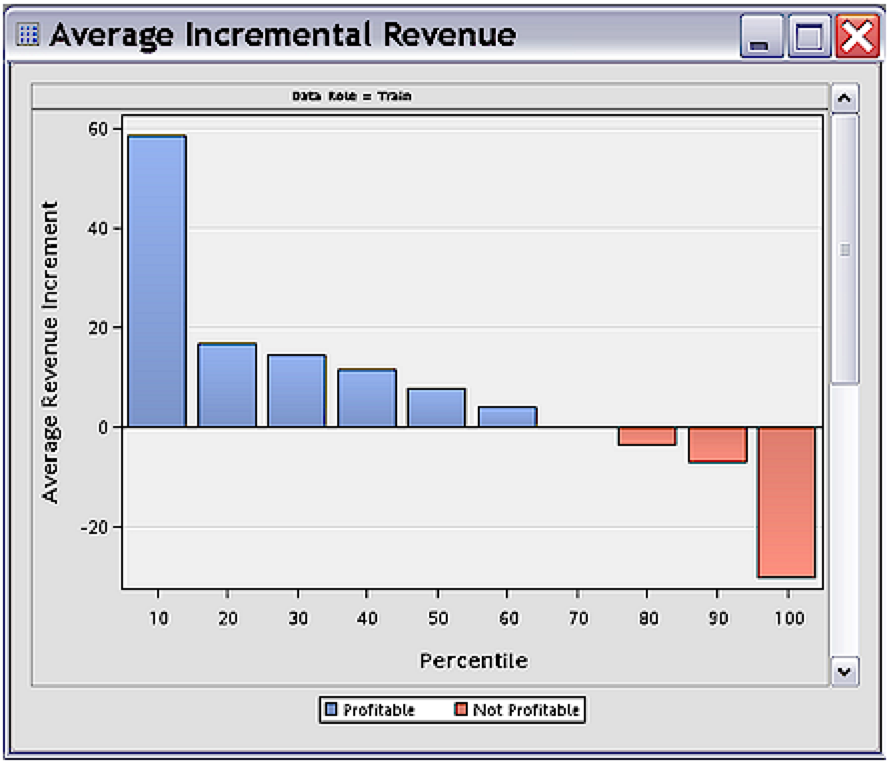

Incremental Outcome Model Diagnostics
Data Role = Train
150
100
50
0
-50
10
20
30
40
50
60
70
80
90
100
Percentile
🟦 Predicted Increment 🟥 Observed Increment

This plot evaluates the model performance from the sales perspective. If the customer makes any purchase during the promotion period, the purchase amount is included as a target variable and a two stage model is run to estimate the outcome as well. The top percentile contains the customers who are most likely to spend more if a marketing incentive is provided. The top 10% of customers indicate an observed incremental outcome of $137.19, with an expected incremental response rate of $96.76. These are both much higher than the average incremental rate of $1.42.

Note: The average incremental rate of $1.42 can be found in the Response Outcome Summary plot by setting the plot selector in the upper left corner to Average Sales. Examining the chart for Average Sales, we see that the average sales for the treatment group (PROMOTION = 1) is $158.95, and the average sales for the control group (PROMOTION = 0) is $157.53. The resulting difference of $1.42 represents the average incremental sales rate.

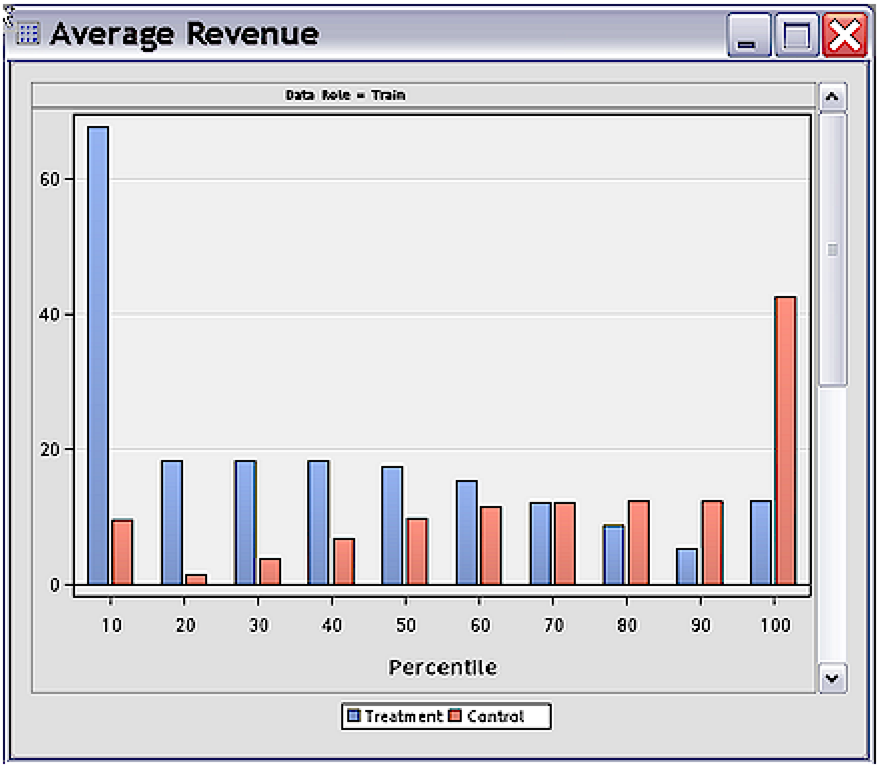
These results help us identify the customers who display the maximal increment in sales from a much larger group that includes the customers who will make purchases whether or not a promotion was offered.

Next, examine the Average Incremental Revenue plot. The Average Incremental Revenue plot intuitively visualizes the incremental revenue realized from the marketing action when viewed across the customer deciles after considering the cost of the promotional marketing treatment.



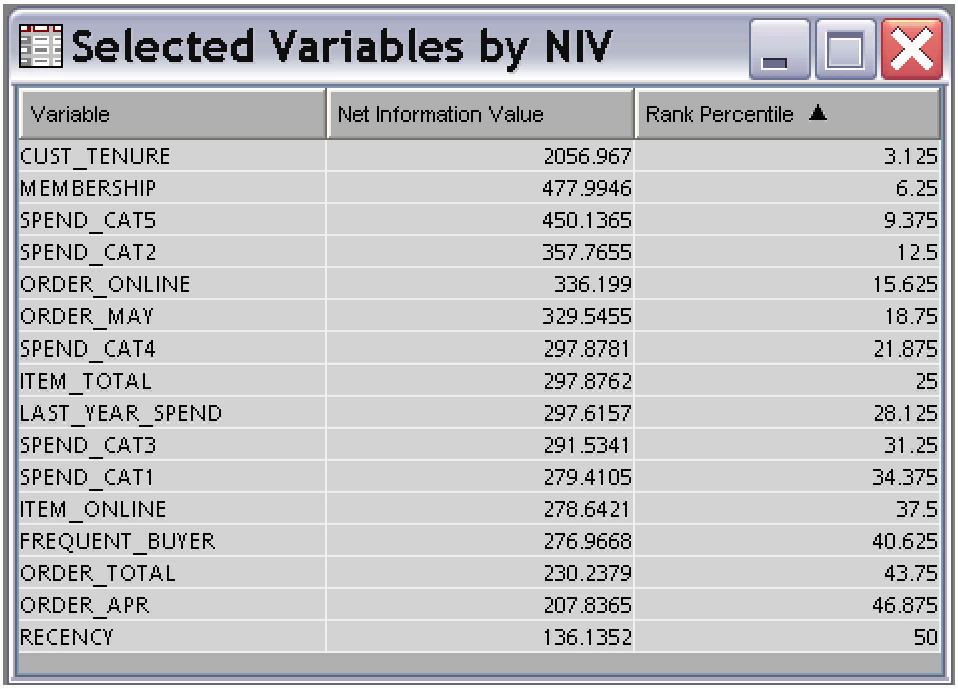
A customer is considered profitable only if the incremental gross revenue is higher than the contact cost. The first 60% of customers indicate positive net revenue increments, and are considered to be profitable. For example, the top 10% of customers have an expected revenue increment of $58.28. A negative value indicates the negative effect of the promotion on individuals such as the sleeping dogs. So it is also important for the campaign or marketing action to target that class of customers for omission from promotional treatment. These results not only quantify expected revenue by customer segment, but also help identify the customer classes for whom incremental promotional costs only result in further income loss.

Next, examine the Average Revenue plot:



The Average Revenue plot provides a decile-by-decile breakdown of the average revenue for treatment and control groups. Customers are rank ordered the same way as in the Average Incremental Revenue plot, from the most profitable deciles on the left to the least profitable deciles on the right.

Next, examine the Selected Variables by NIV (net information value) table in the Incremental Response node Results.


Selected Variables by NIV

Variable	Net Information Value	Rank Percentile ▲
CUST_TENURE	2056.967	3.125
MEMBERSHIP	477.9946	6.25
SPEND_CAT5	450.1365	9.375
SPEND_CAT2	357.7655	12.5
ORDER_ONLINE	336.199	15.625
ORDER_MAY	329.5455	18.75
SPEND_CAT4	297.8781	21.875
ITEM_TOTAL	297.8762	25
LAST_YEAR_SPEND	297.6157	28.125
SPEND_CAT3	291.5341	31.25
SPEND_CAT1	279.4105	34.375
ITEM_ONLINE	278.6421	37.5
FREQUENT_BUYER	276.9668	40.625
ORDER_TOTAL	230.2379	43.75
ORDER_APR	207.8365	46.875
RECENCY	136.1352	50

When you set the Prescreen Variables property of the Incremental Response node to Yes before running, the Selected Variables Table displays the top 50% of input variables ranked by NIV score. The NIV score indicates the variables that have the strongest correlation to model responses. The net information value is calculated as the difference in information values between the treatment and control groups for each input variable. The proportion of variables that is selected for inclusion in the table by NIV ranking can be specified in the Rank Percentage Cutoff property for the node.

References

Lo, Victor S.Y. 2002. "The True Lift Model — A Novel Data Mining Approach to Response Modeling in Database Marketing." *SIGKDD Explorations* Volume 4, Issue 2, pp 78–86.

Chapter 79

Ratemaking Node

Ratemaking Node

Overview

Ratemaking is the process of determining what rates, or premiums, to charge each customer for their insurance. While traditional ratemaking methods are statistically unsophisticated, the Ratemaking node uses generalized linear models, a proven technique, to analyze data and create a ratemaking model. Generalized linear models (GLMs) are extensions of traditional linear models and allow the population mean to depend on a linear predictor through a nonlinear link function.

When you create a GLM using the Ratemaking node, you can model traditional insurance measures such as claim frequency, severity, or pure premium. Claim frequency is typically modeled with a Poisson distribution and a logarithmic link function. Claim severity is typically modeled with a gamma distribution and a logarithmic link function. Pure premiums are modeled with the Tweedie distribution. The table below lists the available distributions and link functions in the Ratemaking node.

Distribution	Available Link Functions
Poisson	Logarithmic
Negative Binomial	Logarithmic
Gamma	Logarithmic, Inverse, Power Squared, Power Cubed

Distribution	Available Link Functions
Binary	Logistic
Normal	Logarithmic, Identity, Inverse, Power Squared, Power Cubed
Inverse Gaussian	Logarithmic, Identity, Inverse, Power Squared, Power Cubed
Tweedie	Logarithmic
Zero-Inflated Poisson	Logistic, complementary Log-Log

The Ratemaking node supports binary and interval targets. The Ratemaking node does not support nominal character variables. The input variables can be interval, binary, or nominal. You can specify only one target variable and you will receive an error message if more than one target variable is specified.

Data Set Requirements for the Ratemaking Node

The Ratemaking node requires that all inputs be binned. The number of levels associated with a rating variable implies the number of prices in the rating structure and the granularity of risk segmentation. Therefore, careful consideration must be given to the binning method and the number of bins that are used. See "Train Properties: Interval Variable Binning Options" on page 1216 for specific binning options.

Additionally, the Ratemaking node requires that all target variables are either binary or interval variables. If your input data set contains a target variable that is not binary or interval, then you will receive an error when running the Ratemaking node.

Properties of the Ratemaking Node

Ratemaking Node General Properties

The following general properties are associated with the Ratemaking node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Ratemaking node that is added to a diagram will have a Node ID of HPG. The second Ratemaking node added to a diagram will have a Node ID of HPG2, and so on.
- **Imported Data** — accesses the Imported Data — Ratemaking window. The Imported Data — Ratemaking window contains a list of the ports that provide data sources to the Ratemaking node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Exported Data** — accesses the Exported Data — Ratemaking window. The Exported Data — Ratemaking window contains a list of the output data ports that the Ratemaking node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and the variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Ratemaking Node Train Properties

- **Variables** — Use the Variables property of the Ratemaking node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. For the Ratemaking node, the Variables table contains the additional column **Ratemaking Role**. This column is used to control the use status for individual variables in certain models.
- When **Ratemaking Role** is set to **Default**, the variable is included in the model statement.
- When **Ratemaking Role** is set to **Model**, the variable is included in the model statement.
- When **Ratemaking Role** is set to **Model and ZIP**, the variable is included in the model and the zeromodel statement.
- When **Ratemaking Role** is set to **Zip**, the variable is included in the zeromodel statement only.
- When **Ratemaking Role** is set to **Offset**, the variable is used as an offset variable. Offset variables must be numeric variables.
- When **Ratemaking Role** is set to **Weight**, the variable is used as a weight variable. Weight variables must be numeric variables. Valid weight variable values are larger than 0. Negative weight variable values are deleted from the analysis.

Note: The **Ratemaking Role** values that correspond to **Model and ZIP** and **ZIP** are specific when modeling a *Zero-Inflated Poisson (ZIP)* model. A ZIP model is typically used to predict claim counts. It is normal for insurance data to contain a large proportion of observations that have zero values for claim count variables. The ZIP approach fits two separate models, and then combines them. First, a logit model is fit to determine whether an observation contains a zero count claim variable, or not. Then a Poisson model is used to fit observations with nonzero values for the claim count variables (claim counts of 1, 2, 3, and so on.) The two models are then combined.

Train Properties: Interval Variable Binning Options

- **Binning Method** — Use the Binning Method property to specify the method to use for pre-binning of interval variables.

  Choose one of the following methods:

  - **Quantile** — generates groups formed by ranked quantities with approximately the same frequency in each group.
  - **Bucket** — generates groups by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.

- **Number of Bins** — Specify the number of bins that you want to use when pre-binning interval input variables. The number of bins can range from 5 to 50 and defaults to 20.

- **Set Reference Level** — Select **Default** to use the variable level that has the most observations. Select **User Defined** to choose the reference level for each variable, via the **Reference Level** property.

- **Reference Level** — The **Reference Level** property becomes available only when **Set Reference Level** is set **User Defined**. Select the ... button to the right of the **Reference Level** property to open the Reference Level window. You must run the Ratemaking node before you can access the Reference Level window. The drop down menu in the upper-left corner enables you to select a variable to change its reference level.

  The four columns in the Reference Level window are **Name**, **Variable Value**, **Frequency**, and **Reference**. The **Variable Value** column displays the levels that were created for that variable. The **Frequency** column indicates the number of observations in that level. When you click on a cell in the **Reference** column, you can select **Yes** if you want to use the corresponding level as your reference variable and **No** if you do not want to use that level.

Train Properties: Ratemaking Model Type

- **Model Type** — specifies the model that is used to predict claims.

  - **Count** — uses a Poisson distribution with log link and requires an interval target
  - **Pure Premium** — uses a default Tweedie distribution.
  - **Severity** — uses a Gamma distribution with log link.
  - **User Defined** — uses the probability distribution and link function specified by the user in the corresponding options.

- **Maximum Iterations** — specifies the maximum number of iterations for all iterative computations. Valid values are integers between 1 and 100,000.

- **Probability Distribution** — specifies the user-defined probability distribution.

  - **Poisson** — requires an interval target variable.
  - **Negative Binomial** — requires an interval target variable.
  - **Gamma** — requires an interval target variable.
  - **Binary** — requires a binary target variable.
  - **Normal** — requires an interval target variable.
  - **Inverse Gaussian** — requires an interval target variable.
  - **Tweedie** — requires an interval target variable.

  - **Zero Inflated Poisson** — requires an interval target variable.
- **Link Function** — specifies the user-defined link function.
  - **Log** — uses a logarithmic link function that is available for Poisson, Negative Binomial, Gamma, Normal, Inverse Gaussian, and Tweedie distributions.
  - **Logit** — uses a logistic link function that is available for Binary and Zero-Inflated Poisson distributions.
  - **Identity** — use the identity link function that is available for Normal and Inverse Gaussian distributions.
  - **Inverse** — uses the inverse link function that is available for Gamma, Normal, and Inverse Gaussian distributions.
  - **Power Squared** — uses a quadratic link function that is available for Gamma, Normal, and Inverse Gaussian distributions.
  - **Power Cubed** — uses a cubic link function that is available for Gamma, Normal, and Inverse Gaussian distributions.
- **ZIP Model Options** — Select the ... button to the right of the ZIP Model Options property to open the ZIP Model Options window. The ZIP Model Options window contains properties that are specific to the ZIP distribution.
  - **ZIP Link Function** — specifies the link function that you want to use. You can choose from **LOGIT** (logistic) or **CLOGLOG** (complementary log-log).
  - **Variables for the binomial part of the Zero-Inflated Poisson Model** — specifies how to use the variables for the binomial part of the Zero-Inflated Poisson model.
    - **Automatic Select** — uses a stepwise regression that chooses certain variables for the logistic model. In this case, the **Minimum R-square** and **Stop R-square** values correspond to the entry and exit significance levels for the rating variables, respectively.
    - **Full Model** — selects all variables that are in the logistic part of the model for the Poisson part of the model. For each variable that you want to use, you must set the value of the **Ratemaking Role** property to **Default**, **Model**, or **Model and Zip**. To change the value of **Ratemaking Role**, you must use the **Variables** property.
    - **User Defined** — enables you to specify that you want certain variables in the logistic part and not in the Poisson part of the model. For example, assume that you have selected **User Definied** for this property and set the **Ratemaking Role** column for a particular variable to **ZIP**. That variable is used only in the logistic part of the model and not in the Poisson part.
  - **Minimum R-square** — specifies a lower bound for the individual R-square value of a variable to be included in the variable selection process for the binomial part of the Zero-Inflated Poisson model.
  - **Stop R-square** — specifies the lower bound for the incremental model R-square value.
- **Tweedie Model Options** — Select the ... button to the right of the Tweedie Model Options property to open the Tweedie Model Options window. The Tweedie Model Options window contains properties that are specific to the Tweedie distribution.
  - **Tweedie Optimization** — specifies the optimization method for the Tweedie distribution.

  - **Automatic** — estimates in three steps. First, based on heuristic sampling rules, the data set is sampled and an approximation is used to obtain the model estimates. Next, the full model is evaluated on the sample set using the estimates from the first step as initial parameter values. Finally, the entire data set is used and an approximation is used to obtain the final model estimates.
  - **Saddlepoint** — estimates in two steps. First, based on heuristic sampling rules, the entire data set is used to obtain an approximation of the model estimates. Next, the entire data set and an approximation are used to obtain the final model estimates using the values from the previous step as initial parameter values.
  - **Likelihood** — estimates in two steps. First, the full model is evaluated on a sample of data. Next, the full model is evaluated on the entire data set using the model estimates from the previous step as initial parameter values.
  - **Final Likelihood** — estimates in four steps. First, based on heuristic sampling rules, the data set is sampled and an approximation is used to obtain the model estimates. Next, the full model is evaluated on a sample set using the estimates obtained from the first step as initial parameter values. Third, the entire data set is used to generate an approximation of the model estimates using the values from step two as initial parameter values. Finally, the full model is evaluated on the entire data set using the estimates from step three as initial parameter values.
- **Standard Errors for Tweedie** — specifies how standard errors are treated.
  - **Automatic** — chooses to use either the standard from the approximation or the full model based on the size of the problem. For large data sets, the procedure will take standard errors from the approximated model.
  - **Approximate** — uses the standard errors from the approximated model.
  - **Yes** — standard errors are computed no matter what optimization method is chosen.
  - **No** — standard errors are not computed no matter what optimization method is chosen.
- **Variance Power** — specifies the variance power for the Tweedie distribution. The variance of a Tweedie distribution is μ^p, where the variance power *p* is also referred to as the shape parameter. That is, as *p* changes, the Tweedie distribution changes to better resemble either a gamma or Poisson distribution. Use this property to avoid estimation of the variance power and to set it directly. Valid values are real numbers between 1.1 and 1.999.
- **Initial Starting Value for the Variance Power** — specifies an initial value for the variance power estimation. Valid values are real numbers between 1.1 and 1.999.

Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.

- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Ratemaking Node Results

After a successful node run, you can open the Ratemaking node Results browser by right-clicking the node, and then selecting **Results** from the pop-up menu. For general information about the Results browser,, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu in the Results — Ratemaking window to view the following results in the Results browser:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Ratemaking node properties configuration when the node was last run.
  - **Run Status** — displays the status of the Ratemaking node run. Information about the run start time, run duration, and completion status are displayed in this window.
  - **Variables** — displays a table of the variables in the training data set.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays notes of interest, such as data or configuration information.
- **SAS Results**
  - **Log** — the SAS log of the Ratemaking run.
  - **Output** — the SAS output of the Ratemaking run.
  - **Flow Code** — the SAS code used to produce the output that the Ratemaking node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS environment in custom user applications.
  - **PMML Code** — the Ratemaking node does not generate PMML code.
- **Model**
  - **Fit Statistics** — a table of the fit statistics from the model.
  - **Effects Plot** — displays a bar chart of the absolute values of the coefficients in the final model. The bars are color coded to indicate the algebraic signs of the coefficients.
  - **Parameter Estimates** — displays a table of the binned variables' parameter estimates and the Chi-Square score for each estimate.
  - **Relativity Plots** — The Relativity Plot is a plot of a variable's relativity against the binned variable values.

  - **Actual versus Fitted** — displays a bar chart of the actual target variable frequency and the model's target variable frequency.
  - **Variables Selected for Binomial Part of Model** — provides incremental R-square and p-value information for the selected variables.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — use the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Ratemaking Node Example

Overview

This example uses the SAS sample data set SAMPSIO.CLAIM_HISTORY. The SAMPSIO.CLAIM_HISTORY data set is found in the SAMPSIO example data library that is included with your copy of SAS Enterprise Miner. This example presumes that you have already created a project and diagram in SAS to contain the example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

- Create and Configure Ratemaking Data Source
- Create and Configure the Ratemaking Process Flow Diagram
- Run the Ratemaking Process Flow Diagram
- Examine the Ratemaking Node Results

Create and Configure Ratemaking Data Source

Use the SAS Enterprise Miner Data Source Wizard to create a SAS Enterprise Miner Data Source from the sample SAMPSIO.CLAIM_HISTORY. In the SAS Enterprise Miner Project Panel, right-click the **Data Sources** folder and select **Create Data Source** to launch the Data Source Wizard.

In the Data Source Wizard, do the following to create your SAS Enterprise Miner Data Source:

1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter **SAMPSIO.CLAIM_HISTORY** in the **Table** field and click **Next**.
3. Continue to the Metadata Advisor and choose the Advanced Metadata Advisor.
4. In the Column Metadata window, make the following variable role assignments:
   - Set the role of the variable **ID** to `ID`.
   - Set the roles of the variables **CLAIM_FLAG** and **CLM_AMT** to `Rejected`.
   - Set the role of the variable **CLM_FREQ** to `Target`. Set the **Level** of the **CLM_FREQ** variable to **Interval**.
   - All of the remaining variables should be set to the role of `Input`.

   Click **Next**.

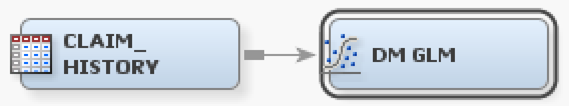
5. Accept the default settings for the remainder of the windows in the Data Source Wizard. Select **Next** to advance through each window until you reach the last Data Source Wizard window, and select **Finish**.
6. The SAMPSIO.CLAIM_HISTORY data set should appear in your **Data Source** folder in your SAS Enterprise Miner Project Panel.

The SAMPSIO.CLAIM_HISTORY data set that you just created has 10,302 observations. Each observation is a unique auto-insurance policyholder. The target variable that you are modeling is **CLM_FREQ**, which represents the number of claims that each policyholder has. This example uses the Ratemaking node to build a model that predicts the number of claims for each policyholder. The model uses input variables, such as age of the car, blue book value of car, type of car, and other information to predict values for the number of anticipated claims by each policy holder.

Create and Configure the Ratemaking Process Flow Diagram

Drag the CLAIM_HISTORY data source that you just created from the Data Sources folder in the Project Panel onto a blank Diagram Workspace.

Next, drag a Ratemaking node from the Applications Toolbar onto the Diagram Workspace and connect the CLAIM_HISTORY data source to the Ratemaking node as follows:



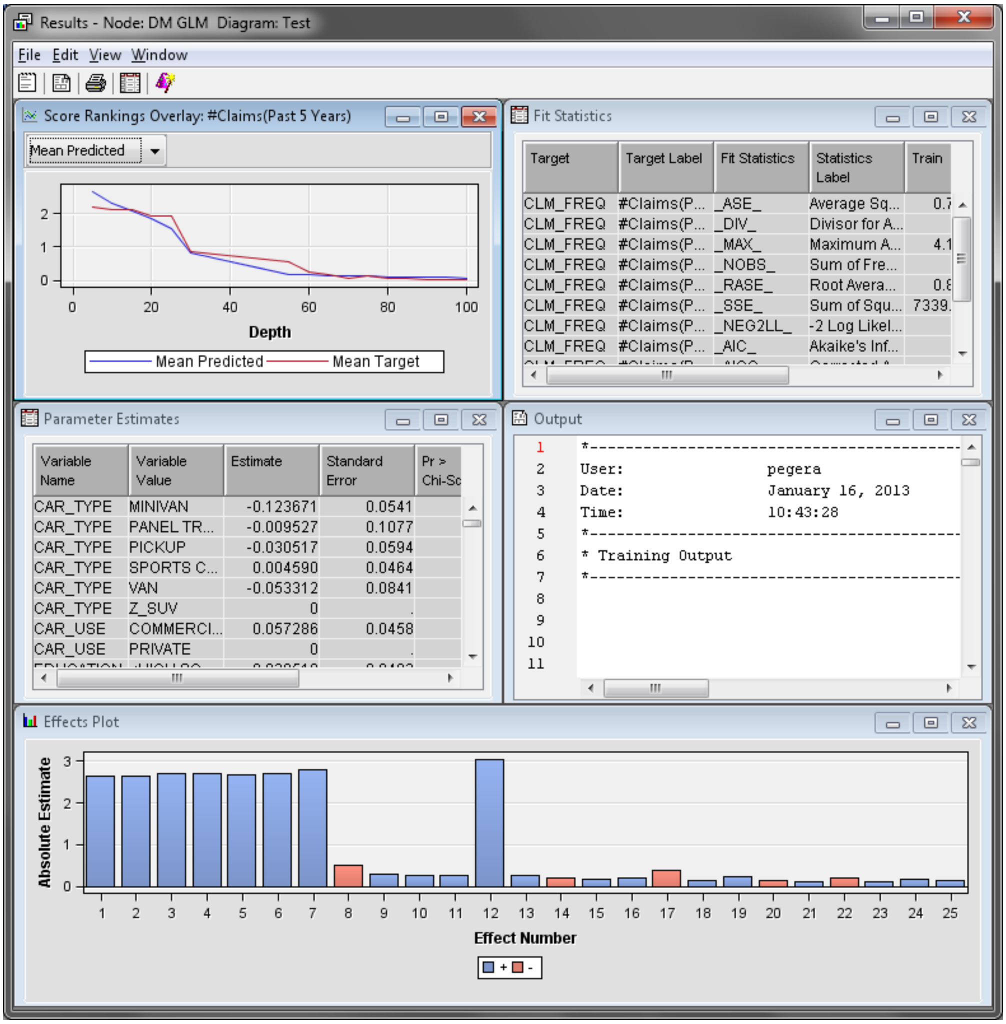
Select the Ratemaking node in the diagram. Check the Properties Panel for the Ratemaking node and ensure that the **Model Type** property is set to **Count**. You configured the target variable **CLM_FREQ** as an interval variable in the Data Source Wizard. As a result, the Ratemaking node will create a claim count model that assumes a Poisson distribution and a logarithmic link function.

Run the Ratemaking Process Flow Diagram

Right-click the Ratemaking node in the Diagram Workspace and select **Run**. Select **Yes** in the Confirmation window. After the process flow diagram finishes running, click **Results** to open the Ratemaking Results browser.

Examine the Ratemaking Node Results

The Ratemaking Results browser displays a number of plots and tables. They include anActual versus Fitted plot, an Effects plot, Relativity Plots, a Fit Statistics table, a Parameter Estimates table, and the SAS Output listing.



Target	Target Label	Fit Statistics	Statistics Label	Train
CLM_FREQ	#Claims(P...	_ASE_	Average Sq...	0.7
CLM_FREQ	#Claims(P...	_DIV_	Divisor for A...	
CLM_FREQ	#Claims(P...	_MAX_	Maximum A...	4.1
CLM_FREQ	#Claims(P...	_NOBS_	Sum of Fre...	
CLM_FREQ	#Claims(P...	_RASE_	Root Avera...	0.8
CLM_FREQ	#Claims(P...	_SSE_	Sum of Squ...	7339.
CLM_FREQ	#Claims(P...	_NEG2LL_	-2 Log Likel...	
CLM_FREQ	#Claims(P...	_AIC_	Akaike's Inf...	

Variable Name	Variable Value	Estimate	Standard Error	Pr > Chi-Sc
CAR_TYPE	MINIVAN	-0.123671	0.0541	
CAR_TYPE	PANEL TR...	-0.009527	0.1077	
CAR_TYPE	PICKUP	-0.030517	0.0594	
CAR_TYPE	SPORTS C...	0.004590	0.0464	
CAR_TYPE	VAN	-0.053312	0.0841	
CAR_TYPE	Z_SUV	0	.	
CAR_USE	COMMERCI...	0.057286	0.0458	
CAR_USE	PRIVATE	0	.	

```
*------------------------------------------------------------
User:                    pegera
Date:                    January 16, 2013
Time:                    10:43:28
*------------------------------------------------------------
* Training Output
*------------------------------------------------------------
```

Actual versus Fitted plot

The **Actual versus Fitted** plot is a bar chart that shows how well the model has performed at classifying each level of the target. In other words, the plot demonstrates how well the model predicts claim counts. The Ratemaking node uses the predicted value of CLM_FREQ to export the expected claim count for each policyholder. The predicted frequency claim count is a rational number, and the raw frequency claim count is an integer. The Ratemaking node uses the integer function to round the predicted frequency claim count. If you specify an Offset variable, then there will be no **Actual versus Fitted** plot.

Relativity Plots

When the Ratemaking node builds a log link model, it produces a relativity bar plot for each variable in the model. The relativity is computed by taking the exponent of the parameter estimate for the associated bin. There are shaded bands around the relativity plot to show the upper and lower bounds for a confidence interval. When you mouse over a point on the relativity plot, a tooltip will display the variable value and the upper and lower bounds of the confidence interval.

Fit Statistics

The Fit Statistics table contains the set of fit statistics that are specific to a generalized linear model. One key statistic for Ratemaking claim count models is the deviance statistic. The deviance statistic is equal to the Pearson chi-square statistic divided by the number of degrees of freedom used in the model. As negative values of the deviance statistic approach 1.0, then more likely the fit of a Poisson distribution. When values of the deviance statistic exceed 1.0 and are greater, then the model distribution is more likely to be overdispersed.

In this example, the value of the deviance statistic is 1.38, indicating there is a problem with overdispersion.

Overdispersion occurs frequently with Poisson count models. In a Poisson distribution, the count mean is equal to the count variance. Real count data are often more spread out, which means that the variance count is larger than the mean count, which means in turn that the data are overdispersed. A possible explanation of the overdispersion might be an important but missing important rating variable. The negative binomial distribution, based on the Poisson distribution, relaxes the assumption that the mean and variance are equal. Therefore, it can deal with overdispersed data. In some count modeling situations, the negative binomial distribution might be better than the Poisson distribution.

Parameter Estimates

The **Parameter Estimates** table contains the parameter estimates and associated p-values for each input variable that is in the model. Additionally, there are lower and upper bound confidence interval numbers reported for each of the estimated levels in the input variables.

Effects Plot

The **Effects Plot** displays the 25 most significant variables in the model. The ranking is based on the p-values of the input variables.

For a more detailed case study, please see the following SAS Global Forum paper, which is also available at **`http://support.sas.com/resources/papers/proceedings11/153-2011.pdf`**.

References

Anderson, Billie. 2011. “Ratemaking Using SAS Enterprise Miner: An Application Study.” *Proceedings of the Thirty-Sixth SAS Global Forum*, Cary, NC, http://support.sas.com/resources/papers/proceedings11/153-2011.pdf.

Chapter 80
Survival Node

Survival Node

Overview of the Survival Node

Survival data mining is the application of survival analysis to data mining problems that concern customers.. The application to the business problem changes the nature of the statistical techniques. The issue in survival data mining is not *whether* an event will occur in a certain time interval, but *when* the next event will occur.

The SAS Enterprise Miner **Survival** node performs survival analysis on mining customer databases when there are time-dependent outcomes. Some examples of time-dependent outcomes are as follows:

- customer churn
- cancellation of all products and services
- unprofitable behavior
- server downgrade or extreme inactivity

The time-dependent outcomes are modeled using multinomial logistic regression. The discrete event time and competing risks control the occurrence of the time-dependent outcomes.

The discrete event time represents the duration from the inception (start) time until the censoring date. Discrete event times are represented by nonnegative integer values. Functions of the discrete event time (in the form of cubic spline basis functions) are used as predictors in the multinomial regression. The hazard and subhazard functions that the **Survival** node generates are based on the multinomial logistic regression. The hazard function is a conditional probability of an event at time *t*. The hazard and subhazard functions depend on the discrete event time. Many times the discrete event time function is nonlinear in nature. Transforming the event time function with cubic spline basis functions allows the hazard and subhazard functions to be more flexible. This results in a greater ability to detect and model customer behavior patterns.

A key element of survival data mining is the concept of *competing risks*. In a traditional medical survival analysis, a patient is considered “lost to follow up” if the patient dies, or moves out of state. The same concept also applies to survival data mining.

An example of competing risks in survival data mining is found in voluntary and involuntary churn. For example, some customers are forced to leave when their account remains delinquent too long. Other customers leave voluntarily by transferring selected services or by changing service providers altogether. Any action that terminates the customer’s relationship with a provider is a competing risk. After a customer has experienced one of the competing risk events, that customer is excluded from the at-risk population for any of the other competing risks. In other words, competing risks are mutually exclusive and exhaustive.

Censoring is a nearly universal feature of survival data. Censoring occurs in many forms for many different reasons. When the data is extracted, usually not all customers have experienced the event. An observation is considered to be censored if the event has not yet occurred by the end date that is chosen.

All of the modeled events are time-dependent because the probability distribution of the time until the event controls their occurrence.

The **Survival** node includes functional modules that prepare data for mining, that expand data to one record per time unit, and perform sampling to reduce the size of the expanded data without information loss. The **Survival** node also performs survival model training, validation, scoring, and reporting.

Input Data Requirements for the Survival Node

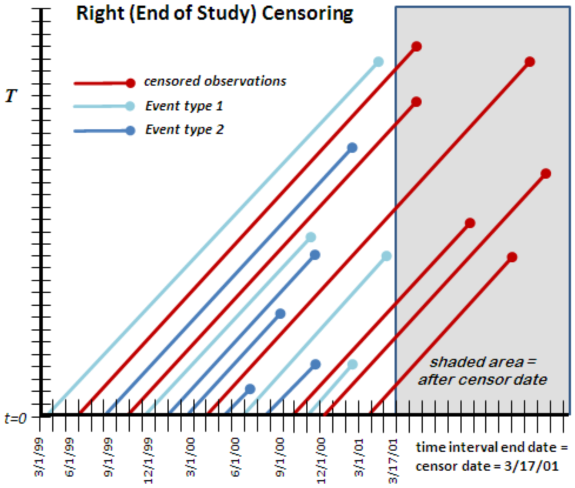
The **Survival** node requires that the input data submitted for analysis meet the following requirements:

- The input data must have a unique ID variable (such as customer ID) for observations.
- At least two TIMEID variables are required. The first TIMEID variable maps to the inception, or start date. The second TIMEID variable maps to the event date.
- The event TIMEID variable must be generated before data is submitted to the **Survival** node for analysis. The event TIMEID variable cannot be the result of a function that uses other columns in the data set as inputs.
- At least one input variable is required for predictive hazard modeling using the **Survival** node.

- All input variables must be time independent. The **Survival** node does not support interactions between variable inputs and time at this point.

There must be one target variable that represents the type of event that occurs on the event date. The target variable is subject to the following constraints:

- Target variable values must be numeric.
- The target variable must be a class variable. Interval values are not permitted.
- The target variable must represent the different outcome levels that are associated with an event. Event outcome levels are discrete values that belong to a mutually exclusive and exhaustive set. For example, if you are modeling churn, you might use three target variable levels. Voluntary churn could be represented by a target variable value of 1. Involuntary churn could be represented by a target variable value of 2. Observations that do not contain a churn event within the defined analysis interval could be represented by a target variable value of 0. Observations that have a target variable value of 0 are said to be *censored* because no churn event occurred during the analysis interval. Censoring is a term that describes observations for which the exact event time is unknown. Churn events that might occur after the analysis interval are not relevant to the analysis. The target variable value for all censored observations must be 0.



Overview of Discrete Time Logistic Hazard Modeling Using the Survival Node

Use of the SAS Enterprise Miner **Survival** node in data mining follows a specific process:

- **Prepare the Data for Survival Analysis**
- **Expand the Data for Survival Analysis**
- **Sample the Expanded Data**
- **Configure and Run the Survival Model**
- **Validate the Survival Model**
- **Specify Reporting Options**
- **Model Scoring**

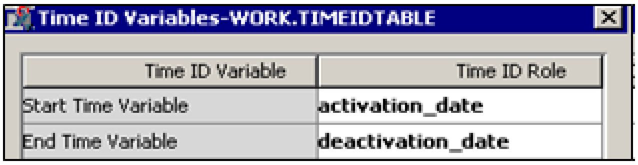
Prepare the Data for Survival Analysis

The **Survival** node supports three types of input data formats: **Standard**, **Change-Time**, or **Fully-Expanded**. The following narrative uses the **Standard** data format. You use the **Change-Time** and **Fully-Expanded** data formats when your model uses time-dependent covariates. a structure requiring multiple rows per ID variable showing changing covariable values. For more information about Survival analysis with time-dependent covariates, see “Survival Modeling with Time-Dependent Covariates” on page 1236,

The data to be mined for survival modeling must be configured for survival analysis in the **Survival** node properties. You must identify the time ID variables, as well as the basic unit of time that will be the basis for censoring the data. You use the SAS Enterprise Miner **Survival** node properties **Time ID variables** and **Time Interval** to specify the required information.

Specify **Time ID variables**: You must configure your survival analysis data set so that SAS Enterprise Miner can identify the variable roles that are required to perform the time interval heuristics.

The **Time ID variables** and **Time Interval** properties specify how the input data to be analyzed is censored. Selecting the ... button to the right of the **Time ID variables** property enables you to specify the mapping for the start and end date variables, as follows:



Time ID Variables-WORK.TIMEIDTABLE

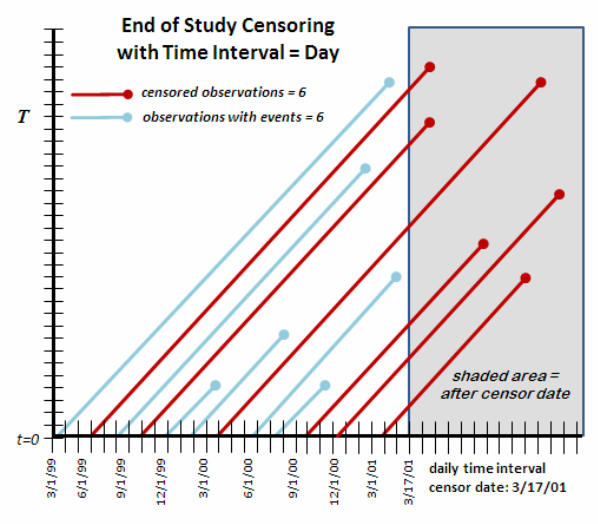
Time ID Variable	Time ID Role
Start Time Variable	activation_date
End Time Variable	deactivation_date

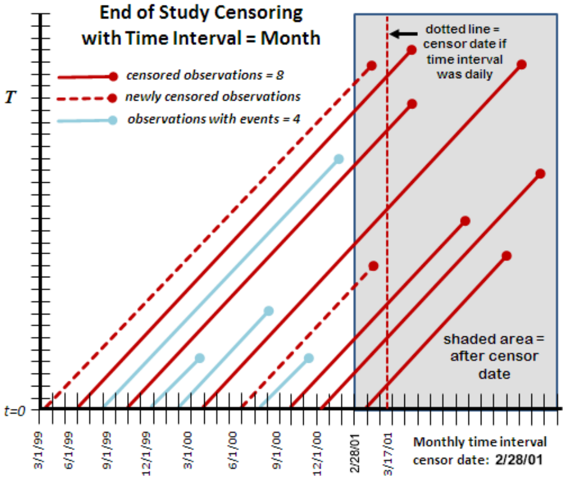
SAS Enterprise Miner scans the input data and chooses the start date for modeling. It does so by finding the minimum date value across all observations for the variable specified as the **Start Time Variable**. SAS Enterprise Miner chooses the modeling end date by finding the maximum date value across all observations for the variable that you specify in the **End Time Variable** field. The censoring date is based on the maximum date for the **End Time Variable** and the **Time Interval** that you selected.

You use the **Time Interval** property to specify the time interval that you would like to use for censoring and reporting. The available time intervals are as follows:

- Day
- Week
- Month
- Quarter
- Semi-year
- Year

To see how the **Time ID variables** and **Time Interval** properties work in conjunction to censor the data, consider the following examples:



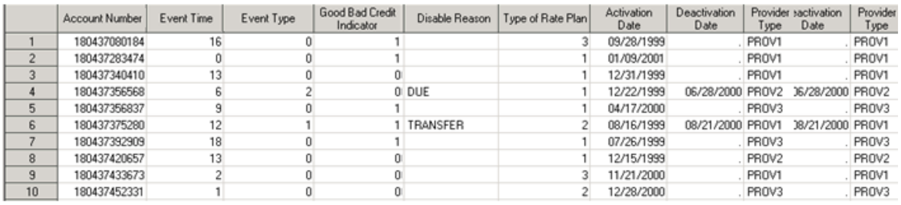


Note that the two data plots have different censor dates due to the chosen time period. Remember that the **Survival** node chooses the modeling end date by finding the maximum date value across all observations for the variable that you specify in the **End Time Variable** field. When the **Time Interval** property is configured as *Day*, the maximum date value for the day time unit is March 17, 2001. When the **Time Interval** property is configured as *Month*, the maximum date value for the month time unit is February 28, 2001.

The difference in censor dates (related to the choice of time interval units) can affect the data to be analyzed and modeled. Note that the example graph for the daily time interval contains multiple event types. There are 9 voluntary churn events, 6 involuntary churn events, and 6 instances where no event is observed before the censor date. In the example graph for the monthly time interval, the interval numbers vary: there are 9 voluntary churn events, 4 involuntary churn events, and 8 instances where no event is observed before the censor date. When months are specified as the time interval, the change in the censor date due to the larger time units affects the event type classifications. Two customers experienced involuntary churn events on March 15, 2001 when the time interval is specified in days and the censor date is March 17, 2001. However, if the time interval is specified in months, the two customer events on March 15 occur *after* the monthly censor date of February 28, 2001. This changes the event type classification for those two customers from involuntary churn events to censored.

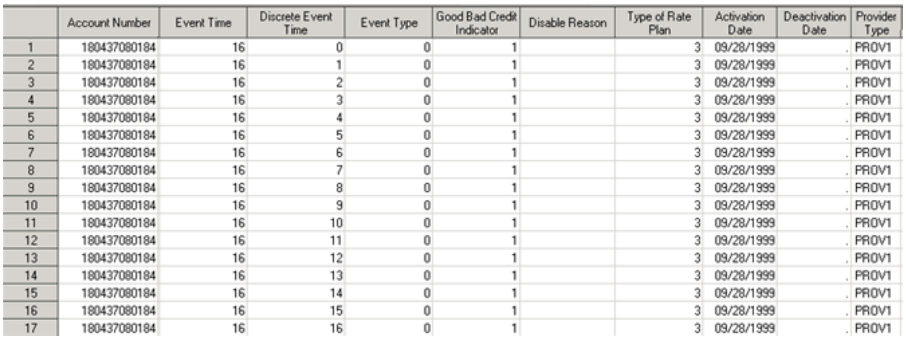
Expand the Data for Survival Analysis

The training data set for the **Survival** node requires one observation for each unique customer (account number in the example below). The time span for the training data set ranges from the start time to the end time (censoring time). After you configure your data for survival analysis by specifying the **Time ID variables** and **Time Interval** properties, the **Survival** node expands the training data set. The training data set is expanded so that each customer has one record for each incremental time interval in which the customer was observed. For example, the following shows the first ten observations in the training data with one observation per ID value (account number):



	Account Number	Event Time	Event Type	Good Bad Credit Indicator	Disable Reason	Type of Rate Plan	Activation Date	Deactivation Date	Provider Type	activation Date	Provider Type
1	180437080184	16	0	1		3	09/28/1999	.	PROV1	.	PROV1
2	180437283474	0	0	1		1	01/09/2001	.	PROV1	.	PROV1
3	180437340410	13	0	0		1	12/31/1999	.	PROV1	.	PROV1
4	180437356568	6	2	0	DUE	1	12/22/1999	06/28/2000	PROV2	6/28/2000	PROV2
5	180437356837	9	0	1		1	04/17/2000	.	PROV3	.	PROV3
6	180437375280	12	1	1	TRANSFER	2	08/16/1999	08/21/2000	PROV1	8/21/2000	PROV1
7	180437392909	18	0	1		1	07/26/1999	.	PROV3	.	PROV3
8	180437420657	13	0	0		1	12/15/1999	.	PROV2	.	PROV2
9	180437433673	2	0	0		3	11/21/2000	.	PROV1	.	PROV1
10	180437452331	1	0	0		2	12/28/2000	.	PROV3	.	PROV3

The 0 value in the **Event Type** column of the first observation means that the customer did not experience an event between the start time and the censoring date. The **Event Time** column for the customer in observation 1 shows that that customer has been active for 16 time periods (months). In the expanded data set shown below, the single observation for customer 1 becomes 17 observations, with one row per observed time period (month).



	Account Number	Event Time	Discrete Event Time	Event Type	Good Bad Credit Indicator	Disable Reason	Type of Rate Plan	Activation Date	Deactivation Date	Provider Type
1	180437080184	16	0	0	1		3	09/28/1999	.	PROV1
2	180437080184	16	1	0	1		3	09/28/1999	.	PROV1
3	180437080184	16	2	0	1		3	09/28/1999	.	PROV1
4	180437080184	16	3	0	1		3	09/28/1999	.	PROV1
5	180437080184	16	4	0	1		3	09/28/1999	.	PROV1
6	180437080184	16	5	0	1		3	09/28/1999	.	PROV1
7	180437080184	16	6	0	1		3	09/28/1999	.	PROV1
8	180437080184	16	7	0	1		3	09/28/1999	.	PROV1
9	180437080184	16	8	0	1		3	09/28/1999	.	PROV1
10	180437080184	16	9	0	1		3	09/28/1999	.	PROV1
11	180437080184	16	10	0	1		3	09/28/1999	.	PROV1
12	180437080184	16	11	0	1		3	09/28/1999	.	PROV1
13	180437080184	16	12	0	1		3	09/28/1999	.	PROV1
14	180437080184	16	13	0	1		3	09/28/1999	.	PROV1
15	180437080184	16	14	0	1		3	09/28/1999	.	PROV1
16	180437080184	16	15	0	1		3	09/28/1999	.	PROV1
17	180437080184	16	16	0	1		3	09/28/1999	.	PROV1

Creating an expanded data set can easily result in very large training data sets that are impractical to use for modeling. The **Survival** node provides a sampling method that you can use to reduce the size of the training data set while minimizing the loss of information.

Sample the Expanded Data

Expanding the modeling event data to represent one customer record per unit time can quickly create very large input data tables that are impractical to use for modeling. Use the **Survival** node sampling properties to specify a sampling method that will reduce the size of the time series training table and minimize the loss of information.

To enable sampling in the **Survival** node, set the **Sampling** property to **Yes**. Use the **Event Proportion** property to specify the proportion of events that you want to include in your sample. The default setting for the **Event Proportion** property is 0.20, or 20%.

For example, suppose you have a 10% event rate in your training data, but you would like a 20% event rate in your sample. The **Survival** node can increase the relative occurrence of events in a sample data set by eliminating observations from the expanded data that do not have an event. The **Survival** node sampling algorithm does not remove any observations that have event outcomes. Only observations with no event are removed during sampling. Here is an overview of how the **Survival** node sampling algorithm operates:

1. Let π represent the training data event rate, which is the proportion of observations in the fully expanded training data. In our example, π = 0.10. Let ρ represent the desired proportion of events in the sample. In our example, ρ = 0.20.
2. The **Survival** node performs heuristic checks to ensure that the expanded training data set contains enough events to support the desired proportion of events in the sample to be created. If ρ < π, then the algorithm sets ρ = π. Otherwise, ρ remains equal to the proportion specified in the properties panel.
3. All observations with events are retained. The remaining observations not associated with an event are randomly sampled with selection probability P(non-event observations) = [((1 - ρ)π)/((1 - π)ρ)]. Because all observations with events are retained, the probability that a given observation in the training data set is selected for the sample is P(event observations) = 1.

   For a training data set with event rate π = 0.10 and a desired sampled data set with event rate ρ = 0.20, the probability for a given non-event observation to be selected from the training data as part of the sample data set is [((1- 0.20)*0.10)/((1 - 0.10)*0.20)] = 0.44 = 44%. The sample data set will be created from the training data by selecting all observations with an event, and 44% of the observations with no event selected at random. The model is then built with the biased sample data set as the training data.

Using the biased sample to train the model allows the predicted subhazard functions to be more precisely estimated than compared to a random sample. To correct the bias caused by sampling, the subhazard functions are adjusted after the model is built. The logit function for each of the subhazard functions is adjusted by adding the log of the selection function: ln[((1 - ρ)π)/((1 - π)ρ)]. The adjustment factor for the subhazard functions can be viewed in the output scoring code of the **Survival** node.

This sampling technique is a well-known method that is used to handle rare categorical outcomes, and is known as *case-control* or *choice-based* sampling.

Configure and Run Survival Model

The basis of survival data mining is the hazard and subhazard functions. These functions represent the chance that a customer account that has existed for a given span of time is going to cancel service before the next unit of time. The subhazard function simply represents the conditional probability of an event occurrence of type *n* at time *t*, given that no event has occurred before time *t*. The **Survival** node uses observations that have a target value of 0 (observations with no observed event, or censored observations) as the reference level. The hazard and subhazard functions can be suitably modeled using multinomial regression models.

The discrete time variable determines the shape of the hazard function. The multinomial regression model should include an appropriate parameterization of time. The discrete time variable is often non-linear in nature. One effective approach that allows for more flexibility in the time effect is to use regression spline terms as transformations of the discrete time effect. The transformations are included in the multinomial regression. The specific discrete time transformations that the **Survival** node uses are cubic spline basis functions. The cubic spline basis functions are of the following form:

$$csb(t,k_j) = \begin{cases} -t^3 + 3k_j t^2 - 3k^2 t & \text{if } t \leq k_j \\ -k^3 & \text{if } t > k_j \end{cases}$$

The preceding assumes that *j* is the number of knots and *k* is the value of the knot.

The cubic spline basis functions are segmented functions that consist of polynomials. The join points between the segments are called knots. The knots are points where the function makes a transformation. For example, a knot is the point at which one of the cubic spline basis functions changes from a cubic function to a constant function. Use the **Number of Knots** property in the Regression Spline Model section of the **Survival** node properties panel to specify how many knots the cubic spline basis functions will have.

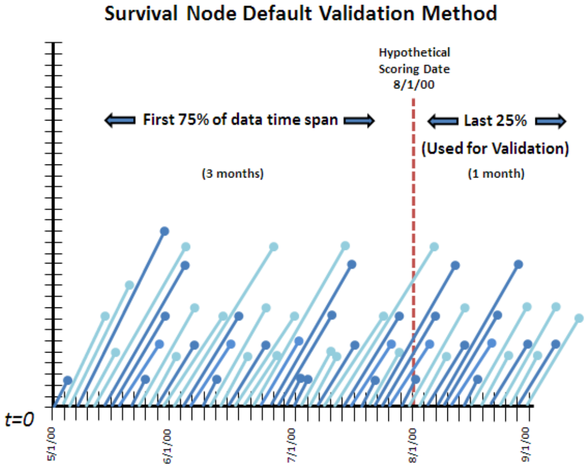
The Regression Spline Model section of the **Survival** node properties panel also contains a **Stepwise Regression** property. When you set the **Stepwise Regression** property to Yes, the multinomial regression model that is created will use stepwise regression. Use the **Entry Significance Level** and **Stay Significance Level** properties to specify the p-value settings for model effects. When you configure the **Survival** node to perform stepwise regression, the discrete time effect is not part of the stepwise variable selection procedure. You can choose to include cubic spline basis functions in the stepwise regression procedure if you set the **Knot Selection** property to **Yes**. Otherwise, discrete time effects and cubic spline basis functions are always used as inputs in the multinomial regression model.

Validate the Survival Model

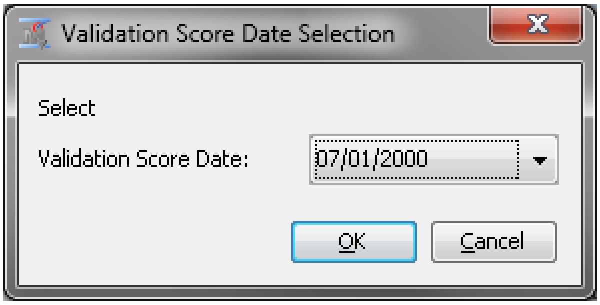
The measure of validity for models that you create is based on the model's performance while using a subset of the data. Survival model validation can be challenging, because event outcomes are time dependent. It is normal for some of the survival data in your validation range to be censored. This is because some observations in the validation data will not have outcomes within the allotted time interval. Data for the **Survival** node is organized chronologically, so the **Survival** node validates survival models by using a subset of the time interval. For each of the imported data sets for the **Survival** node (Train, Validate, and Test), a subset is taken based on the Model Validation properties that define the hypothetical scoring date and scoring interval as described below. Model validation (based on the model that was built using the entire training data set) is performed with the subset of observations in the scoring interval from the train data set, and when available, subsets from the validation and test data sets are also used.

The **Survival Validation Method** property has two settings: **Default** and **User Specified**. The **Default** method automatically assigns values to the **Validation Score Date** and **Interval Length** properties for you. The **User-Specified** method enables you to specify your own values for **Validation Score Date** and **Interval Length**.

The **Validation Score Date** property represents a hypothetical scoring date within the time interval that is represented by the data used for model validation. If the **Survival Validation Method** is set to **Default**, then the time interval used for model validation is divided into quarters. The date most closely associated with the beginning of the last quarter of the interval automatically becomes the hypothetical scoring date (**Validation Score Date**) for the data used for model validation.



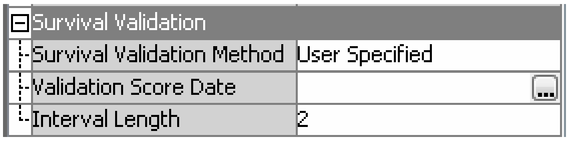
If the **Survival Validation Method** is set to **User Specified**, you can use the **Validation Score Date** property to open the Score Date Selection window. Here you can manually specify your own hypothetical scoring date from the time interval that is represented by the data used for validation.


Validation Score Date Selection

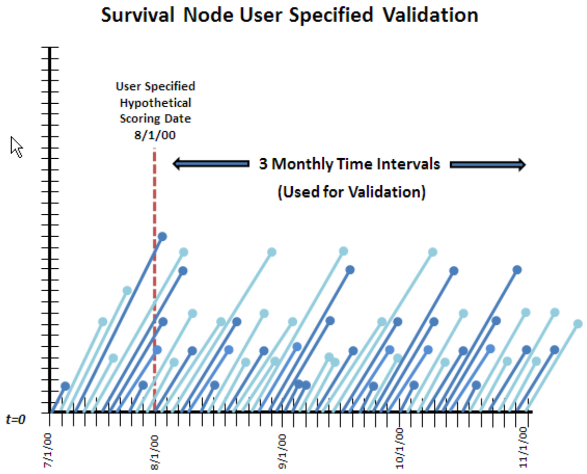
Select

Validation Score Date: 07/01/2000

The **Interval Length** property represents the span of time that follows the hypothetical scoring date to be used for model validation. If the **Survival Validation Method** is set to *Default*, then the **Interval Length** property is automatically set to the last quarter of the model validation data interval. This interval is the time remaining between the automatically assigned **Validation Score Date** and the end of the time interval that is represented by the data used for validation. If the **Survival Validation Method** is set to **User Specified**, you must use the **Interval Length** property to define the time interval to be used for model validation. You do so by entering a scalar multiple of the time unit that is defined in the Survival Train **Time Interval** property.


Survival Validation	
Survival Validation Method	User Specified
Validation Score Date	...
Interval Length	2

For example, if the Train **Time Interval** property is set to *Month*, then entering a value of *3* for the **Interval Length** property would result in a validation interval of three months. The three-month validation interval begins immediately after the hypothetical scoring date, or **Validation Score Date**.



CAUTION:

If you manually specify values for the Validation Score Date and Interval Length properties, the Interval Length end point must fall within the validation data time interval.

Specify Reporting Options

Use the Reporting properties for the **Survival** node to specify reporting options for your model output tables. Report tables provide information about the customers with the highest event probabilities for training, validation, and test data sets.

The **Survival** node report tables identify the customers with the highest likelihood for occurrence of churn, or the given event of interest. You can use the **High Risk Account Tables** option to specify how to report on high-risk customers. The **High Risk Account Tables** settings are as follows:

None
No report tables are generated. When **High Risk Account Tables** is set to **None**, the remaining properties in the Survival **Report** section become dimmed and unavailable.

Event Probability
The report table sorts customers by descending probability of experiencing the event of interest.

Survival Probability
The report table sorts customers by descending probability of survival.

Hazard Probability
The report table sorts customers by descending overall hazard probability.

All
All three report tables (Event Probability, Survival Probability, and Hazard Probability) are generated.

The reporting tables are useful for studying customer retention. For example, the top 100 customers with the highest survival probabilities could be sent a promotion to retain them as customers.

If you want to report the top number of customers, use the **Fixed Numbers** setting for the **Risk Account Selection** property. This enables the **Number** property, which you use to specify the number of customers that you would like summarized in your report table.

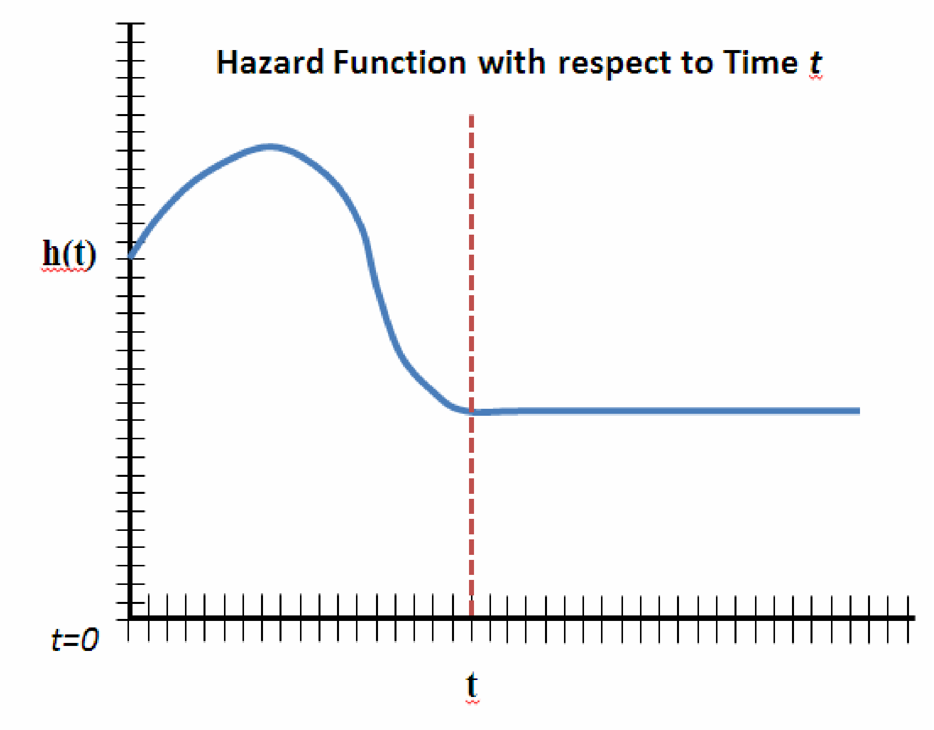
If you would like to report on high-risk customers using percentiles, use the **Percentiles** setting for the **Risk Account Selection** property. This enables the**Percentile** property, where you can choose the top *n*th percentile of customers that you would like included in the report.

Model Scoring

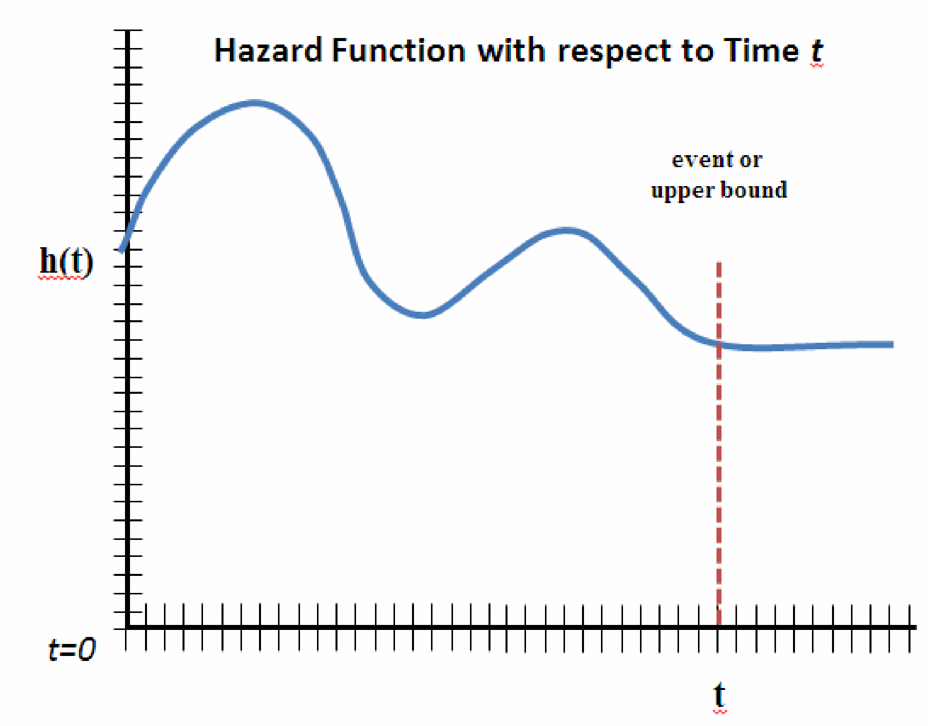
The last step in the Survival data mining model process is to score the data. Use the **Survival** node Score properties to configure your model's scoring output.

You use the first three Score properties for the **Survival** node (**Mean Residual Life**, **Default Maximum MRL**, and **User-Specified Maximum MRL**) to specify options for Mean Residual Life (MRL). In survival data mining, *mean residual life* is the time that remains until an event occurs, such as an existing customer's terminating a business relationship.

The SAS Enterprise Miner **Survival** node computes two types of mean residual life, constant hazard extrapolation and the restricted mean. The constant hazard extrapolation MRL assumes that at some point in time *t*, the hazard function becomes constant, as shown in the graph below:



For many business survival modeling problems, it makes more sense to use a restricted mean MRL. The restricted mean MRL goes to a constant hazard function when either the event of interest occurs, or when some upper limit is reached. The maximum value (or upper bound) often represents the limit on a meaningful remaining lifetime, such as a cable TV or cell phone contract duration. For example, a housing loan might be bounded by the terms of a mortgage. The mortgage might be 30 years in duration, but some customers pay off the loan in 15 years. All customers who paid off their loan in 15 years would have a constant hazard function value from the 15-year point going forward. This allows customers whose residual life (time remaining until the event) exceeds the time horizon to be considered equal. The following plot illustrates a restricted mean MRL:



If you use the restricted mean MRL, you can specify a maximum (upper bound) by using the **Default Maximum MRL** and **User Specified MRL** properties.

The future value of a customer depends on the remaining lifetime. Mean residual life is concerned with scoring future time intervals with the hazard model. If you compute mean residual life, the **Default Forecast Intervals** and **Number of Forecasted Intervals** properties specify the number of time units into the future that the mean residual life function will use for calculations.

When you specify how many time units into the future you would like the mean residual life to be computed, you will also obtain a future survival function as well. For example, if you select *Yes* in the **Default Forecast Interval** property, and set the **Time Interval** property to *Year*, then the survival probabilities are calculated for every customer in the training data set. The survival probability indicates the likelihood that a given current customer will still be a customer one year from the time that the model was trained.

Survival Modeling with Time-Dependent Covariates

Overview

Some survival models need to measure how the values of certain variables related to an ID change over time. For example, Internet, phone, and cable service providers might track the type and number of complaints customers register in order to predict churn, or medical care providers might track the number of symptoms a patient might exhibit prior to death or incapacitation. Problems such as these require data that is formatted with multiple rows per ID, in order to show changing covariate values over time. The **Survival** node uses the **Fully-Expanded** and **Change-Time** data formats to perform time-dependent covariate survival modeling.

All inputs in the **Change-Time** and **Fully-Expanded** data formats are treated as time-dependent. It is not necessary to distinguish any of the variables that do not change over time. For interval inputs, the node uses the weighted average over the time interval. For class inputs, the last recorded value for a time interval is used. When scoring, the current value at censor time is used. Only the last observation for an ID (with the change date before the score or censor time) is used for scoring, validation, or for empirical function evaluation.

Choosing Your Time Dependent Covariate Data Format

You must consider several criteria when choosing the data format for a time-varying covariate survival data mining problem.

Generally, the **Change-Time** format allows for smaller input data sets, and provides greater flexibility results calculations. **Change-Time** data requires records only for the initial start date, and whenever covariate variable values change, as opposed to the standard approach which requires a record for every successive time interval. When you use **Change-Time** formatting, you can use the **Survival** node property settings to manipulate the discrete time interval at which model results are calculated, as well as left truncation dates and right censoring dates.

By contrast, data sets in the **Fully-Expanded** format must be altered with SAS DATA step code or SQL queries in order to affect different discrete time intervals, left truncation dates and right censoring dates. The length of the recorded time interval for data in **Fully-Expanded** format will also influence modeling results. If you want your modeling results to contain the most detail, you should submit the most granular input data via the **Fully-Expanded** data format.

For example, weekly data would typically capture more information than monthly data, monthly data would typically capture more information that quarterly data, and so on. To change an input data set to **Change-Time** or **Fully-Expanded**, see the following detailed instructions.

The INTNX() DATA step function and the EM_SURV_SVEXP macro variable are two tools that you can use to prepare or alter **Change-Time** or **Fully-Expanded** data sets. SAS Enterprise Miner includes three example Survival data sets: SAMPSIO.churn_changetime, SAMPSIO.churn_fullyexpanded_weekly, and SAMPSIO.churn_fullyexpanded_monthly. You can use these three example Survival data sets to experiment with the two data formats using different time intervals.

Note: If you are using **Change-Time** or **Fully-Expanded** data formats, and if your process flow diagram contains a Data Partition node that precedes the **Survival** node, your ID variable must be configured in the **Data Partition** node as a cluster variable, with the partitioning method set to **Cluster**.

Fully Expanded Format Data Requirements

Data in the **Fully-Expanded** data must meet the following requirements:

- An ID variable is required. Generally, an ID variable represents a unique customer or patient identifier. When possible, ID variables should be configured as nominal variables.
- Two TimeID variables are required, a start time and an end time. A time interval index variable named *_t_* is also required. All three variables must share the same date or datetime format. For each TimeID variable in the data set, there must be accompanying rows numbered from 0 to *n* for the time interval index variable, where *n* represents the time interval count from the start date to the end or censor date, whichever comes first.

  CAUTION:

  In general, SAS Enterprise Miner will not process any data set with the role TRAIN that contain variables named with an initial underscore character (_). However, certain data sets with the role SCORE processed by the SAS Enterprise

Miner **Survival** node require a variable named **_t_**. If you are not scoring survival data, creating SAS Enterprise Miner variable names with an initial underscore (_) is generally not recommended, because the software reserves the use of variable names with an initial underscore (_) in all training data sets.

- The time index variable must reflect start and end dates in consistent time index variable units. For example, in a model that uses a monthly time interval, if the start date is March 15 and the end date is April 2, the time index variable must have a row for _t_=0 that corresponds to March 1, and a row for _t_=1 that corresponds to April 1, with the event occurring at _t_=1.
- An ID variable can have only one target variable value.
- Right-censored records in **Fully-Expanded** data should have the end Time ID variable set to ".", the value for numerical missing.
- Use caution when switching to **Fully-Expanded** data that is less granular than the original, more detailed data source. For example, if a customer churns within a short period of time, valuable data regarding time-varying covariates might be compressed into a single, less informative record.
- When using the **Fully-Expanded** data format, the time interval, left truncation date and training time ranges cannot simply be changed by changing Survival node properties. The input data itself must be manipulated to make any changes to the time frame and time interval that the model will use as a basis for calculations and results.

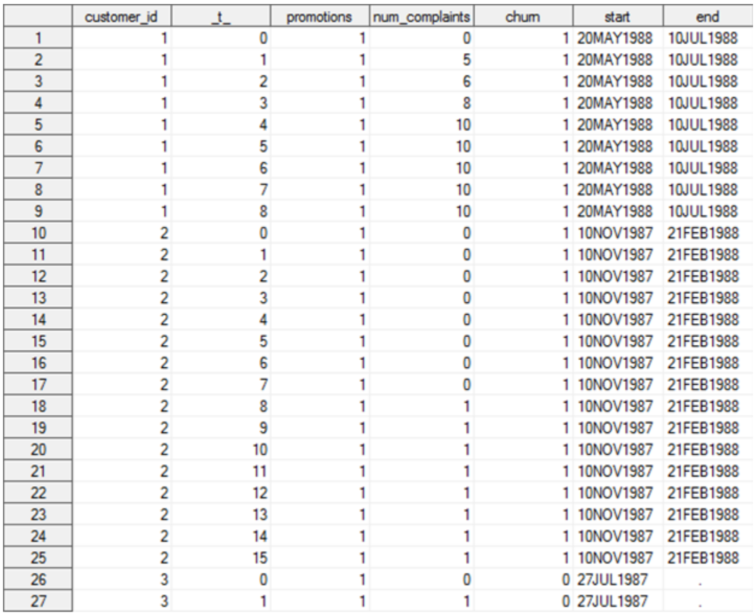
Time Intervals and Fully-Expanded Data Format

Choosing the right time interval for your time-varying covariate survival analysis is profoundly important. Let us look at some fully expanded example churn data sets, recorded at different time intervals, and examine the sample data to see how survival model results might be impacted.

The following example fully-expanded data charts come from two data sets that are included in your SAS Enterprise Miner sample data library, SAMPSIO. One data set contains summarized weekly data, SAMPSIO.churn_fullyexpanded_weekly, and the other data set contains summarized monthly data, SAMPSIO.churn_fullyexpanded_monthly. You can locate the expanded-data format tables by browsing the SAMPSIO library using the SAS Enterprise Miner Data Source Wizard. (For more information about creating SAS Enterprise Miner data sources using the Data Sources Wizard, see "SAS Enterprise Miner Data Sources" on page 287.)

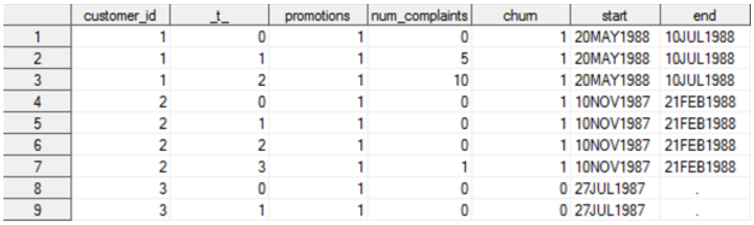
The weekly data contains more detailed figures for the time varying covariate num_complaints. The monthly data contains the same basic trends, but at a less granular level. However, the monthly data will also require less disk and memory resources to process, a potential advantage for modelers that are using extremely large data sets.

Figure 80.1 View of Expanded Weekly Data Table SAMPSIO.churn_fullyexpanded_weekly



	customer_id	_t_	promotions	num_complaints	churn	start	end
1	1	0	1	0	1	20MAY1988	10JUL1988
2	1	1	1	5	1	20MAY1988	10JUL1988
3	1	2	1	6	1	20MAY1988	10JUL1988
4	1	3	1	8	1	20MAY1988	10JUL1988
5	1	4	1	10	1	20MAY1988	10JUL1988
6	1	5	1	10	1	20MAY1988	10JUL1988
7	1	6	1	10	1	20MAY1988	10JUL1988
8	1	7	1	10	1	20MAY1988	10JUL1988
9	1	8	1	10	1	20MAY1988	10JUL1988
10	2	0	1	0	1	10NOV1987	21FEB1988
11	2	1	1	0	1	10NOV1987	21FEB1988
12	2	2	1	0	1	10NOV1987	21FEB1988
13	2	3	1	0	1	10NOV1987	21FEB1988
14	2	4	1	0	1	10NOV1987	21FEB1988
15	2	5	1	0	1	10NOV1987	21FEB1988
16	2	6	1	0	1	10NOV1987	21FEB1988
17	2	7	1	0	1	10NOV1987	21FEB1988
18	2	8	1	1	1	10NOV1987	21FEB1988
19	2	9	1	1	1	10NOV1987	21FEB1988
20	2	10	1	1	1	10NOV1987	21FEB1988
21	2	11	1	1	1	10NOV1987	21FEB1988
22	2	12	1	1	1	10NOV1987	21FEB1988
23	2	13	1	1	1	10NOV1987	21FEB1988
24	2	14	1	1	1	10NOV1987	21FEB1988
25	2	15	1	1	1	10NOV1987	21FEB1988
26	3	0	1	0	0	27JUL1987	.
27	3	1	1	1	0	27JUL1987	.

Figure 80.2 View of Expanded Monthly Data Table SAMPSIO.churn_fullyexpanded_monthly

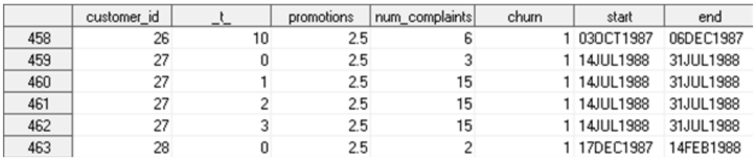


	customer_id	_t_	promotions	num_complaints	churn	start	end
1	1	0	1	0	1	20MAY1988	10JUL1988
2	1	1	1	5	1	20MAY1988	10JUL1988
3	1	2	1	10	1	20MAY1988	10JUL1988
4	2	0	1	0	1	10NOV1987	21FEB1988
5	2	1	1	0	1	10NOV1987	21FEB1988
6	2	2	1	0	1	10NOV1987	21FEB1988
7	2	3	1	1	1	10NOV1987	21FEB1988
8	3	0	1	0	0	27JUL1987	.
9	3	1	1	0	0	27JUL1987	.

Less granular data might be advantageous from a processing resources perspective, but using less granular data can result in data loss that impacts the quality of the survival model outcomes. These situations can be hard to find and prevent without taking a look at the how different fully-expanded data sets are created.

An example of such information loss can be found by browsing the data for Customer ID 27 in both of the weekly and monthly fully expanded sample data sets. By examining the weekly expanded data, we can see that Customer ID 27 opened an account mid-month, logged a net total of 48 complaints over the following two weeks, and then churned by month's end.

Figure 80.3 View of CustID 27 in Expanded Weekly Data



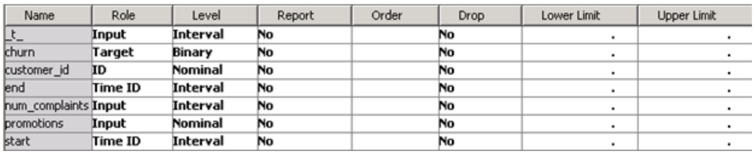
	customer_id	_t_	promotions	num_complaints	churn	start	end
458	26	10	2.5	6	1	03OCT1987	06DEC1987
459	27	0	2.5	3	1	14JUL1988	31JUL1988
460	27	1	2.5	15	1	14JUL1988	31JUL1988
461	27	2	2.5	15	1	14JUL1988	31JUL1988
462	27	3	2.5	15	1	14JUL1988	31JUL1988
463	28	0	2.5	2	1	17DEC1987	14FEB1988

Now, let us examine CustID 27 in the monthly data. CustID 27 is represented in the figure below by a single record. The time varying covariate num_complaints for CustID27 reads 0 (as opposed to a total of 48 complaints in the weekly data). This is because the monthly data table records the existing number of complaints on the first day of the month, the time when monthly data is summarized. Discrepancies such as these can negatively impact parameter estimates in the logistic survival model. You need to know the point in the time interval when your data are summarized and recorded. In this case, the monthly expanded data would reveal a more accurate summary if the data were summarized and recorded at the end of the month. Another option to avoid data compression errors like these is to use change-time data. For more information about change-time data, see “Change Time Format Data Requirements” on page 1242.

It is not hard to see that the larger discrete time scale for monthly data, as well as discrepancies between the time-varying covariates across the weekly- and monthly-summarized data is very likely to produce two significantly different survival models.

In order to see how time intervals within expanded data affects the models, we can assign the following data roles to the models:

Figure 80.4 Variable Roles for Example Expanded Data Models



Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
_t_	Input	Interval	No		No	.	.
churn	Target	Binary	No		No	.	.
customer_id	ID	Nominal	No		No	.	.
end	Time ID	Interval	No		No	.	.
num_complaints	Input	Interval	No		No	.	.
promotions	Input	Nominal	No		No	.	.
start	Time ID	Interval	No		No	.	.

Using the variable roles shown above, Survival models were built and run using the expanded weekly and monthly data. Model results were generated. Plots for Empirical Survival Function and Hazard Function and regression coefficient statistics are provided for both models below.

Figure 80.5 Empirical Survival Function Using Expanded Weekly Data

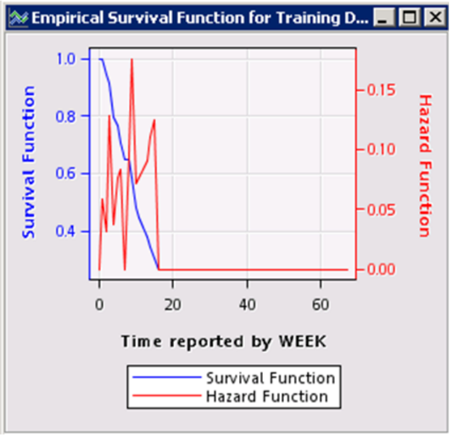
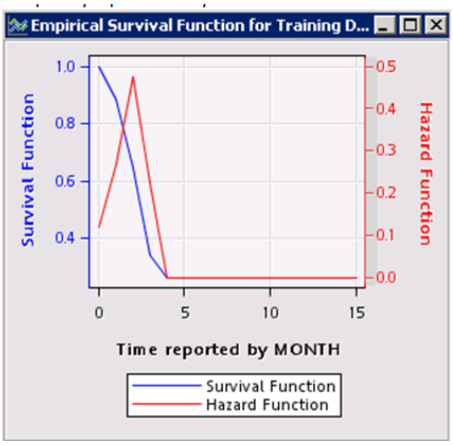


Figure 80.6 *Empirical Survival Function Using Expanded Monthly Data*



Notice the increased detail in the Empirical Hazard Function plot that was constructed from the weekly data.

Now, let us examine the regression coefficients for the weekly expanded data and the monthly expanded data models.

Figure 80.7 *Regression Coefficients for Survival Model Using Weekly Expanded Data*

Parameter:	Estimate:
Intercept	-4.9412
_t_	-90.0439
_csb1	-0.1847
_csb2	0.6972
_csb3	-1.0924
_csb4	0.8209
_csb5	-0.2434
promotions(1)	-1.9007
promotions(2.5)	-0.3250
promotions(10)	0
num_complaints	0.1471

Figure 80.8 *Regression Coefficients for Survival Model Using Monthly Expanded Data*

Parameter:	Estimate:
Intercept	-1.3461
_t_	5.6155
_csb1	2.6030
_csb2	-2.4526
_csb3	1.9465
_csb4	-1.0729

_csb5	0.2920
promotions(1)	-2.5410
promotions(2.5)	-0.3056
promotions(10)	0
num_complaints	0.3013

Note the dissimilar values for regression coefficients between the expanded data weekly and expanded data monthly models. In the case of this particular example, one could make a strong argument that the model that was created from the weekly fully-expanded data is more accurate.

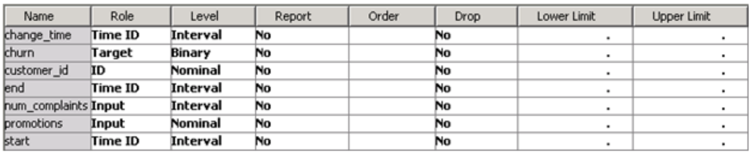
Change Time Format Data Requirements

Change-time data format can be used to avoid data compression errors as illustrated above. You can set the **Survival** node Time Interval property to calculate survival models at several discrete time scales, while concurrently specifying properties for left truncation and right censoring dates. To do so requires data manipulation with the fully-expanded format.

The SAS Enterprise Miner example SAMPSIO library provides a sample data set called SAMPSIO.churn_changetime. You can use the SAMPSIO.churn_changetime data set to visualize the extra functionality that the change-time data form provides. Using the fully-expanded SAMPSIO.churn_changetime data set, you can generate weekly and monthly survival models by simply changing the Time Interval property. Empirical results would be expected to be exactly similar to those built from the expanded data. Regression parameters would be expected to differ only slightly, as the node uses a weighting scheme to expand change-time data before the modeling procedures are invoked.

Variable roles must be appropriately specified for change-time data. The following figure provides suggested variable role settings.

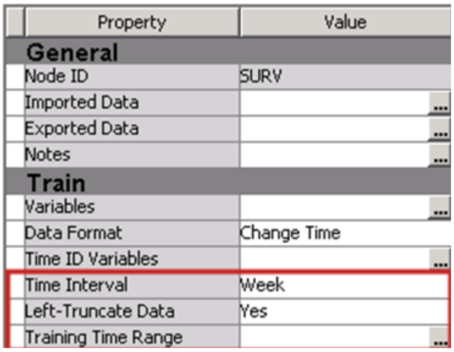
Figure 80.9 Variable Roles for Example Change-Time Data Survival Model



Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
change_time	Time ID	Interval	No		No	.	.
churn	Target	Binary	No		No	.	.
customer_id	ID	Nominal	No		No	.	.
end	Time ID	Interval	No		No	.	.
num_complaints	Input	Interval	No		No	.	.
promotions	Input	Nominal	No		No	.	.
start	Time ID	Interval	No		No	.	.

Time interval, truncation settings, and training time range must also be specified for time-change data modeling. You use these property settings specify the time frame and time interval that the model will use without having to manipulate the input data.

Figure 80.10 *Properties to Specify Time Frame and Time Interval for Change-Time Data Modelers*



Property	Value
General	
Node ID	SURV
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Data Format	Change Time
Time ID Variables	...
Time Interval	Week
Left-Truncate Data	Yes
Training Time Range	...

Survival Node Properties

Survival Node General Properties

The following general properties are associated with the **Survival** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Survival node that is added to a diagram will have a Node ID of Survival. The second Survival node added to a diagram will have a Node ID of Survival2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Survival window. The Imported Data — Survival window contains a list of the ports that provide data sources to the **Survival** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Survival window. The Exported Data — Survival window contains a list of the output data ports that the **Survival** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Survival Node Train Properties

The following train properties are associated with the **Survival** node:

- **Variables** — Use the Variables window to view variable information, and change variable values using the **Survival** node. Select the ... button to open a window that contains the variables table. You can specify *Use* and *Report* values for a variable. You can view the columns metadata. You can also open an Explore window to view a variable's sampling information, observation values, or a variable distribution plot. By default, columns for the variables **Name**, **Use**, **Report**, **Role**, and **Level** are displayed.

  To modify these values, and add additional columns, you can use the following options:

  - **Apply** — Changes metadata based on the values that are supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the **Apply** button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

- **Data Format** — Specifies the data format that is used. Use **Standard** when you do not need to accommodate time-dependant covariates. Use **Change Time** and **Fully-Expanded** to enable multiple observations per time ID, which accommodates time-dependent covariates. **Fully-Expanded** requires a _t_ variable with a row for each time interval. **Change Time** requires a row only for each time interval where a covariate changes value. An additional time ID variable contains the time when this happens.

  Note: If you specify **Standard** or **Change-Time** as the setting for your **Data Format** property, if your data set contains a variable named _t_ with role=Input, the _t_ variable will not be used.

  If you specify either the **Change Time** or **Fully-Expanded** data format, the exported data contains only columns from the score code for the last observation for each time ID. That is, every observation except the last observation contains missing values for any variables added to the import data set.

  Note: If you want to partition your data and use the **Change Time** or **Fully-Expanded** data format, you must specify certain settings in the **Data Partition** node. The time ID variable must be used as a cluster variable and the **Partitioning Method** must be set to **Cluster**.

- **Time ID Variables** — The time ID variables identify the variables that will indicate the start and end times for the discrete time span until the event of interest. You assign start and end time variables via the mapping table that appears when you select the ellipsis button ... to the right of the Time ID Variables property in the

Survival node properties panel. The **Survival** node scans the TIMEID column's meta-variable values to determine maximum and minimum TIMEID values in the table. The variable that has the earliest time unit is mapped to **Start Time Variable**, and the variable that has the latest time unit is mapped to **End Time Variable**.

If you specify **Change Time** as the **Data Format**, then the **Change Time Variable** selection becomes available.

- **Time Interval** — The time interval variable specifies the unit of time to be used for measurement and prediction. The results of the analysis are expressed in terms of the chosen time scale. They should represent the smallest units that are measurable and actionable. Typically, the values represent days, weeks, months, billing cycles, quarters, or years.

  The smallest time interval found in most databases is a day. Many customers initiate, cancel, or change their products and services on the same day. After you specify the time interval in the **Survival** node properties panel, SAS Enterprise Miner performs error checking to ensure that your start time date format supports the corresponding time interval.

- **Left-Truncate Data** — Specifies whether the **Training Time Range** property enables you to specify a truncation date. All data that is observed before the truncation date is omitted from analysis.

- **Training Time Range** — Opens the Train Date Selection window that enables you specify the **Left Truncation** date and the **Right Censoring** date. Data outside of this range is omitted from analysis.

Survival Node Train Properties: Sampling Options

- **Sampling** — Specifies whether sampling of the expanded data should occur. If it should, simple random sampling of the non-events only is performed. All events remain in the table to be modeled.

- **Event Proportion** — Specifies the event proportion in the sample. Event proportion values are real numbers ranging between 0.0 and 0.9. The default setting is 0.2.

- **Seed** — Specifies the seed number to be used during sample selection. The default sampling seed value is *12345*.

Survival Node Train Properties: Regression Spline Model

- **Covariate x Time Interactions** — Specifies the variables that interact with the _t_ variable in the regression model. Select **Do not include** to suppress all interactions, **Include All** to enable interactions with all variables, and **Include Selected** to manually specify the interactions. You can manually specify the variables with the **New Covariates for Interactions** property.

- **New Covariates for Interactions** — Opens a window that enables you to specify that variables that interact with the _t_ variable in the regression model.

- **Stepwise Regression** — Specifies whether stepwise regression should be used for variable selection during the modeling stage.

- **Entry Significance Level** — Specifies the significance level for adding variables in stepwise regression.

- **Stay Significance Level** — Specifies the significance level for removing variables in a stepwise regression.

- **Number of Knots** — Specifies the number of knots used in the cubic spline basis function.

- **Knot Selection** — Specifies whether to use automatic knot selection for the cubic spline basis functions when stepwise selection is being performed. When set to **Yes**, the cubic spline functions that are created are entered into the regression model and are part of the stepwise variable selection procedure. This enables statistical determination of whether a cubic spline basis function is significant or not. When **Knot Selection** is set to **Yes**, the **Number of Knots** property specifies the number of knots to be created for consideration as part of variable selection.

Survival Node Train Properties: Survival Validation

The Survival Validation section of the **Survival** node properties panel provides the following settings for configuring the validation of your Survival model:

- **Survival Validation Method** — Use the **Survival Validation Method** property to specify how the validation holdout sample is generated. By default, subsets are created from each of the training, validation, and test data sets that are passed to the **Survival** node, and those subsets are used for model validation. When the default survival validation method is selected, the last 25% of the total time interval is set aside for survival model validation and scoring. The scoring interval is used for validation calculations.

  If you desire, you can choose to create a user-specified scoring interval. When you select the user-specified validation method, you can specify a specific hypothetical scoring date to define the beginning of the scoring interval as well as the scoring interval length.

  All Survival model validation is performed with the model that was built using the entire Train data set.

- **Validation Score Date** — When you select **User-Specified** as the value for your **Survival Validation Method** setting, you can use the **Validation Score Date** property to specify a hypothetical scoring date. The hypothetical scoring date defines the beginning of the scoring interval that is used to subset the data for model evaluation. Select the ... button to the right of the **Validation Score Date** property to open a table in which you specify the date value for scoring. The **Validation Score Date** date value that you specify must fall between your defined start date and censor date.

- **Interval Length** — When you select *User Specified* as the value for your **Survival Validation Method**, the **Interval Length** property specifies the length of the time interval that is used to perform survival validation. The interval will begin with the hypothetical scoring date that you specified in the **Validation Score Date** property setting, and then extend the specified number of time intervals forward. If the **Time Unit** property is set to *Day* and the **Interval Length** property is set to *15*, your scoring interval is the 15 days that follow the hypothetical scoring date.

Survival Node Score Properties

The Score section of the **Survival** node properties panel provides the following settings for configuring your Survival model scoring:

- **Mean Residual Life (MRL)** — This property specifies whether Mean Residual Life (MRL) should be calculated. MRL represents the remaining time until the event occurs. MRL can be processor-intensive to calculate, so the default setting for this property is NONE, which suppresses the MRL calculation. If you select the Constant Hazard Extrapolation setting, you assume that from time *t* onward, the hazard function is constant from the final value. If you select the Restricted Mean Residual Life setting, the hazard function continues trending until an event occurs, or until the maximum value for MRL is reached, whichever comes first. After the maximum value for MRL is reached, the hazard is held constant from that point forward.

- **Default Maximum (MRL)** — This property is used in conjunction with Restricted Mean Residual Life. It specifies whether default values that are based on the specified Time Interval will be used to set the maximum MRL used in calculations. When **Default Maximum MRL** is set to Yes, the following default values will be used based on Time Interval: Day=108, Week=108, Month=60, Quarter=40, Semi-Year=50, and Year=50.
- **User-Specified Maximum MRL** — If the **Default Maximum MRL** property is set to *No*, the **User-Specified Maximum MRL** property enables you to specify the maximum MRL value. The value that you specify is used to calculate Restricted Mean Residual Life.
- **Default Forecast Intervals** — The **Default Forecast Intervals** property specifies the number of time units into the future that you want to use for score code generation, survival functions, and so on. The **Default Forecast Intervals** property indicates whether default values based on the Time Unit should be selected, or whether user-specified values will be used. If **Default Forecast Intervals** is set to Yes, the following time unit values will be used: Day=30, Week=4, Month=3, Quarter=4, Semi-Year=2, and Year=1.
- **Number of Forecasted Intervals** — If the **Default Forecast Intervals** property is set to *No*, the **Number of Forecasted Intervals** property specifies the number of time intervals into the future to use as the basis for your survival calculations.

Survival Node Report Properties

The Report section of the **Survival** node properties panel provides the following settings for configuring your Survival model reports:

- **Risk Account Selection** — This reporting property specifies whether the table should be generated based on percentile or count.
- **High Risk Account Tables** — Specifies whether High Risk tables should be generated and, if so, which tables to create. If **None** is selected, no additional tables are created. If **Event Probability** is selected, those IDs with the highest event occurrence probability within the forecast time period given survival until the current time are identified and displayed in the results. If **Survival Probability** is selected, those IDs with the lowest survival probability at the forecasted time are identified and displayed in the results. If **Hazard Rate** is selected, those IDs with the highest overall hazard rate at the forecasted time are identified and displayed in the results. If **All** is selected, all three tables are generated and displayed.
- **Percentile** — The **Percentile** reporting property specifies the corresponding percentage value to be used in creating the table.
- **Count** — The **Count** reporting property specifies the corresponding count value to be used in creating the table.

Survival Node Status Properties

The following status properties are associated with the **Survival** node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.

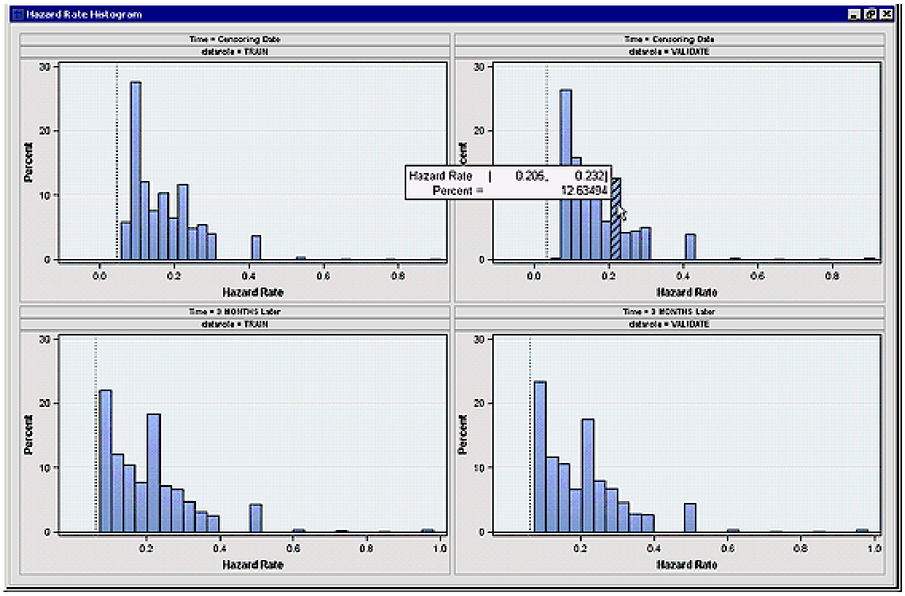
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Survival Node Results

The **Survival** node Results browser contains the following output plots and data tables:

- **Hazard Rate Histogram**
- **Event Occurrence Probability Histogram**
- **Survival Probability Histogram**
- **Empirical Subhazard Function for Training Data**
- **Empirical Survival Function for Training Data**
- **Model Validation Plot**
- **Model Validation Statistics**
- **SAS Output**

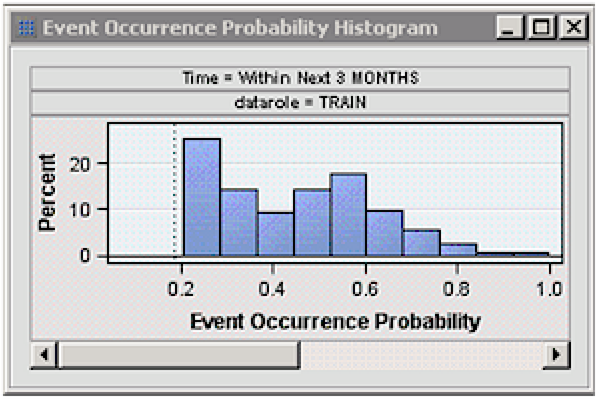
Hazard Rate Histogram (Time=Censoring Date and Time=3 Time Units Later):



The Hazard Rate histogram displays the plots of the hazard rate distribution for the time interval (for example, month) that contains the censoring date. If **f** is the number of forecast intervals specified, then the hazard rate is also plotted for the **fth** interval that follows the censor date. For example, for an analysis that uses month as the time interval, and that uses the default setting of three forecast intervals (**f** = 3), the Hazard Rate histogram displays the hazard rate through the third month that follows the censor date. Each bar in the histogram represents the percentage of hazard rates that are between the beginning and ending bins (where the bins are the hazard rates).

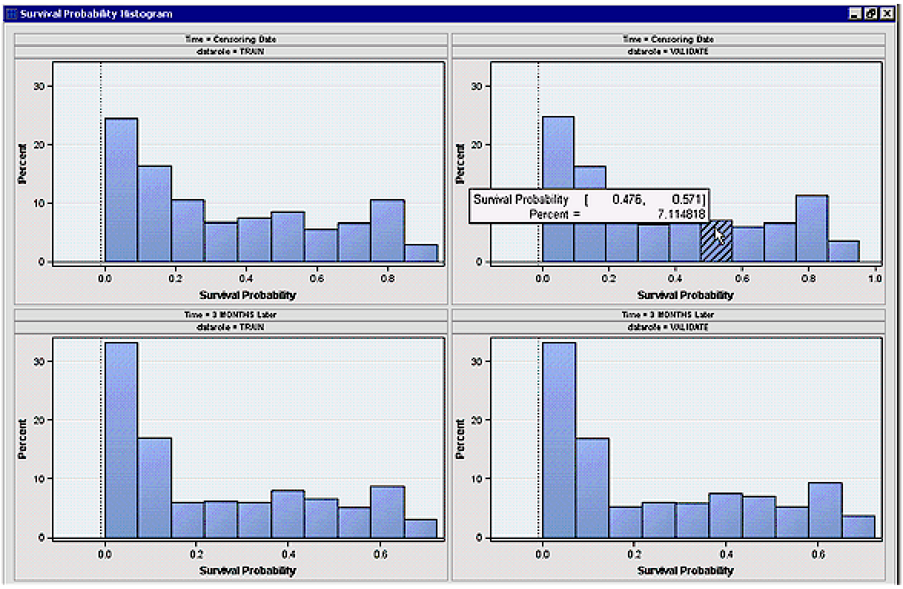
The hazard rate tells you the chance that someone is going to cancel or stop service. You can use the histogram that extrapolates three time units into the future to identify customers that have high hazard probabilities. You might intervene with high hazard probability customers by offering them some type of promotion.

Event Occurrence Probability Histogram (within next 3 time units):


Event Occurrence Probability Histogram
Time = Within Next 3 MONTHS
datarole = TRAIN
Percent
20
10
0
0.2
0.4
0.6
0.8
1.0
Event Occurrence Probability

The Event Occurrence Probability histogram displays the distribution of the probabilities of having an event of interest occur within the next **f** forecast time intervals, where **f** is the number of forecast intervals that the **Survival** node is configured to analyze (default setting is **f** = 3). You specify the time units for each forecast interval in the **Survival** node properties panel when you configure the node. Each bar in the plot represents the percentage of customers who will experience an event of interest during the next **f** forecast intervals.

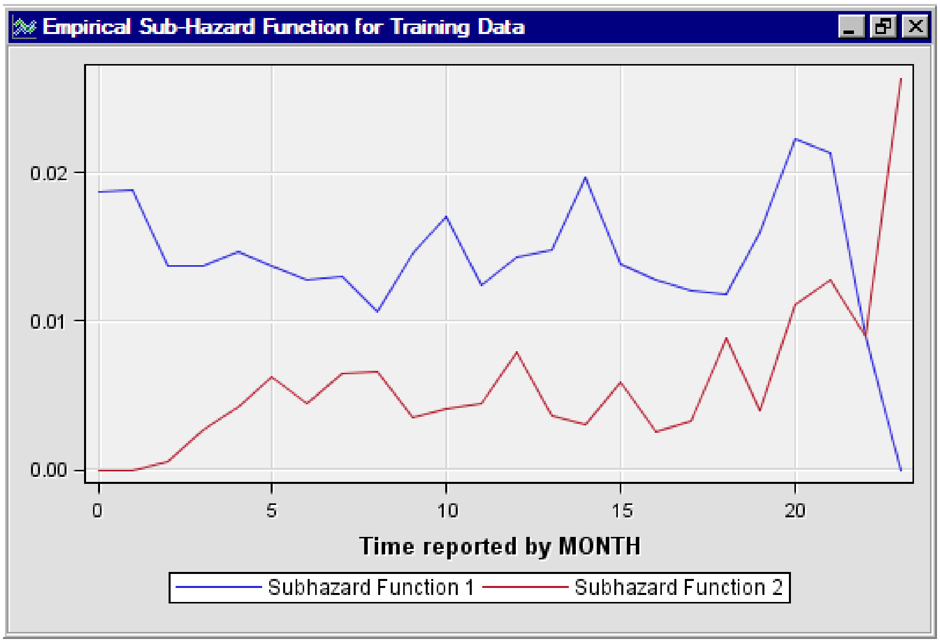
Survival Probability Histogram (Time=Censoring Date and Time=3 Time Units Later):



The Survival Probability Histogram displays the distribution of the survival probabilities for the time interval (for example, month) that contains the censoring date. If **f** is the number of forecast intervals that are specified, then the Survival Probability histogram is also plotted for the **fth** interval that follows the censor date. For example, for an analysis that uses month as the time interval, and that uses the default setting of three forecast intervals (**f** = 3), the Survival Probability histogram displays the survival probability through the third month that follows the censor date. Each bar in the histogram represents the survival probabilities that are between the beginning and ending bins (where the bins are the hazard rates)

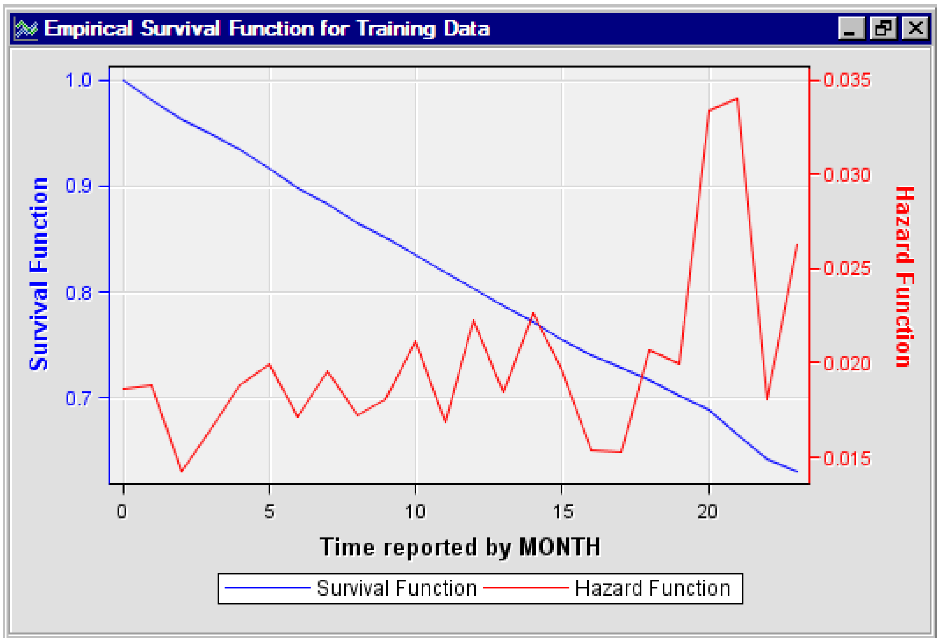
You could use the histogram to determine which customers would be suitable for a loyalty promotion by targeting customers that have high survival probabilities. The Survival Probability histogram for three time units later displays the probabilities that a customer account will remain active during the three-month interval that follows the censoring date.

Empirical Subhazard Function for Training Data:



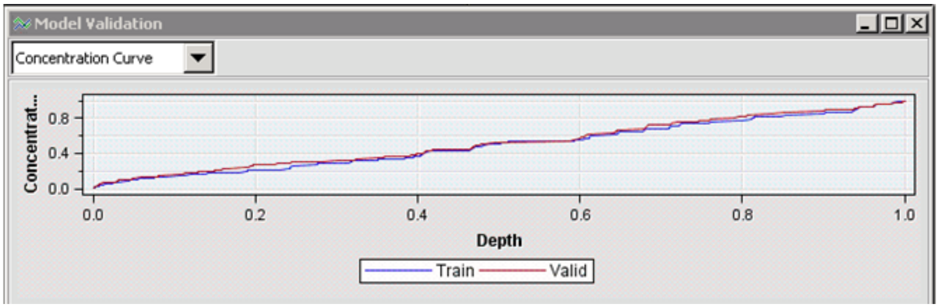
For each one of the competing risks, the subhazard function is computed and displayed in this graph. The vertical axis is the values of the subhazard functions, and the horizontal axis is time.

Empirical Survival Function for Training Data:



This plot overlays the Hazard function and the Survival function across the entire time continuum. The left vertical axis represents the Survival function, and the right vertical axis represents the Hazard function. The horizontal axis is the time unit.

Model Validation Plot:



The three Model Validation Plots are as follows:

- **Concentration Curve**:

  In a concentration curve, the concentration is plotted on the vertical axis. The horizontal axis is the depth (deciles) of the data. For each decile, the concentration is computed. The concentration is the ratio of the number of events in the decile to the total number of events. The concentration curve is similar to a Cumulative Captured Response curve from the **Model Comparison** node.

- **Lift**:

  The lift chart is a data mining measure that is used to choose among competing models. Assume a model that predicts the probability of some event, such that a higher calculated probability value implies a higher likelihood of an event.

  Consider submitting a data set to the model to score the data, and then rank the scored data by descending posterior probability values. In a model with good discriminatory performance, the cases that are predicted to be events should be at the top of the sorted list. The cases that are predicted to be non-events should be at the bottom of the list.

- **Benefit**:

  The Benefit is plotted on the vertical axis. The horizontal axis is the depth (deciles) of the data. The Benefit statistics represent the difference between the Concentration Curve and the random model (represented by a 45–degree diagonal line). The largest Benefit value indicates the depth at which the model is doing the best job at predicting the outcome. The Benefit curve can be used to establish an appropriate cutoff value.

Model Validation Statistics:

The Model Validation Statistics table displays the following statistics:

Benefit
the maximum benefit value

Average Hazard Ratio
the average hazard ratio that gives the maximum benefit value

Depth
the depth at the maximum benefit value

Lift
the lift at the maximum benefit value

Kolmogorov-Smirnov statistic
the maximum distance between the event and non-event distributions

Gini Concentration Ratio
twice the area between the concentration curve and the random model (represented by a 45–degree diagonal line). The Gini concentration ratio statistic is a measure of the separation between the probabilities of event and non-event..

SAS Output
The output window in the Survival Results browser contains the SAS output from running the LIFETEST and DMREG procedures.

Survival Node Example

Overview

The **Survival** node usage example proceeds sequentially through the following steps:

1. Create and Configure Survival Data Source on page 1252
2. Place Survival Analysis Nodes in the Diagram Workspace on page 1254
3. Configure the Survival Node Settings on page 1254
4. Run the Survival Node on page 1254
5. Examine Survival Node Results on page 1254
6. Submit a Data Set for Scoring on page 1260

Create and Configure Survival Data Source

This example uses the SAS sample data set SAMPSIO.CELLPHONE. The SAMPSIO.CELLPHONE data set is found in the SAMPSIO example data library that is included with your copy of SAS Enterprise Miner.

Note: You also use the SAMPSIO.CELLPHONE_SCORE data set from the SAMPSIO library in the last step of this example to use your trained model to score data.

Use the SAS Enterprise Miner Data Source Wizard to create a SAS Enterprise Miner Data Source from the example SAMPSIO.CELLPHONE data set. In the SAS Enterprise Miner Project Panel, right-click the **Data Sources** folder and select **Create Data Source** to launch the Data Source Wizard.

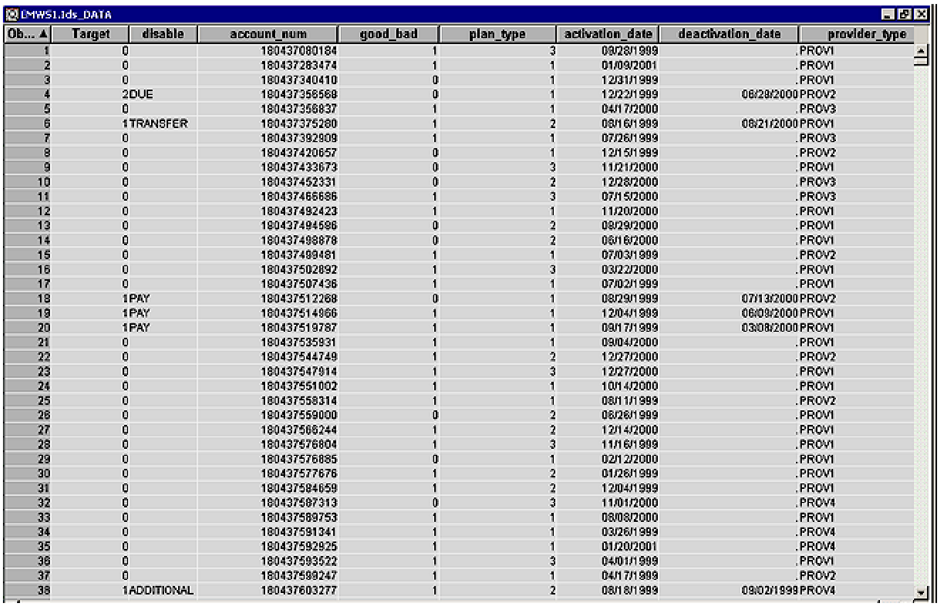
In the Data Source Wizard, do the following to create your SAS Enterprise Miner Data Source:

1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter SAMPSIO.CELLPHONE in the **Table** field and click **Next**.
3. Continue to the Metadata Advisor and choose the **Advanced Metadata Advisor**.
4. In the Column Metadata window, make the following variable role assignments:
   - Set the role of the variable ACCOUNT_NUM to **ID**.
   - Set the roles of the variables ACTIVATION_DATE and DEACTIVATION_DATE to **TimeID**.
   - Set the role of the variable DISABLE to **Rejected**.
   - Set the role of the variable TARGET to **Target**.

- All of the remaining variables (GOOD_BAD, PLAN_TYPE, and PROVIDER_TYPE), should be set to the role of **Input**.
- Click **Next**.

5. Accept the default settings for the remainder of the Data Source Wizard windows. Select **Next** for each window, and when you reach the last Data Source Wizard window, select **Finish**.
6. The SAMPSIO.CELLPHONE data set should appear in your Data Sources folder in the SAS Enterprise Miner Project Panel.

The SAMPSIO.CELLPHONE data source that you just created should have 10,185 observations. Each observation is a unique customer observation from a cell phone provider company. The data source that you created contains the cell phone data as follows:



Ob... ▲	Target	disable	account_num	good_bad	plan_type	activation_date	deactivation_date	provider_type
1	0		180437080184	1	3	08/28/1999	.	PROV1
2	0		180437283474	1	1	01/09/2001	.	PROV1
3	0		180437340410	0	1	12/31/1999	.	PROV1
4	2	DUE	180437358568	0	1	12/22/1999	06/28/2000	PROV2
5	0		180437358837	1	1	04/17/2000	.	PROV3
6	1	TRANSFER	180437375280	1	2	08/16/1999	08/21/2000	PROV1
7	0		180437392909	1	1	07/26/1999	.	PROV3
8	0		180437420657	0	1	12/15/1999	.	PROV2
9	0		180437433673	0	3	11/21/2000	.	PROV1
10	0		180437452331	0	2	12/28/2000	.	PROV3
11	0		180437466686	1	3	07/15/2000	.	PROV3
12	0		180437492423	1	1	11/20/2000	.	PROV1
13	0		180437494596	0	2	08/29/2000	.	PROV1
14	0		180437498878	0	2	06/16/2000	.	PROV1
15	0		180437499481	1	1	07/03/1999	.	PROV2
16	0		180437502892	1	3	03/22/2000	.	PROV1
17	0		180437507436	1	1	07/02/1999	.	PROV1
18	1	PAY	180437512268	0	1	08/29/1999	07/13/2000	PROV2
19	1	PAY	180437514966	1	1	12/04/1999	06/09/2000	PROV1
20	1	PAY	180437519787	1	1	09/17/1999	03/08/2000	PROV1
21	0		180437535931	1	1	09/04/2000	.	PROV1
22	0		180437544749	1	2	12/27/2000	.	PROV2
23	0		180437547914	1	3	12/27/2000	.	PROV1
24	0		180437551002	1	1	10/14/2000	.	PROV1
25	0		180437558314	1	1	08/11/1999	.	PROV2
26	0		180437559000	0	2	08/26/1999	.	PROV1
27	0		180437566244	1	2	12/14/2000	.	PROV1
28	0		180437576804	1	3	11/16/1999	.	PROV1
29	0		180437576885	0	1	02/12/2000	.	PROV1
30	0		180437577676	1	2	01/26/1999	.	PROV1
31	0		180437584659	1	2	12/04/1999	.	PROV1
32	0		180437587313	0	3	11/01/2000	.	PROV4
33	0		180437589753	1	1	08/08/2000	.	PROV1
34	0		180437591341	1	1	03/26/1999	.	PROV4
35	0		180437592925	1	1	01/20/2001	.	PROV4
36	0		180437593522	1	3	04/01/1999	.	PROV1
37	0		180437599247	1	1	04/17/1999	.	PROV2
38	1	ADDITIONAL	180437603277	1	2	08/18/1999	09/02/1999	PROV4

In this data set, TARGET is the target variable that is modeled. TARGET indicates what type of risk is being modeled. There are three risks that are being modeled:

- If TARGET=0, the customer still has an active account
- If TARGET=1, the customer has voluntarily ended the relationship with cell phone company
- If TARGET=2, the customer has involuntarily ended the relationship with the cell phone company. For example, a customer might be terminated for delinquent payments for services.

The TARGET variable value is computed as follows:

If TARGET=(DEACTIVATION_DATE ne .) and DISABLE=DUE, then the DEACTIVATION_DATE variable contains a date value, and the class variable DISABLE is set to DUE. On the date in DEACTIVATION_DATE, the customer relationship was ended. The class value for the DISABLE variable provides the reason that the customer relationship ended.

If the customer has a DEACTIVATION_DATE value and the customer was deactivated because account payments were past due, that would be an involuntary churn event (TARGET=2).

If a customer has a DEACTIVATION_DATE value and the DISABLE variable is anything other than DUE, then the customer relationship was ended by the customer, resulting in voluntary churn (TARGET=1).

If the DEACTIVATION_DATE variable is a missing value, and the DISABLE variable is a missing value, then the customer is still active (TARGET=0).

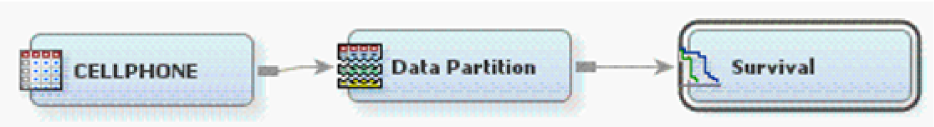
The data set also contains other explanatory variables, such as a good or bad credit indicator (GOOD_BAD), the customer's type of rate plan (PLAN_TYPE) and the customer's provider plan (PROVIDER_TYPE).

Place Survival Analysis Nodes in the Diagram Workspace

Drag the CELLPHONE data source that you just created from the Data Sources folder in the Project Panel onto a blank Diagram Workspace.

Next, drag a **Data Partition** node from **Sample** tab of the Nodes Toolbar onto the Diagram Workspace. Connect the CELLPHONE data source to the **Data Partition** node. Select the **Data Partition** node. Set the **Training** property to `70.0` the **Validation** property to `30.0`, and the **Test** property to `0.0`.

Drag a **Survival** node from the **Applications** tab of the Nodes Toolbar onto the Workspace and connect the **Data Partition** node to the **Survival** node.



Configure the Survival Node Settings

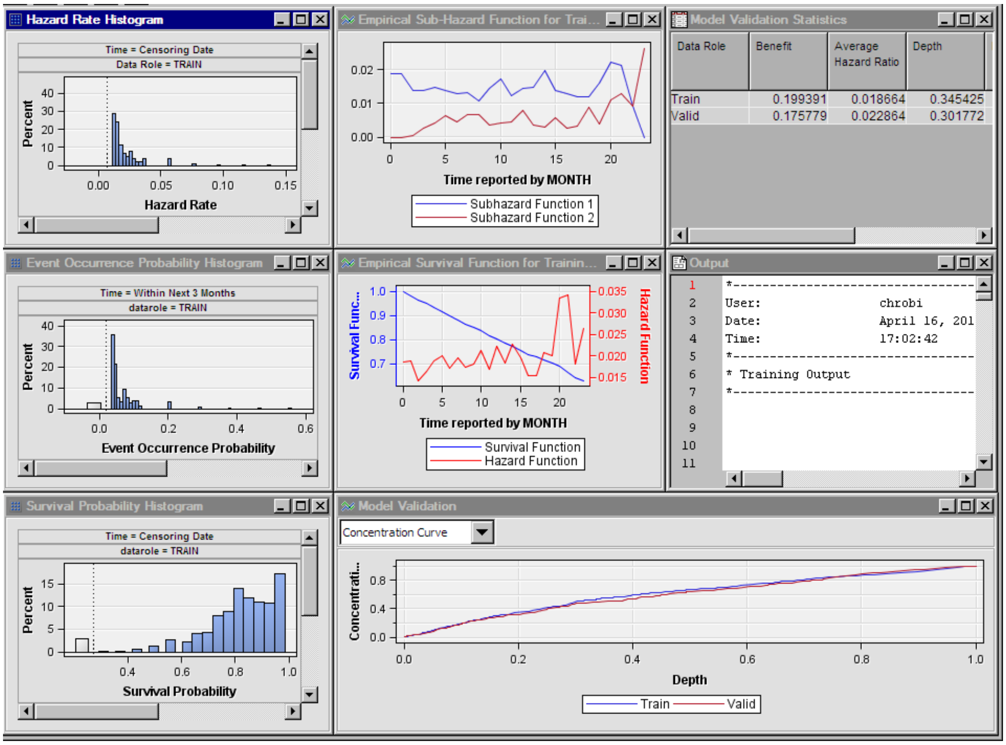
Select the Survival node in the Diagram Workspace, and then use the Properties Panel to set the **Survival** node **Stepwise Regression** property to *Yes*.

Run the Survival Node

Right-click the Survival node in the Diagram Workspace and select **Run**. After the Process Flow Diagram finishes running, click **Yes** to open the Results browser.

Examine Survival Node Results

The **Survival** node Results window shows a number or plots and tables. They include histograms of the hazard rates, event occurrence probabilities, and survival probabilities. There are overlay plots of the empirical sub-hazards and the survival and hazard functions. There is also a line plot of the model validation statistics, a table of model validation statistics, as well as the SAS output.

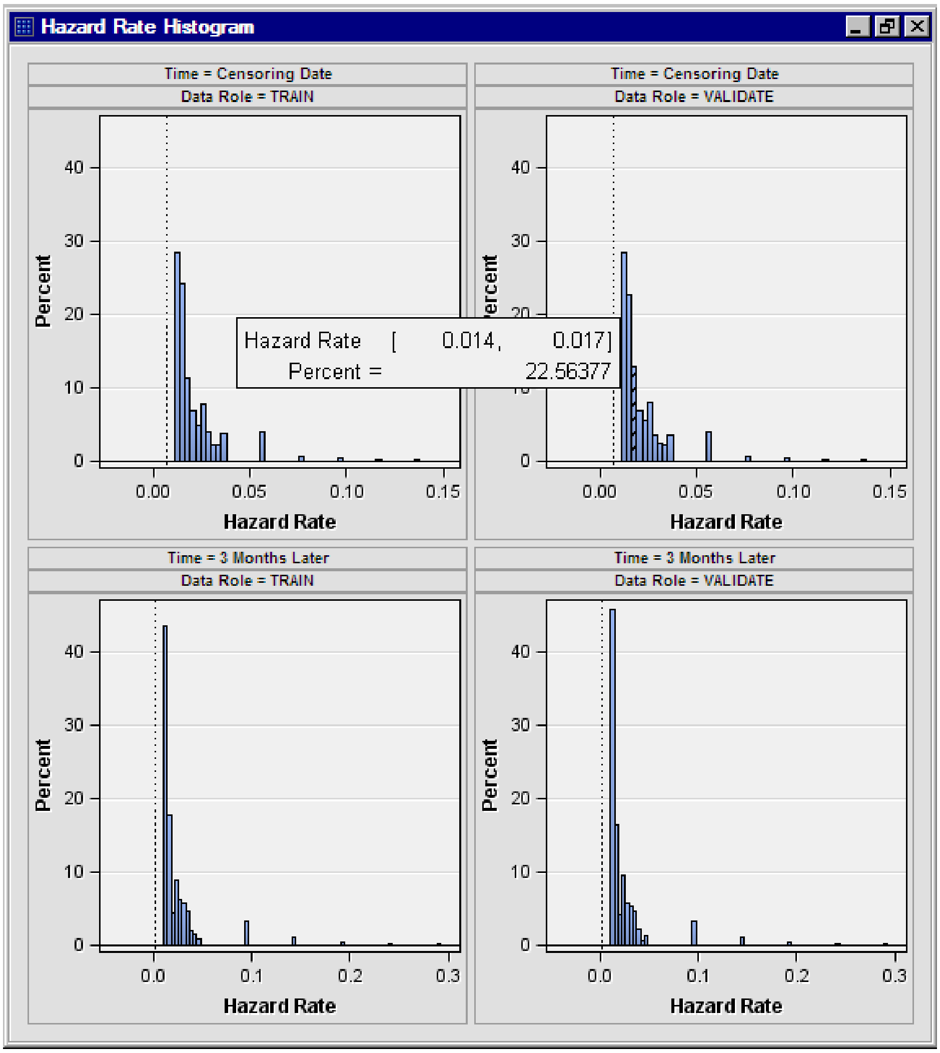


Data Role	Benefit	Average Hazard Ratio	Depth
Train	0.199391	0.018664	0.345425
Valid	0.175779	0.022864	0.301772

```
*-----------------------------------
User:                     chrobi
Date:                     April 16, 201
Time:                     17:02:42
*-----------------------------------
* Training Output
*-----------------------------------
```

Hazard Rate histogram:

The Hazard Rate Histogram displays the distribution of the hazard rate for time unit that contains the censoring date, as well as the third time interval that follows the censoring date.

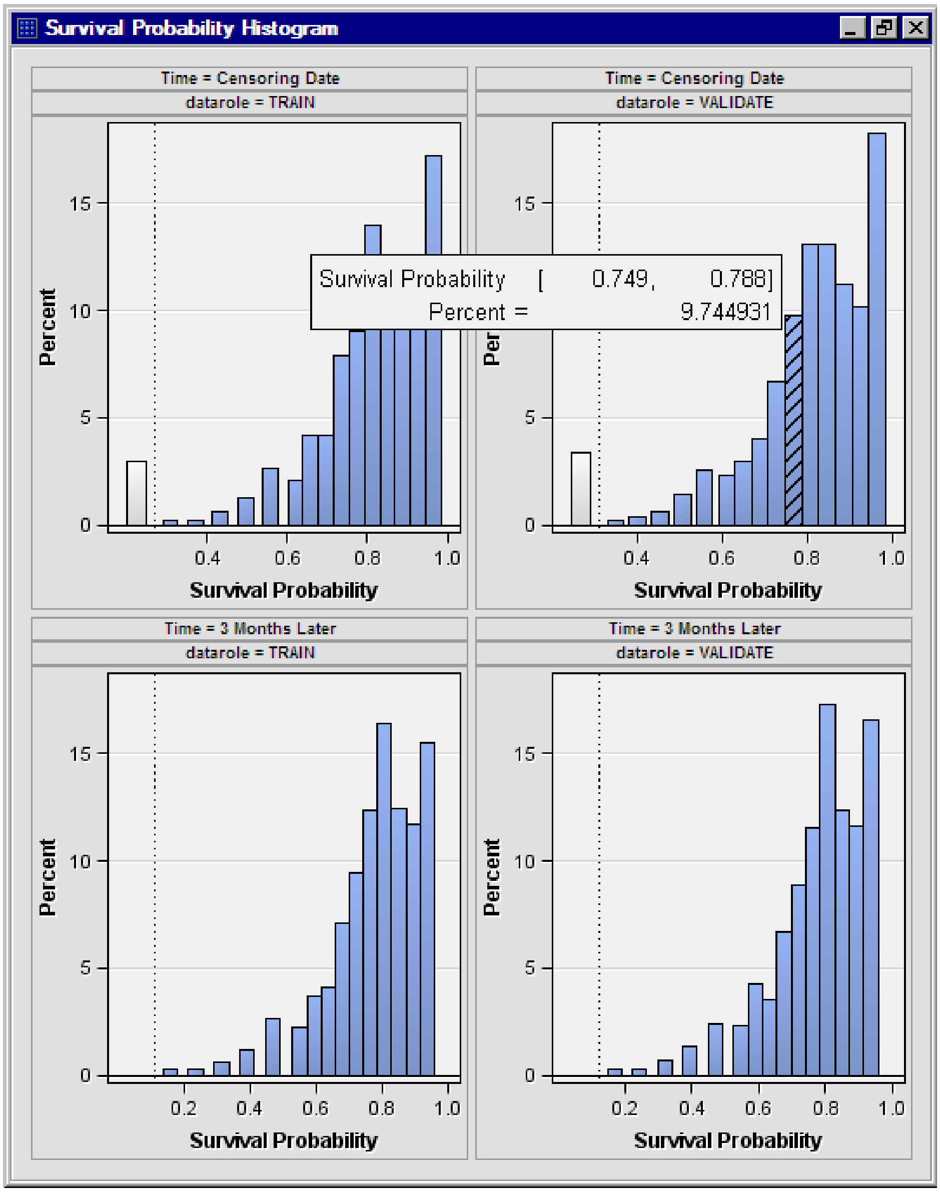


```
Hazard Rate    [      0.014,       0.017]
     Percent =                  22.56377
```

For example, the highlighted bar tells you that 22.6% of the customers in the validation data set on the censoring date will have a hazard probability between 0.014 and 0.017. The hazard probability indicates the chance that someone is going to cancel or stop service. You can use the histogram that is extrapolated for three time units into the future to identify customers that have high hazard probabilities. You might want to consider intervening with high hazard customers by offering them some sort of promotion or retention incentive.

Survival Probability Histogram:

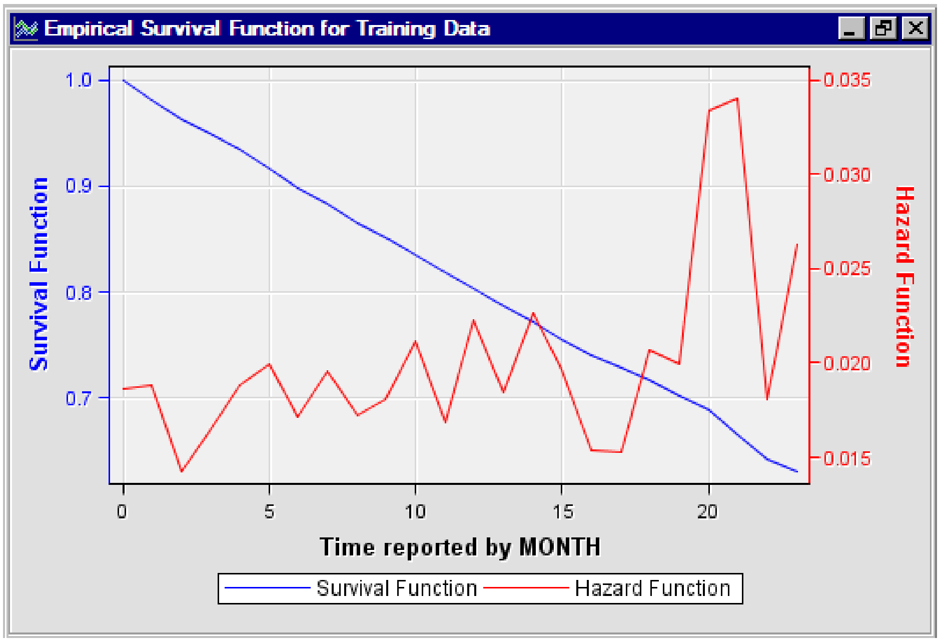
The Survival Probability histogram displays the distribution of the survival probabilities for the censoring time and for three time units into the future. For example, the highlighted bar tells you that 9.7% of the customers in the validation data set will have a survival probability between 0.749 and 0.788.



You can use the Survival Probability histogram for 3 Months later to determine which customers might be suitable for a loyalty promotion. The Survival Probability histogram for three months later displays the probabilities that a customer will still be a customer three months after the censor date. You can use this information to identify the customers that have high survival probabilities.

Empirical Survival Function for Training Data:

The Empirical Survival Function for Training Data plot overlays the hazard and survival functions.



The right horizontal axis on the Empirical Survival Function for Training Data plot represents the tenure of customers measured in months. The hazard function highlights different important events in the customer's lifecycle.

The very first hazard probability at time zero is 1.86%. This might represent customers that do not start the service right away, or customers that experience buyer's remorse.

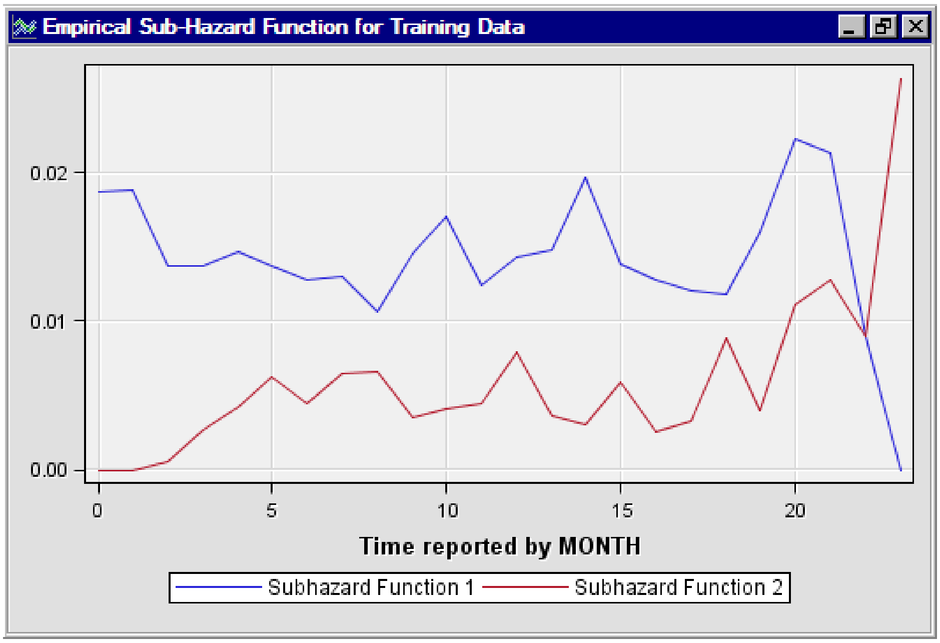
The first peak in the hazard function occurs in month 5. The peak might be a result of the cell phone company trying several customer payment incentives that did not work. Eventually, the cell phone service provider must force some churn due to non-payment.

The peaks that are shown in the hazard probability plot around months 10, 12, 14, and 20 might be due to the end of promotional periods. Customers who sign up for a service because of a highly discounted initial offer often discontinue service when the initial discount expires. On the bright side, the customers who tend to discontinue service when promotions expire are typically customers with no delinquent payments.

If hazard plots (in red) provide a snapshot of the customer lifecycle, then survival plots (in blue) provide a more complete picture. The survival probability at time *t* is the likelihood that a customer will survive to that point in time. The left horizontal axis represents the survival likelihood of the customers. Notice that the curve starts at 100% (because at *t*=0, all customers are still active) and gradually declines to 0. The survival values will always be between 0% and 100%. The survival probability curve can be used in conjunction with the hazard function. When a spike is observed in the hazard plot, a steep decline can be expected in the survival function, indicating that customers are not surviving beyond this point. In general, the smaller the hazards are, the flatter the survival curve is. When hazards are large, the survival slope is usually steeper.

Empirical Sub-Hazard Function:

The Empirical Sub-Hazard Function plot shows the subhazard functions for voluntary (Sub-Hazard Function 1 in blue) and involuntary churn (Sub-Hazard Function 2 in red).

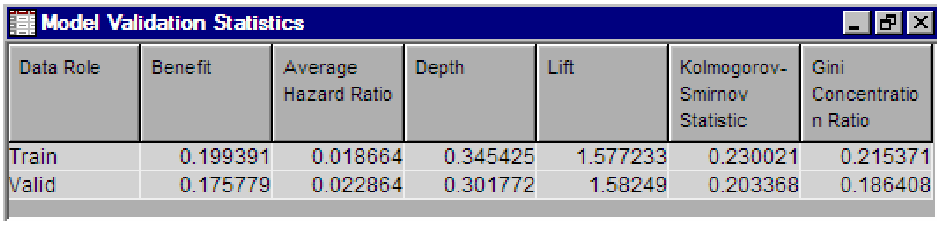


This plot graphically describes competing risks in survival data mining. A good example of competing risks is the distinction between voluntary and involuntary churn. Some customers are forced to leave (typically due to non-payment) whereas others leave voluntarily. When modeling churn, sometimes models are built that ignore one or the other group of customers. Such practices could bias the model.

In the case of competing risks, there is a separate probability for each risk. After a customer experiences an at-risk event (such as voluntary churn), that customer is excluded from the remaining at-risk events. The plot above shows the competing risks for voluntary and involuntary churn. The top blue line shows that customers who succumb to voluntary churn are at greater risk.

Model Validation Statistics:

The Model Validation Statistics table shows at what depth the best benefit, lift, K-S statistic, and Gini concentration ratio statistics can be found.



Model Validation Statistics

Data Role	Benefit	Average Hazard Ratio	Depth	Lift	Kolmogorov-Smirnov Statistic	Gini Concentratio n Ratio
Train	0.199391	0.018664	0.345425	1.577233	0.230021	0.215371
Valid	0.175779	0.022864	0.301772	1.58249	0.203368	0.186408

For example, in the validation data set, at a depth of 0.30 (about the top 30% of the data), the best lift value occurs.

The benefit value of 0.176 in the validation data set indicates this is the depth of the predicted probabilities at which the model best differentiates between the customers who experience a churn event, and customers who do not.

The K-S statistic and the Gini concentration ratio are both measures of how well the model is separating between the probabilities of churn and no churn. The K-S statistic measures the maximum vertical distance between the curves that represent the cumulative distributions of customers who experience a churn event and customers who do not churn.

Submit a Data Set for Scoring

After you build a good predictive model with the **Survival** node, you are ready to score a data set. The data set that contains the score data for this example is SAMPSIO.CELLPHONE_SCORE. Like the training data SAMPSIO.CELLPHONE, the score data set is included in the SAMPSIO sample data library that is included with your copy of SAS Enterprise Miner.

Before you can score a data set using score code from the **Survival** node, you must create a _t_ variable. The _*t*_ variable has already been created for the SAMPSIO.CELLPHONE_SCORE data set. When scoring data, the _*t*_ variable represents the number of time intervals in a given time span. In our example, the _*t*_ variable represents the elapsed time between the activation and the censoring date. If you look at the **Survival** node Results browser and view the flow code that was generated during Survival model training, you see the following:

```
format _currentdate MMDDYY10.0 ;
_currentdate=input("31DEC2000",anydtdte10.);
if (activation_date> _currentdate or deactivation_date< _currentdate)
  and deactivation_date^=.
  then _WARN_ ='U';
_T_=intck("MONTH",activation_date, _currentdate);
if _T_ ne . then do;
  ... <scoring code> ...
```

The **`if _T_ ne . then do;`** code statement indicates that if the _*t*_ variable is not present in the score data, then the score values for sub-hazards, survival functions, and so on, are not computed.

To create the scoring data set variable _*t*_ that the **Survival** node needs, the code below was submitted. The purpose of the code is to create the _*t*_ variable. You can write similar code of your own to create _*t*_ variables for other data sets that you want to score.

```
data sampsio.cellphone_score;
set sampsio.cellphone;
format _currentdate MMDDYY10.0 ;
_currentdate=input("31DEC2000",anydtdte10.);
_T_=intck("MONTH",activation_date, _currentdate);
drop Target _currentdate;
run;
```

The SAS **`intck`** function returns the number of time intervals in a given time span. The first argument of the SAS **`intck`** function is the time unit that you want to use (MONTH in this example). The second and third arguments are the “from” and “to” values for the time interval. In this example, the “from” argument is the activation date of a customer’s cell account. The “to” argument is the censoring date that defines the end of the analysis interval.

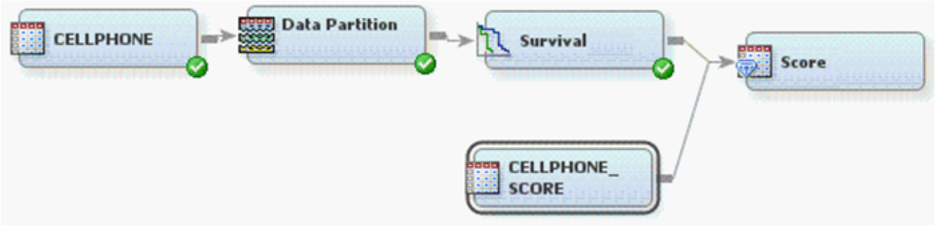
Right-click the Data Sources folder in the Project Panel, and then select **Create Data Source** to launch the Data Source Wizard.

In the Data Source Wizard, do the following to create your score data set as a SAS Enterprise Miner Data Source:

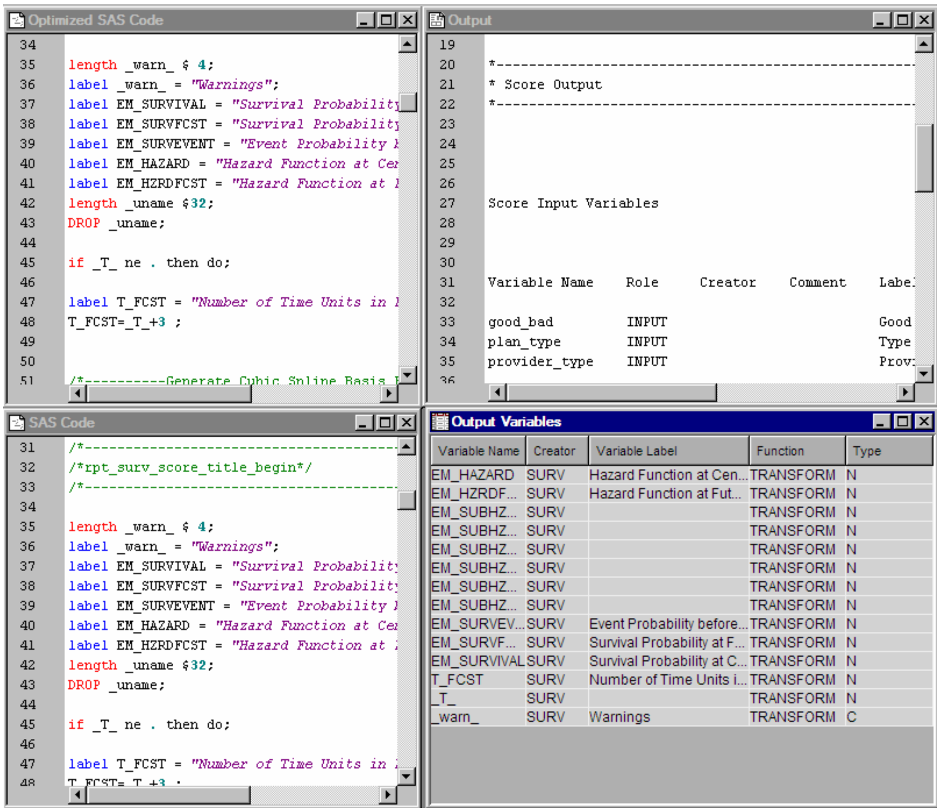
1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter SAMPSIO.CELLPHONE_SCORE in the **Table** field and click **Next**.
3. Continue to the Metadata Advisor and choose the **Advanced Metadata Advisor**.

4. In the Column Metadata window, make the following variable role assignments:
    a. Set the role of the variables ACCOUNT_NUM, ACTIVATION_DATE, DEACTIVATION_DATE, and DISABLE to **Rejected**.
    b. Set the roles of GOOD_BAD, PLAN_TYPE, and PROVIDER_TYPE, and _*t*_ to **Input**.
    c. Click **Next**.
5. Continue to the Data Source Attributes window and change the **Role** of the score data set to **Score**. Click **Next**.
6. Accept the default settings for the remainder of the Data Source Wizard windows. Select **Next** for each window, and when you reach the last Data Source Wizard window, select **Finish**.
7. The SAMPSIO.CELLPHONE_SCORE data set should appear in your Data Sources folder in the SAS Enterprise Miner Project Panel.

Drag the CELLPHONE_SCORE data source that you just created from the Data Sources folder in the Project Panel onto the Diagram Workspace. Next, drag a **Score** node from the **Assess** tab of the node Tools bar onto the Diagram Workspace. Then, connect the nodes as shown below:



Right-click the Score node and select **Run**. After the Process Flow Diagram runs, follow the prompts to open the Results browser.



```
length _warn_ $ 4;
label _warn_ = "Warnings";
label EM_SURVIVAL = "Survival Probability
label EM_SURVFCST = "Survival Probability
label EM_SURVEVENT = "Event Probability B
label EM_HAZARD = "Hazard Function at Cen
label EM_HZRDFCST = "Hazard Function at F
length _uname $32;
DROP _uname;

if _T_ ne . then do;

label T_FCST = "Number of Time Units in F
T_FCST=_T_+3 ;


/*----------Generate Cubic Spline Basis F
```

```
*---------------------------------------------------
* Score Output
*---------------------------------------------------





Score Input Variables



Variable Name    Role     Creator    Comment    Label

good_bad         INPUT                          Good
plan_type        INPUT                          Type
provider_type    INPUT                          Prov
```

```
/*---------------------------------------
/*rpt_surv_score_title_begin*/
/*---------------------------------------

length _warn_ $ 4;
label _warn_ = "Warnings";
label EM_SURVIVAL = "Survival Probability
label EM_SURVFCST = "Survival Probability
label EM_SURVEVENT = "Event Probability B
label EM_HAZARD = "Hazard Function at Cen
label EM_HZRDFCST = "Hazard Function at F
length _uname $32;
DROP _uname;

if _T_ ne . then do;

label T_FCST = "Number of Time Units in F
T_FCST=_T_+3 ;
```

Variable Name	Creator	Variable Label	Function	Type
EM_HAZARD	SURV	Hazard Function at Cen...	TRANSFORM	N
EM_HZRDF...	SURV	Hazard Function at Fut...	TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SURVEV...	SURV	Event Probability before...	TRANSFORM	N
EM_SURVF...	SURV	Survival Probability at F...	TRANSFORM	N
EM_SURVIVAL	SURV	Survival Probability at C...	TRANSFORM	N
T_FCST	SURV	Number of Time Units i...	TRANSFORM	N
_T_	SURV		TRANSFORM	N
warn	SURV	Warnings	TRANSFORM	C

The SAS scoring code is shown in the left two panes. The Output Variables tables shows that the hazard, sub-hazards, and survival functions were all generated.

Part 17

Node Reference: Time Series Nodes

Chapter 81

Time Series Data Preparation Node

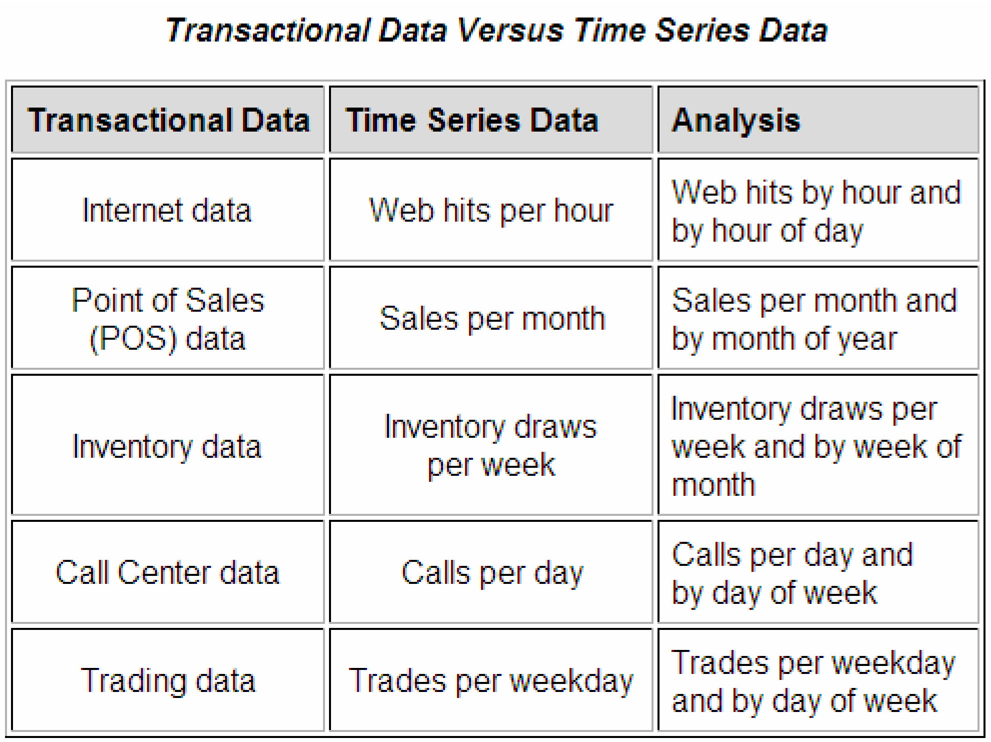
Time Series Data Preparation Node

Overview of Time Series Data Preparation

Over time, businesses collect large amounts of data related to the business activities and transactions that they conduct with their customers, suppliers, and monitoring devices. Time Series analysis enables you to better understand trends and seasonal variations in time series data, such as the buying patterns of your customers, your buying patterns from your suppliers, or the ongoing performance of your equipment and machinery.

The **TS Data Preparation** node helps you organize and format your data for time series data mining. For example, a business might have many suppliers and many customers, both with very large sets of associated transactional data. Traditional data mining tasks become difficult because of the size of the data set. By condensing the information into a time series, you can discover trends and seasonal variations in the data, such as customer and supplier habits that might not have been visible in the transactional data.

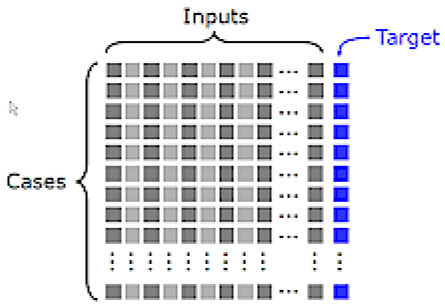
Transactional data is timestamped data that is collected over time at no particular frequency. By contrast, time series data is timestamped data that is collected over time at a specific frequency. The following table displays examples of transactional data, the data that results from conversion to time series data, and the analysis that might be done on the time series data.



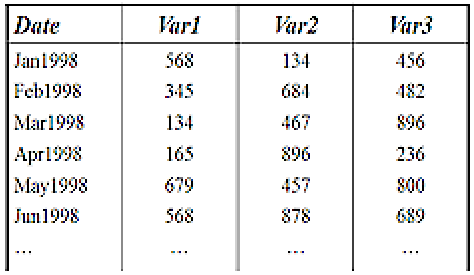
Transactional Data Versus Time Series Data

Transactional Data	Time Series Data	Analysis
Internet data	Web hits per hour	Web hits by hour and by hour of day
Point of Sales (POS) data	Sales per month	Sales per month and by month of year
Inventory data	Inventory draws per week	Inventory draws per week and by week of month
Call Center data	Calls per day	Calls per day and by day of week
Trading data	Trades per weekday	Trades per weekday and by day of week

Below is an example of transactional or cross-sectional data organization:



Below is an example of time series data organization:



Date	*Var1*	*Var2*	*Var3*
Jan1998	568	134	456
Feb1998	345	684	482
Mar1998	134	467	896
Apr1998	165	896	236
May1998	679	457	800
Jun1998	568	878	689
...	...	...	...

The frequency that is associated with the time series data varies with the business problem at hand. The frequency or time interval might be hourly, daily, weekly, monthly, quarterly, yearly, or many other variants of the basic time intervals. The choice of frequency is an important modeling decision. The time series data in the table above uses monthly time intervals.

How does one get from raw, timestamped transactional data to data that can be easily included into a predictive time series model? The first step is to transform the irregularly recorded timestamped data into data that is measured on a regular time interval. For this, we define a time interval suitable for our analysis (hour, day, week, and so on) and accumulate the data for the chosen interval. This accumulation can result in a variety of statistics, such as daily total, daily average, daily minimum, maximum, and so on. An analyst might want to keep several aggregated statistics, which can result in a substantial number of variables.

There are often several cross-sectional variables available in time series data, such as customer regions, customer groups, products monitored, and so on. These cross-sectional variables are called Cross ID variables in SAS Enterprise Miner. The analyst typically decides which cross-sectional variables to use for data aggregation. For each category of the variables, an aggregated time series must be created using the **TS Data Preparation** node in SAS Enterprise Miner.

To prepare transactional or cross-sectional data for conversion to time series data with the **TS Data Preparation** node, consider the following steps:

1. **Create time series ID**: Time series data cannot be formed without a time series ID. Time series data requires one observation per time period. The time series ID must be a numeric variable that uniquely identifies observations in the input and output data sets. The best variable candidates for time series ID variable values are SAS DATE or DATETIME values.

2. **Create time series metadata**: Specify time series data variable roles (Time ID, Target, Cross-ID, or Rejected) and levels (Interval or Nominal) for the input data set that you want to use for time series analysis.

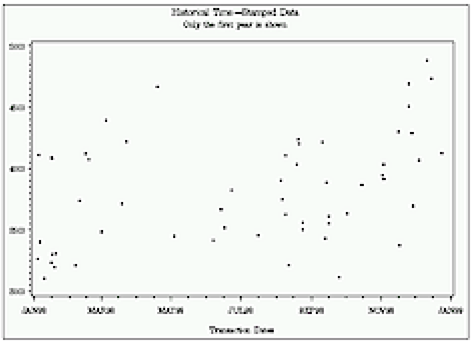
3. **Transpose time series data**: Cross-sectional data sets that contain timestamped information that is stored in rows of customer transactions can be transposed into time series data sets that contain customer transactions stored in rows of timestamped intervals.

4. **Detect and specify time interval, seasonality, start and end times**: Specify the necessary time series parameters for periodicity, seasonality, and analytic interval.

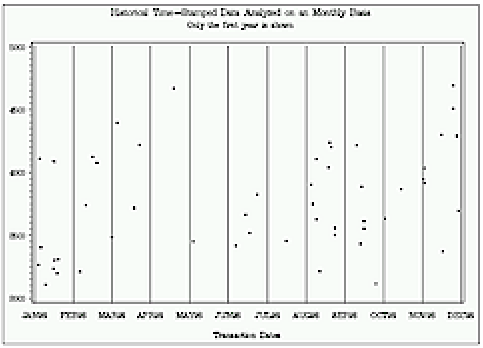
5. **Accumulate transactional data into time-based bins to form time series data**:

   Data accumulation is the process used to convert transactional data into time series data, and in some cases, to change the frequency of existing time series data. The main difference between transactional data and time series data is that transactional data has no time frequency specification. Transactional data is converted into time series data by introducing a time frequency. This is accomplished by bundling, or aggregating transactional data into quarterly, monthly, weekly, daily, hourly, or some other interval measure that represents a frequency of observation. When transactional data has a frequency attribute, it can be processed and handled as time series data.

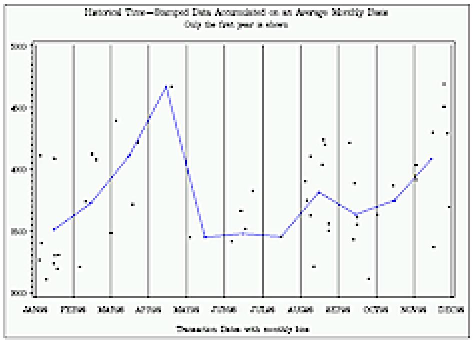
   You can use the Time Interval properties of the **TS Data Preparation** node to specify frequency information about your accumulated data, such as SAS time interval, interval start and end times, and seasonal cycle length.



Data recorded at no particular frequency



Apply monthly time bins



Accumulate data on a monthly basis

6. **Interpret zero and missing values in the time series data**: Specify how missing values (either actual or accumulated) are to be interpreted in the accumulated time series data.

   For a given timestamped data set, each observation of the data set can be accumulated by a time index and a season index. Accumulation can generate zero or missing values. These missing or zero values might or might not have an interpretation. Data interpretation is performed after data accumulation.

   Time series missing value interpretation assigns numeric values that are based on global properties of the accumulated time series. Missing value interpretation is performed only on missing values. Nonmissing values are left unchanged.

   Time series zero value interpretation assigns numeric values that are based on global properties of the time series. Zero value interpretation is performed only on zero values. Nonzero values are left unchanged.

7. **Perform transformation, differencing, or other techniques to facilitate time series analysis**: Differencing can help identify the hidden nature of seasonal

dependencies in a time series when serial dependencies are removed. In a time series with a lag of *k*, differencing converts every i^{th} element of the series into its difference from the $(i-k)^{th}$ element. Removing some of the autocorrelations might eliminate them or make other seasonalities more apparent.

Also consider unknown events in your data. Sometimes you do not know the events, but examination of the data reveals outliers that indicate a variation from the stochastic process of your series. These outliers represent events for which you have no information, or could represent measurement or data collection errors. For the sake of analysis, these types of outliers should be replaced or removed from the data.

Input Data Requirements for the Time Series Data Preparation Node

The **TS Data Preparation** node requires that the following roles be assigned to variables in the **Input Data Source** node:

- **TimeID** — You must define a single numeric interval time ID variable. The time ID provides either timestamp information or sequential information. When the time ID variable does not contain a valid SAS date, time, or datetime format, the values of the time ID variable are interpreted as sequential numbers.
- **CrossID** (optional) — You can define any number of optional cross ID variables. A cross ID variable must have a non-interval measurement. A cross ID variable represents a cross-sectional dimension to the time series data.

Time Series Data Preparation Node Properties

Time Series Data Preparation Node General Properties

The following general properties are associated with the **TS Data Preparation** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **TS Data Preparation** node that is added to a diagram has a Node ID of TSDP. The second **TS Data Preparation** node that is added to a diagram has a Node ID of TSDP2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — TSDP window. The Imported Data — TSDP window contains a list of the ports that provide data sources to the **TS Data Preparation** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — TSDP window. The Exported Data — TSDP window contains a list of the output data ports that the **TS Data Preparation** node creates data for when it runs. Select

the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Time Series Data Preparation Node Train Properties

The following train properties are associated with the **TS Data Preparation** node:

- **Variables** — Use the Variables window to view variable information, and change variable values using the **TS Data Preparation** node. Select the ... button to open a window that contains the variables table. You can specify the Use and Report value of a variable, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution. By default, columns for the Name, Use, Report, Role, and Level of a variable are displayed.

  You can modify these values, and add additional columns with the following options:

  - **Apply** — Changes metadata based on the values that are supplied in the drop-down menus, check box, and selector field.
  - **Reset** — Changes metadata back to its state before use of the **Apply** button.
  - **Label** — Adds a column for a label for each variable.
  - **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
  - **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
  - **Statistics** — Adds statistics metadata for each variable.
  - **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.

Time Series Data Preparation Node Train Properties: Time Interval

The time interval variable specifies the unit of time to be used for measurement and prediction. The results of the analysis are expressed in terms of the chosen time scale. They should represent the smallest units that are measurable and actionable. Typically, the values represent days, weeks, months, quarters, or years.

- **Specify an Interval** — Specifies a valid SAS time interval, which is the frequency of the accumulated time series. The default **Automatic** setting enables the SAS Enterprise Miner software to heuristically detect and configure the time series interval. Alternatively, you can manually configure your time interval. For example, if the input data set consists of quarterly observations, then you can specify **Quarter** as your Specify an Interval value. Alternatively, you can specify **Year** as your

Specify an Interval value to output yearly accumulated time series. For more information, see the Accumulation property. The following interval values are available for selection:

- **Automatic** — The interval length is determined from the data.
- **Year** — Yearly
- **Semiyear** — Semiannual
- **Quarter** — Quarterly
- **Month** — Monthly
- **Semimonth** — 1st and 16th of each month
- **Ten Days** — 1st, 11th, and 21st of each month
- **Week** — Weekly
- **Weekday** — Daily, ignoring weekend days
- **Day** — Daily
- **Hour** — Hourly
- **Minute** — Every minute
- **Second** — Every second

Note the following conditions when selecting your time interval:

- When the time interval variable does not contain a valid SAS date, time, or datetime format, the time interval variable is interpreted as a sequential variable. Thus, there is no automatic detection of the time interval.
- When your time ID variable uses a valid SAS date format, you cannot specify a time interval shorter than **Day**.
- When your time ID variable uses a valid SAS time format, you cannot specify a time interval longer than **Hour**.
- Automatic time interval detection is not supported for a time ID variable with a valid SAS time format. You must specify either **Hour**, **Minute**, or **Second**.

- **Seasonal Cycle Selection** — Use the Seasonal Cycle Selection property to specify how the seasonal cycle should be configured.

  When the Seasonal Cycle Selection property is set to **User Specified**, use the Length of Cycle property to specify a value. When the value is set to **User Specified**, SAS Enterprise Miner performs seasonal analysis for time series with a sequential time ID variable that uses the seasonal length specified by the Length of Cycle property.

  When the Seasonal Cycle Selection property is set to **Default**, SAS Enterprise Miner automatically determines the cycle length based on the time interval that is configured in the Specify an Interval property. Furthermore, when set to **Default**, SAS Enterprise Miner does not perform seasonal analysis for time series with a sequential time ID variable (a time ID variable without SAS date, time, or datetime format).

- **Length of Cycle** — Use the Length of Cycle property to specify the seasonal cycle length if the Seasonal Cycle Selection property is set to **User Specified**. A valid value must be greater than 1. When the Seasonal Cycle Selection property is set to **Default**, the default cycle lengths by time interval types are as follows:

  - **Year** — 1 time unit per cycle (no seasonal analysis)
  - **Semiyear** — 2 time units per cycle

  - **Quarter** — 4 time units per cycle
  - **Month** — 12 time units per cycle
  - **Semimonth** — 24 time units per cycle
  - **Ten Days** — 36 time units per cycle
  - **Week** — 52 time units per cycle
  - **Weekday** — 5 time units per cycles
  - **Day** — 7 time units per cycle
  - **Hour** — 24 time units per cycle
  - **Minute** — 60 time units per cycle
  - **Second** — 60 time units per cycle
- **Start and End Time** — Specifies how to set the start and the end of the time series data. The **Default** setting specifies that each time series use its own start and end times. The initial values for user-specified start and end times are retrieved from the overall range of the values that are associated with the TimeID variables. The **User Specified** setting requires you to specify the values with the Date Time Selector property.
- **Date Time Selector** — Select the ... button to open a window that contains controls that you use to select the starting point and ending point of the time series data. When the TimeID variable is in SAS DATE or DATETIME format, you select a date or time to start and end the time series. When the TimeID variable is a sequential variable, you select the starting point and ending point for the time series.
- **Accumulation** — Specifies how the data set observations are to be accumulated within each time period. The frequency (width of each time interval) is specified by the **Specify an Interval** option. The **Accumulation** option is useful when there is more (or less) than one input observation in a given time period. The available values for the Accumulation property are as follows:
  - **Total** — observations are accumulated based on the total sum of all values in the time period.
  - **Average** — observations are accumulated based on the mean of all values in the time period.
  - **Minimum** — observations are accumulated based on the smallest value in the time period.
  - **Median** — observations are accumulated based on the median value of all values in the time period.
  - **Maximum** — observations are accumulated based on the largest value in the time period.
  - **First** — observations are accumulated based on the first value in the time period.
  - **Last** — observations are accumulated based on the last value in the time period.

Time Series Data Preparation Node Train Properties: Transformation Options

- **Transformation** — Specifies the transformation function that you want to use to transform your time series data. The time series data must be strictly positive. The transformation function choices are as follows: **None**, **Log**, **Square Root**, **Logistic**, and **Box-Cox**.

- **Box-Cox Parameter** — When you specify **Box-Cox** as the transformation function for the Transformation property, you use the Box-Cox Parameter property to specify the value for λ, the Box-Cox power parameter. The value for the power parameter λ must be a real number between -5 and 5. The default setting is 0.

Time Series Data Preparation Node Train Properties: Difference Options

Differencing can help identify the hidden nature of seasonal dependencies in a time series when serial dependencies are removed. In a time series with a lag of *k*, differencing converts every i^{th} element of the series into its difference from the $(i\text{-}k)^{th}$ element. Removing some of the autocorrelations might eliminate them or make other seasonalities more apparent.

- **Apply Differencing** — Use the Apply Differencing property to specify whether to apply differencing to the (accumulated) time series data. The default setting for the Apply Differencing property is **No**.
- **Difference Order** — Use the Difference Order property to specify the order of differencing that you want to apply. The order of differencing is the number of time period lags between the differenced time periods. A difference order of 1 performs differencing with a lag of 1 time period. A difference order of 12 performs differencing with a lag of 12 time periods. The Difference Order property setting must be a positive integer. The **Difference Order** setting is dimmed and unavailable if the Apply Differencing property is set to **No**.
- **Seasonal Differencing** — Use the Seasonal Differencing property to specify whether you want to apply seasonal differencing using the order of seasonal cycle. When set to **Yes**, the Seasonal Differencing property uses the Seasonal Cycle Selection property to determine the appropriate number of time period lags. For example, if the Seasonal Cycle Selection property is monthly, and the Apply Differencing property is set to **Yes**, and the Seasonal Differencing property is set to **Yes**, then differencing will be performed using a lag of 12 time periods.

Time Series Data Preparation Node Train Properties: Missing Values

- **Set Value** — Use the Set Value property to specify a replacement method when missing values are encountered in accumulated data. The available Set Value missing data replacement methods are as follows:
  - **Missing** — missing variable values are not replaced.
  - **Average** — missing variable values are replaced with the mean of all values for that variable.
  - **Minimum** — missing variable values are replaced with the minimum of all values for that variable.
  - **Maximum** — missing variable values are replaced with the maximum of all values for that variable.
  - **Median** — missing variable values are replaced with the median of all values for that variable.
  - **First** — missing variable values are replaced with the value of the first accumulated nonmissing variable.
  - **Last** — missing variable values are replaced with the value of the last accumulated nonmissing variable.
  - **Previous** — missing variable values are replaced with the value of the previous accumulated nonmissing variable.

  - **Next** — missing variable values are replaced with the value of the next accumulated nonmissing variable.
  - **Constant** — missing variable values are replaced with a constant value that you specify.

- **Constant Value for Missing Observations** — When the Set Value property is set to **Constant**, use the Constant Value for Missing Observations property to specify the real constant that you want to use as the replacement value for missing observations.
- **Zero Missing** — Use the Zero Missing property to specify how beginning and ending observation zero values in the input or accumulated data are interpreted in the accumulated time series.
  - **None** — beginning and ending zeros are left unchanged. **None** is the default setting.
  - **Left** — beginning zeros are set to **MISSING**.
  - **Right** — ending zeros are set to **MISSING**.
  - **Both** — both beginning and ending zeros are set to **MISSING**.

Time Series Data Preparation Node Train Properties: Transpose Options

You can use the **TS Data Preparation** node to transpose your data. Transposing your data by Time Series ID might be useful for similarity searches or time series clustering.

- **Transpose** — Use the Transpose property to specify if you want to transpose cross-sectional data or timestamped data in order to create time series data. The default setting for the Transpose property is **No**.
- **By Variable** — When the Transpose property is set to Yes, use the By Variable property to specify the BY variable that you want to use for the transpose data operation. You can assign the BY variable role to either By TimeID, or By TSID (Time Series ID).
- **Keep Variable Role** — When the Transpose property is set to Yes, use the Keep Variable Role property to indicate whether you want transposed variables to retain their role.

Time Series Data Preparation Node Report Properties

The following report properties are associated with the **TS Data Preparation** node:

- **Maximum Number of Plots** — Specifies the maximum number of time series that are plotted in the results window. The data is not retrained if this is the only property that you change. If you specify a value that is larger than the number of plots that your system is able to graph, then your time series plots will be empty.

Time Series Data Preparation Node Report Properties: Season Statistics

- **Export Season Statistics** — Specifies whether season statistics are exported. The seasonal statistics replace the time series information as the exported data set.
- **Seasonal Statistics** — Specifies which season statistics are exported. The following season statistics are available:
  - **All**
  - **Sum**

  - **Mean**
  - **Minimum**
  - **Maximum**
  - **Median**

Time Series Data Preparation Node Status Properties

The following status properties are associated with the **TS Data Preparation** node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Time Series Data Preparation Node Results

You can open the Results window of the **TS Data Preparation** node after a successful run by clicking **Results** in the Run Status window, or by right-clicking the node in the Diagram Workspace and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **TS Data Preparation** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **TS Data Preparation** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **TS Data Preparation** node run.
  - **Output** — the SAS output of the **TS Data Preparation** node run.
  - **Flow Code** — the **TS Data Preparation** node does not generate flow code.
- **Scoring**
  - **SAS Code** — the **TS Data Preparation** node does not generate SAS Code.
  - **PMML Code** — the **TS Data Preparation** node does not generate PMML code.

- **Model**
  - **Time Series Meta Data Table** — displays a table that shows the metadata structure for the converted time series table.
  - **TSID Map Table** — displays a table that shows the input variable grouping that is associated with each time series ID.
  - **TSID Map Summary Table** — displays a table that lists level, count, and frequency information for the crossID variables.
- **Plots**
  - **Time Series Summary** — displays in a bar plot a selected set of values for each cross ID variable that is configured in the **TS Data Preparation** node. The value choices available for the Time Series Summary plots are: Mean Value of Series, Sum of Series, Maximum Value of Series, Minimum Value of Series.
  - **Time Series Plots** — displays multiple time series in a single plot.
  - **Seasonal Statistics Plot** — displays a plot of the seasonal statistics that is indexed by the time interval. For example, if your data is accumulated monthly, then this plot contains 12 data points per time series.
- **Table** — displays a table that contains the underlying data used to produce a chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph wizard to modify an existing Results plot or create a Results plot of your own. The **Plot** menu item is dimmed and unavailable unless a Results chart or table is open and selected.

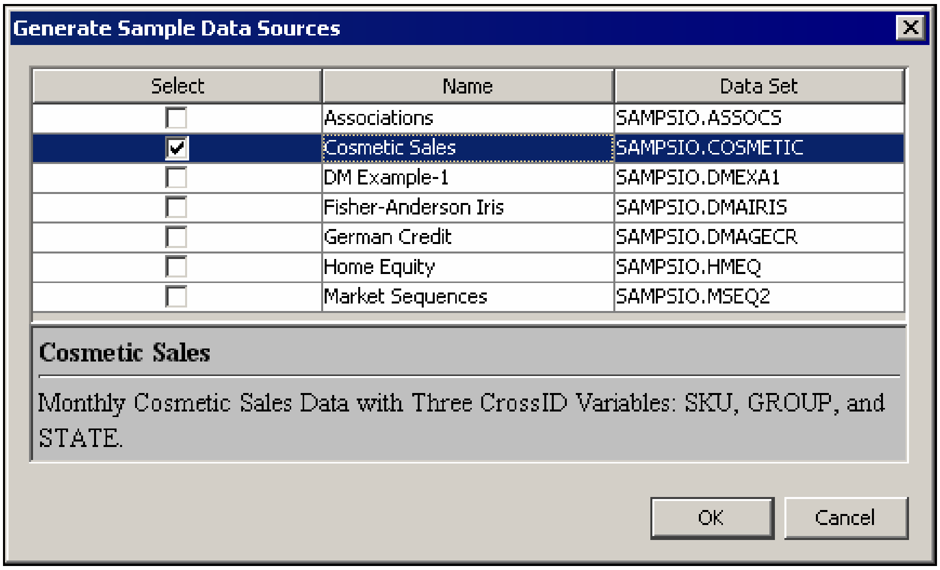
Time Series Data Preparation Node Examples

Example 1: Time Series Data Preparation

This example uses the Cosmetic Sales data set from the example SAMPSIO library that is included with SAS Enterprise Miner. The **TS Data Preparation** node is a precursor tool that enables you to submit time series-formatted data to successor node tools, such as the **TS Exponential Smoothing** and **TS Similarity** nodes. The focus of this example is to provide content on the configuration and use of the **TS Data Preparation** node. The Reference Help chapters for the Time Series Exponential Smoothing node on page 1313 and the Time Series Similarity node on page 1296 provide additional example process flow diagrams that highlight their respective analytical capabilities.

The **TS Data Preparation** node example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the SAS Enterprise Miner main menu. In the Generate Sample Data Sources window, select the Cosmetic Sales data set, as shown below.



Generate Sample Data Sources

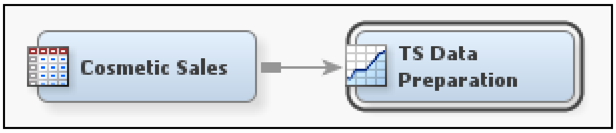
Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Cosmetic Sales

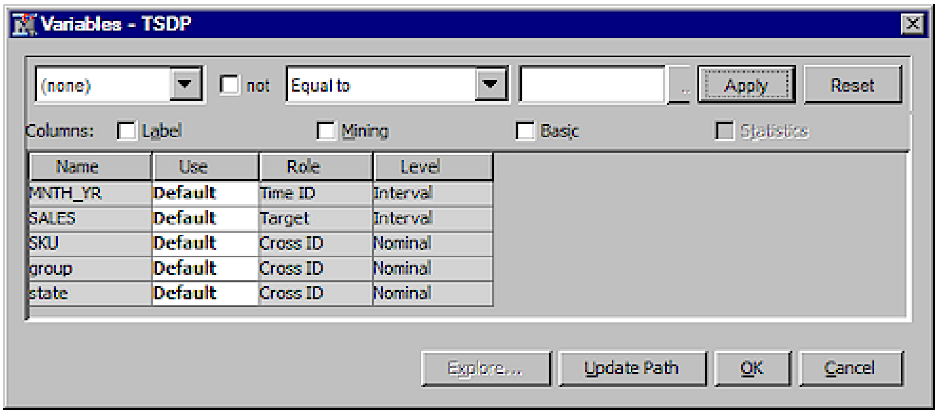
Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

Select the **Cosmetic Sales** node in your diagram workspace. In the Train properties group, ensure that the Role property is set to **Transaction**.

2. Next, drag the Cosmetic Sales data set from the Data Sources folder in the SAS Enterprise Miner Project Panel onto your Diagram Workspace. Drag the **TS Data Preparation** node from the **Time Series** tab of the Nodes Toolbar to your process flow diagram. Connect the Cosmetic Sales data set to the **TS Data Preparation** node as shown below.



3. Right-click the **TS Data Preparation** node and select **Edit Variables** from the menu. This opens the Variables window, where you can set the Use status for each variable.



Variables - TSDP

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Use	Role	Level
MNTH_YR	Default	Time ID	Interval
SALES	Default	Target	Interval
SKU	Default	Cross ID	Nominal
group	Default	Cross ID	Nominal
state	Default	Cross ID	Nominal

Notice that there are three Cross ID variables for the Cosmetic Sales data set. The **TS Data Preparation** node creates a separate time series for each unique combination of variable values for each Cross ID variable.

In the Cosmetic Sales data set, the SKU variable has 5 values, the state variable has 5 values, and the group variable has 3 values. Using all three Cross ID variables results in 5 * 5 * 3 = 75 different time series.

Select **OK** to close the Variables window.

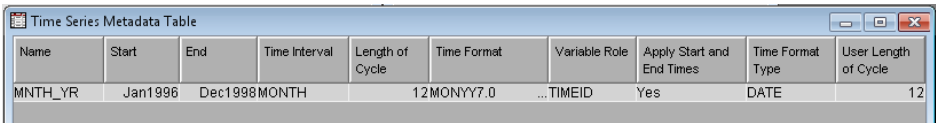
4. In the Properties Panel for the **TS Data Preparation** node, find the Set Value property in the Missing Value subgroup and set this to **Constant**, and set the Constant Value for Missing property to **0**.

   Next, find the Transpose Options subgroup, and set the value for the Transpose property to **Yes**. Set the By Variable property to **By TSID** and the Keep Variable Role property to **No**.

5. Right-click the **TS Data Preparation** node in the process flow diagram, and select **Run** from the pop-up menu. In the Confirmation window, click **Yes**.

6. Click **Results** in the Run Status window after the node successfully runs to open the Results browser.

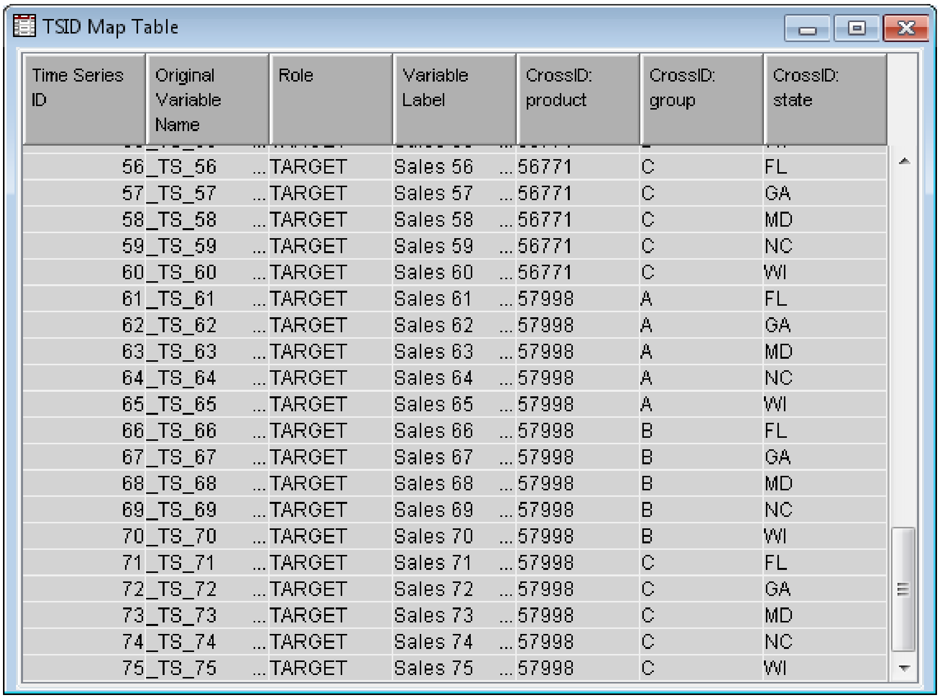
7. The Time Series Metadata Table window shows that the Cosmetic Sales input data uses MNTH_YR as the TimeID variable to analyze monthly data from January 1996 to December 1998.

   Time Series Metadata Table

Name	Start	End	Time Interval	Length of Cycle	Time Format	Variable Role	Apply Start and End Times	Time Format Type	User Length of Cycle
MNTH_YR	Jan1996	Dec1998	MONTH	12	MONYY7.0 ...	TIMEID	Yes	DATE	12

   The Time Series Map ID table shows that the original data set with one time series variable and three CrossID variables has been transposed and converted into 75 time series with 75 unique variables.

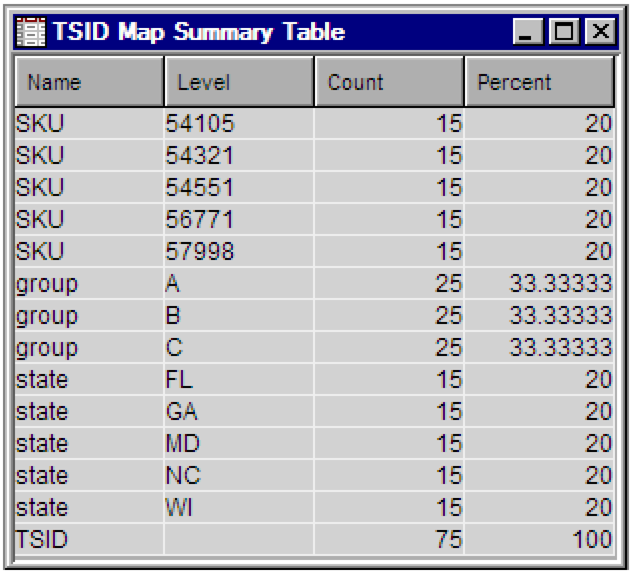
   

   TSID Map Table

Time Series ID	Original Variable Name	Role	Variable Label	CrossID: product	CrossID: group	CrossID: state
56	_TS_56 ...	TARGET	Sales 56 ...	56771	C	FL
57	_TS_57 ...	TARGET	Sales 57 ...	56771	C	GA
58	_TS_58 ...	TARGET	Sales 58 ...	56771	C	MD
59	_TS_59 ...	TARGET	Sales 59 ...	56771	C	NC
60	_TS_60 ...	TARGET	Sales 60 ...	56771	C	WI
61	_TS_61 ...	TARGET	Sales 61 ...	57998	A	FL
62	_TS_62 ...	TARGET	Sales 62 ...	57998	A	GA
63	_TS_63 ...	TARGET	Sales 63 ...	57998	A	MD
64	_TS_64 ...	TARGET	Sales 64 ...	57998	A	NC
65	_TS_65 ...	TARGET	Sales 65 ...	57998	A	WI
66	_TS_66 ...	TARGET	Sales 66 ...	57998	B	FL
67	_TS_67 ...	TARGET	Sales 67 ...	57998	B	GA
68	_TS_68 ...	TARGET	Sales 68 ...	57998	B	MD
69	_TS_69 ...	TARGET	Sales 69 ...	57998	B	NC
70	_TS_70 ...	TARGET	Sales 70 ...	57998	B	WI
71	_TS_71 ...	TARGET	Sales 71 ...	57998	C	FL
72	_TS_72 ...	TARGET	Sales 72 ...	57998	C	GA
73	_TS_73 ...	TARGET	Sales 73 ...	57998	C	MD
74	_TS_74 ...	TARGET	Sales 74 ...	57998	C	NC
75	_TS_75 ...	TARGET	Sales 75 ...	57998	C	WI

   Each time series is named _TS_n where n is the value of the TSID created for the series. Each time series is labeled `Target_n`, where Target is the label of the original variable and n is the value of the TSID created for the series. Each of the 75 time series observations represents a unique combination of the crossID variables product, group, and state.

   The TSID Map Summary Table lists level, count, and frequency information for the crossID and TSID variables. You can see the 5 SKU values, the 3 group values, and the 5 state values whose unique combinations result in 75 TSIDs.

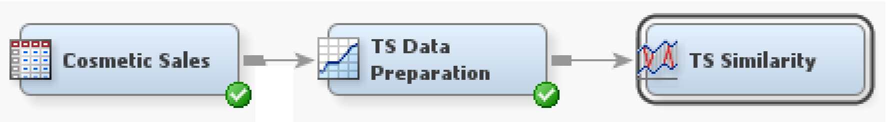


TSID Map Summary Table

Name	Level	Count	Percent
SKU	54105	15	20
SKU	54321	15	20
SKU	54551	15	20
SKU	56771	15	20
SKU	57998	15	20
group	A	25	33.33333
group	B	25	33.33333
group	C	25	33.33333
state	FL	15	20
state	GA	15	20
state	MD	15	20
state	NC	15	20
state	WI	15	20
TSID		75	100

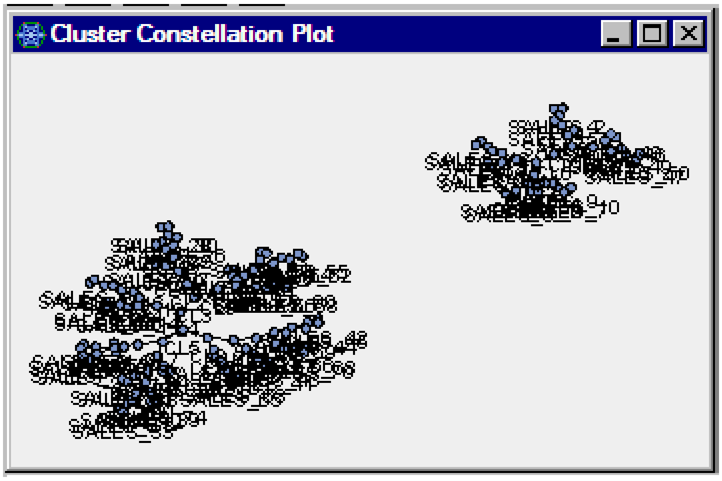
In addition, the data is now organized for further time series data mining with SAS Enterprise Miner time series tools, such as Time Series Similarity. Close the Results window.

8. To verify that the **TS Data Preparation** node has properly structured the data for time series analysis, drag a **TS Similarity** node from the **Time Series** tab of the Nodes Toolbar onto the diagram, and connect it as shown below:



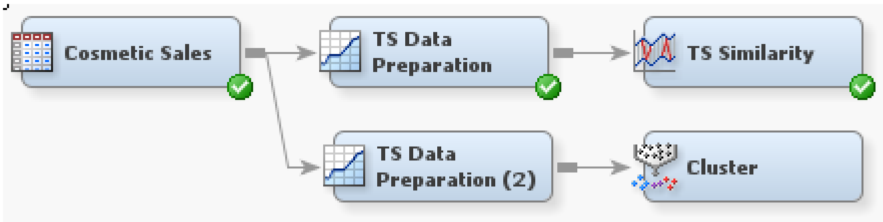
9. Use the **TS Similarity** node in its default configuration. Right-click the node, and select **Run** from the menu. Click **Yes** in the Confirmation window. When the node completes running, select **Results** in the Run Status window to open the Time Series Similarity Results browser.

10. The **TS Similarity** node was able to process the time series data exported by the **TS Data Preparation** node. A cursory examination of the Time Series Similarity Results browser shows a cluster constellation plot that indicates two distinct clusters among the time series input data.



Further discussion of the Time Series Similarity results is beyond the scope of this example. More information about the **TS Similarity** node is available in the Overview of Similarity Analysis on page 1289 section.

11. You can attach successor nodes to the **TS Data Preparation** node that are not time series data mining nodes. For example, to learn more about the clusters that we saw in the Time Series Similarity results, add another **TS Data Preparation** node and a **Cluster** node (from the **Explore** tab of the Node Toolbar) to your diagram, and connect them as shown below:



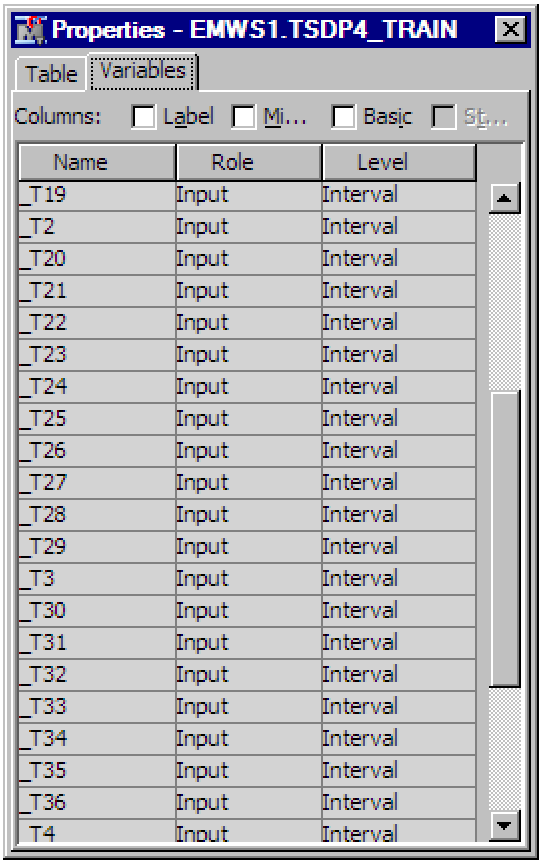
In order to use the **Cluster** node, you must change the role of the Cosmetic Sales data set. To do so, select the Cosmetic Sales data set and find the Role property in the Train property group. Set the value of Role to **Raw**. Alternatively, you can set the value of Role to **Train**.

To transpose the time series data for clustering, select the **TS Data Preparation (2)** node and set the Transpose property to **Yes**, and set the By Variable property to **By Time ID**. The **TS Data Preparation** node converts each time point in the Cosmetic Sales data to an input variable suitable for clustering analysis.

Configure the **Cluster** node by setting the Internal Standardization property to **None**.

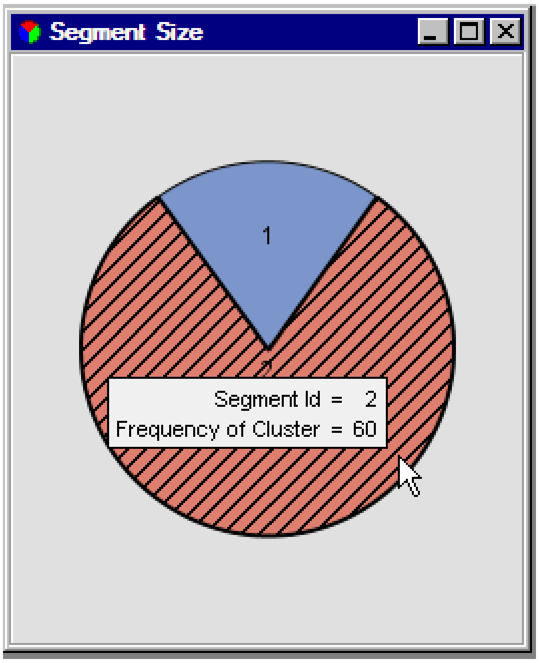
Right-click the **Cluster** node and select **Run** to run the new process flow diagram. In the Confirmation window, click **Yes**. When the node completes running, click **OK** in the Run Status window.

12. Select the **TS Data Preparation (2)** node, and then select the ellipsis button ... for the Exported Data property, located in the General properties. This opens the Exported Data — TS Data Preparation (2) window. Select the TRAIN row and click **Properties** to open the training data Properties window. In the Properties window, select the **Variables** tab. You can see in the exported data set that the **TS Data Preparation** node created 36 (3 years of data * 12 months per year = 36 months) input variables (**_T1** ... **_T36**) for cluster analysis.



Name	Role	Level
_T19	Input	Interval
_T2	Input	Interval
_T20	Input	Interval
_T21	Input	Interval
_T22	Input	Interval
_T23	Input	Interval
_T24	Input	Interval
_T25	Input	Interval
_T26	Input	Interval
_T27	Input	Interval
_T28	Input	Interval
_T29	Input	Interval
_T3	Input	Interval
_T30	Input	Interval
_T31	Input	Interval
_T32	Input	Interval
_T33	Input	Interval
_T34	Input	Interval
_T35	Input	Interval
_T36	Input	Interval
_T4	Input	Interval

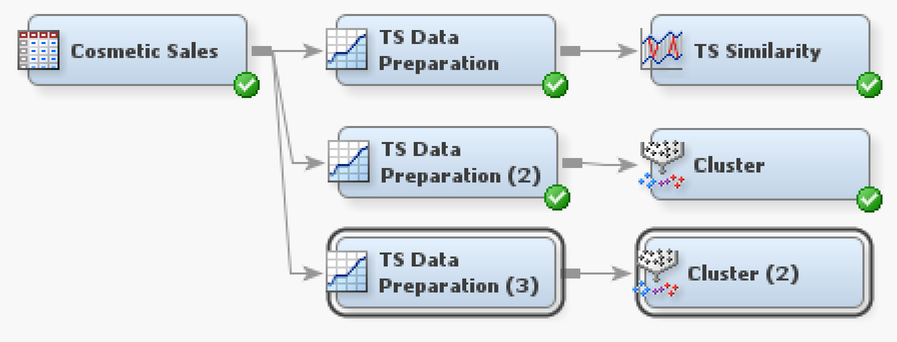
Right-click the **Cluster** node and select **Results** to open the Cluster Results window. A quick look at the Segment Size plot for the 36 monthly time periods shows two cluster segments. The 75 observations in the training data were clustered into the two segments with frequencies of 15 and 60.



Segment Id = 2
Frequency of Cluster = 60

You can use the **TS Data Preparation** node to reduce the number of dimensions by changing the time interval and seeing if clusters change.

13. Drag another **TS Data Preparation** node and **Cluster** node from the Nodes Toolbar onto the diagram and connect them as shown below:

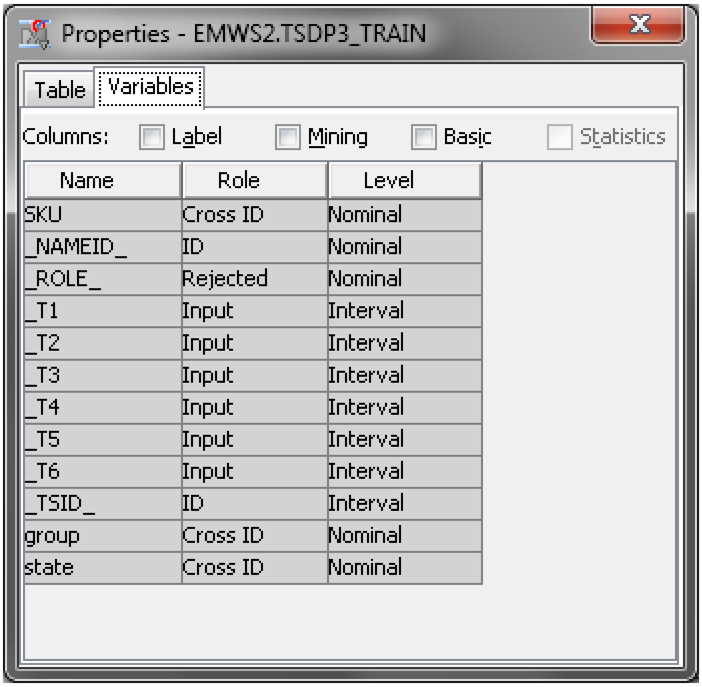


We want to use the **TS Data Preparation** node to reduce dimensions. Instead of examining 36 monthly periods, we will accumulate and summarize the monthly data semiannually, resulting in 6 periods. To configure the **TS Data Preparation (3)** node, set the Specify an Interval property to **Semiyear**, set the Accumulation property to **Average**, set the Transpose property to **Yes**, and set the By Variable property to **By Time ID**.

In the **Cluster (2)** node, set the Internal Standardization property to **None**.

Right-click the **Cluster (2)** node and select **Run** to run the new process flow diagram. Click **Yes** in the Confirmation window. When the node completes running, select **OK** in the Run Status window.

14. First, we will confirm that we exported data to the **Cluster** node with six summarized semiannual time period inputs. Select the **TS Data Preparation (3)** node, and then select the ellipsis button ... in the Exported Data property to open the Exported Data — TS Data Preparation (3) window. Select the TRAIN row and click **Properties** to open the training data Properties window. In the Properties window, select the **Variables** tab. You can see in the exported data set that the **TS Data Preparation** node created 6 semiannual input variables (_T1 ... _T6) for cluster analysis.

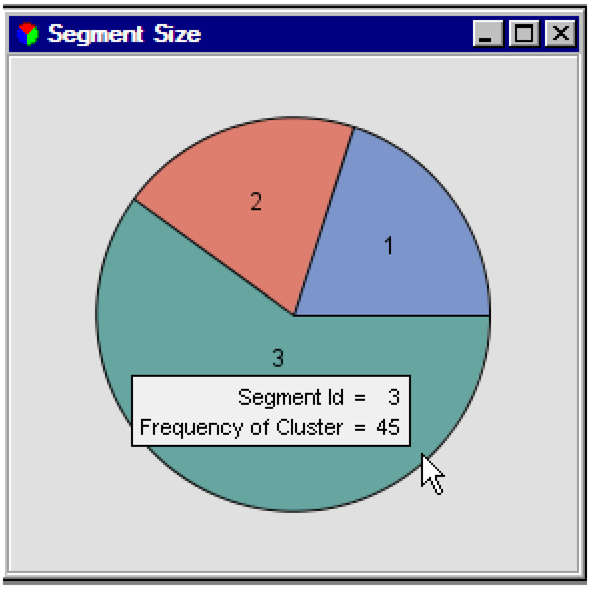

Properties - EMWS2.TSDP3_TRAIN

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Role	Level
SKU	Cross ID	Nominal
NAMEID	ID	Nominal
ROLE	Rejected	Nominal
_T1	Input	Interval
_T2	Input	Interval
_T3	Input	Interval
_T4	Input	Interval
_T5	Input	Interval
_T6	Input	Interval
TSID	ID	Interval
group	Cross ID	Nominal
state	Cross ID	Nominal

Close the Properties window and the Exported Data window.

15. Right-click the **Cluster (2)** node and select **Results** to open the Cluster Results window. A quick look at the Segment Size plot for the 6 semiannual time periods shows three cluster segments. The 75 observations in the training data were clustered into the three segments with frequencies of 15, 15, and 45.

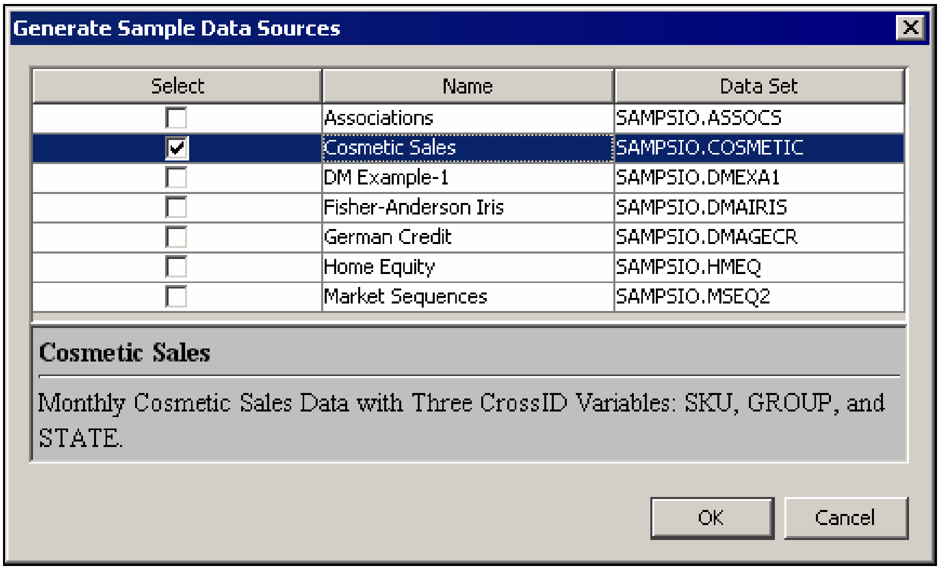


Example 2: Output Seasonal Statistics for Clustering

This example uses the Cosmetic Sales data set from the example SAMPSIO library that is included with SAS Enterprise Miner. First, the **TS Data Preparation** node is used to export seasonal statistics. Then, the **Cluster** node is used to group the time series based on the seasonal information exported from the **TS Data Preparation** node.

The **TS Data Preparation** node example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the SAS Enterprise Miner main menu. In the Generate Sample Data Sources window, select the Cosmetic Sales data set, as shown below.



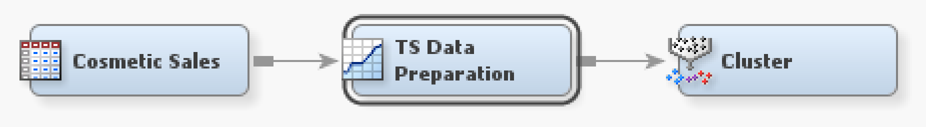
Generate Sample Data Sources

Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Cosmetic Sales

Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

2. Next, drag the Cosmetic Sales data set from the Data Sources folder in the SAS Enterprise Miner Project Panel onto your Diagram Workspace. Drag the **TS Data Preparation** node from the **Time Series** tab of the Nodes Toolbar to your process flow diagram. Connect the Cosmetic Sales data set to the **TS Data Preparation** node. Drag the **Cluster** node from the **Explore** tab of the Nodes Toolbar to your process flow diagram. Connect the **TS Data Preparation** node to the **Cluster** node, as shown below.

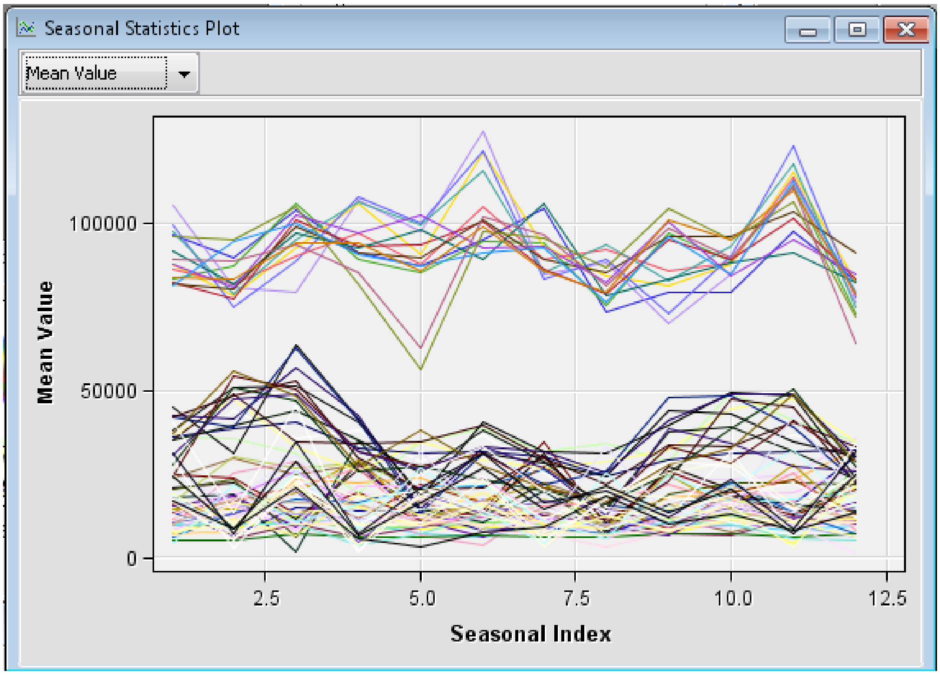


3. Select the **Cosmetic Sales** node in your diagram workspace. In the Train properties group, change the Role property to **Train**.

4. Select the **TS Data Preparation** node in your diagram workspace. In the Report properties group, set the Export Seasonal Statistics property to **Yes**and the Seasonal Statistics property to **Mean**.

5. Right-click the **Cluster** node and select **Run**from the drop-down menu. In the Confirmation window, click **Yes**. After a successful run on the **Cluster** node, click **OK** in the Run Status window.

6. Right-click the **TS Data Preparation** node and click **Results** from the drop-down menu. In the Seasonal Statistics Plot window, shown below, the summary statistics for each time series are plotted, indexed by the seasonal index. By default, the mean value is plotted. You can use the drop-down menu in the upper left corner of the Seasonal Statistics Plot window to select a different statistic. Available statistics include the sum, minimum, maximum, and median values of observations at each of the seasonal indices. The Seasonal Statistics Plot shown below indicates that mean seasonal sequences form two groups.



Close the Results window.

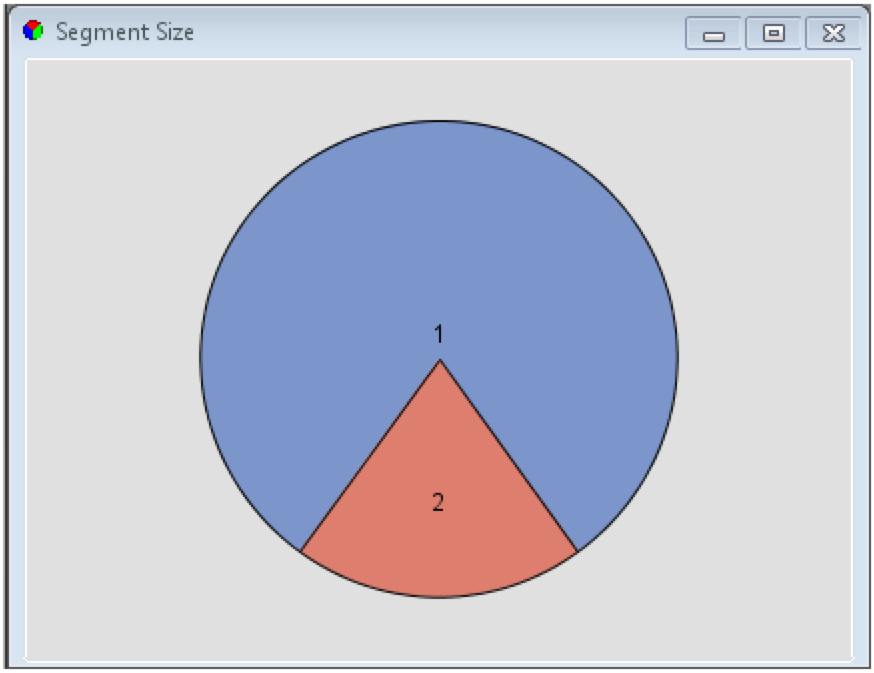
7. Select the **TS Data Preparation** node and click the ellipses next to the Exported Data property. Unlike Example 1, the seasonal summary statistics are exported instead of the time series data.

   In the Exported Data window, select the TRAIN data set and click **Explore**.

   In the TSDP_TRAIN variables table, note that the exported data contains the variables SEASON1 through SEASON12. The **TS Data Preparation** node detected that the Cosmetic Sales data set contains monthly data and set the default seasonal length to 12. You can use the Seasonal Cycle Selection property to define a user-specified time interval and seasonal length.

   Close the Explore window, and then close the Exported Data window.

8. Right-click the **Cluster** node and click **Results**from the drop-down menu. The Segment Size window indicates, as expected, that the seasonal time series consists of two clusters. The Mean Statistics window provides several statistics for each cluster.



Close the Results window.

References

Barry, M. J. and Linoff, G. S. 1997. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. New York, NY: John Wiley & Sons, Inc.

Box, G. E. P., Genkins, G. M., and Reinsel, G. C. 1994. *Time Series Analysis: Forecasting and Control*. Englewood Cliffs,, N.J.: Prentice Hall, Inc.

Brockwell, P. J. and Davis, R. A. 1996. *Introduction to Time Series and Forecasting*. New York, NY: Springer-Verlag.

Chatfield, C. 2000. *Time Series Models*. Boca Raton, FL: Chapman & Hall / CRC.

Fuller, W. A. 1995. *Introduction to Statistical Time Series*. New York, NY: John Wiley & Sons, Inc.

Hamilton, J. D. 1994. *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Han, J. and Kamber, M. 2001. *Data Mining: Concepts and Techniques*. San Francisco, CA: Morgan Kaufmann Publishers.

Harvey, A. C. 1994. *Time Series Models*. Cambridge, MA: MIT Press.

Leonard, M. J. 2002. “Large Scale Automatic Forecasting: Millions of Forecasts.” *International Symposium of Forecasting*, Dublin, Ireland, 156 pp.

Leonard, M. J. 2004. “Large Scale Automatic Forecasting with Inputs and Calendar Events.” *International Symposium of Forecasting*, Sydney, Australia, 27 pp.

Leonard, M. J. and Wolfe, B. L., 2005. “Mining Transactional and Time Series Data.” *SAS Users Group International (SUGI) 30*, Philadelphia, PA, 142 pp.

Leonard, M. J. 2010. “Time Series Data Preparation Using SAS®.” Whitepaper Draft, SAS Institute, Cary, NC.

Makridakis, S. G., Wheelright, S. C., and Hyndman, R. J. 1997. *Forecasting: Methods and Applications*. New York, NY: John Wiley & Sons, Inc.

Pyle, D. 1999. *Data Preparation for Data Mining*. San Francisco, CA: Morgan Kaufman Publishers.

Chapter 82
Time Series Similarity Node

Time Series Similarity Node

Overview of Similarity Analysis

Over time, websites and transactional databases collect large amounts of timestamped data that is related to suppliers and customers. Analyzing this timestamped data can help business leaders make better decisions by enabling them to listen to their suppliers and customers. A business might have many suppliers and customers and might have a set of transactions that are associated with each supplier and customer. However, each set of transactions might be quite large, making it difficult to perform many traditional data mining tasks. This problem can be addressed using large-scale similarity analysis that uses similarity measures combined with automatic time series analysis and decomposition. After similarity analysis, traditional data mining techniques can then be applied to similarity analysis results along with other profile data.

Given two ordered numeric sequences (input and target), such as two time series, a similarity measure is a metric that measures the distance between the input sequence and the target sequence. At the same time, it takes into account the ordering. Similarity measures can be computed between the input sequence and the target sequence in addition to similarity measures that “slide” the target sequence with respect to the input sequence. The “slides” can occur by observation index (sliding-sequence similarity measures) or by seasonal index (seasonal-sliding-sequence similarity measures).

To compute the similarity measure between two time series, you need to prepare the time series data for comparison. This preparation includes normalizing sequences, scaling sequences, interpreting missing values, and computing metrics or measures. When used as a predecessor node, the **TS Data Preparation** node can transpose the time series data or perform other data preparation tasks.

The **TS Similarity** node computes similarity measures for timestamped data with respect to time. The tool does so by accumulating the data into a time series format, and then it computes several similarity measures for sequentially ordered numeric data by respecting the ordering of the data.

Similarity measure analysis can be used to compare target and input sequences of different lengths by compressing or expanding the input sequence with respect to the target sequence. At the same time, similarity measure analysis preserves the order of the sequence elements. Dynamic time warping expands or compresses the time dimension and uses a dynamic program that minimizes a cost function, such as the sum of squared or absolute distances.

A similarity matrix can be constructed by computing the pairwise measures between time series sequences. The similarity matrix, also called the distance matrix, can be exported for further clustering analysis. The **TS Similarity** node performs basic hierarchical clustering when no target variable is specified or when clustering analysis is desired. In these cases, the **TS Similarity** node generates a distance matrix among all input sequences and returns clustering segments, which indicate the similarity among all input sequences.

The **TS Similarity** node also provides controls that enable modelers to specify parameters such as similarity measure, sequence sliding, normalization, interval, accumulation, similarity matrix, hierarchical clustering, as well as expanded and compressed sliding sequence ranges.

Time series similarity measure analysis is useful for clustering or classifying time series sequences that vary in length or timing. Similarity measure analysis is useful for pattern recognition, new product forecasting, and short life-cycle forecasting.

Input Data Requirements for the Time Series Similarity Node

The **TS Similarity** node requires panel data. Therefore, if the input data set contains a CrossID variable, the **TS Similarity** node must be preceded by a **TS Data Preparation** node with the Transpose property set to **Yes**.

The **TS Similarity** node requires at least one interval input variable.

The **TS Similarity** node requires that the following roles be assigned to variables in the **Input Data Source** node:

- **TimeID** — You must define a single numeric interval time ID variable. The time ID provides either timestamp information or sequential information. When the time ID variable does not contain a valid SAS date, time, or datetime format, the values of the time ID variable are interpreted as sequential numbers.

Time Series Similarity Node Properties

Time Series Similarity General Properties

The following general properties are associated with the **TS Similarity** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **TS Similarity** node that is added to a diagram has a Node ID of TSSIM. The second **TS Similarity** node added to a diagram has a Node ID of TSSIM2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Time Series Similarity window. The Imported Data — Time Series Similarity

window contains a list of the ports that provide data sources to the **TS Similarity** node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and take one of the following actions:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Time Series Similarity window. The Exported Data — Time Series Similarity window contains a list of the output data ports that the **TS Similarity** node creates data for when it runs. Click the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and take one of the following actions:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Time Series Similarity Train Properties

The following train properties are associated with the **TS Similarity** node:

- **Variables** — Use the Variables property of the **TS Similarity** node to specify the properties of each variable that you want to use in your data source. Click the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can select which variables you want included in reports.

- **Specify an Interval** — Specifies a valid SAS time interval, which is the frequency of the accumulated time series. The default **Automatic** setting enables the SAS Enterprise Miner software to heuristically detect and configure the time series interval. Alternatively, you can manually configure your time interval. For example, if the input data set consists of quarterly observations, then you can specify **Quarter** as your Specify an Interval value. Alternatively, you can specify **Year** as your Specify an Interval value to output yearly accumulated time series. For more information, see the Accumulation property. The following interval values are available for selection:

  - **Automatic** — The interval length is determined from the data.
  - **Year** — Yearly
  - **Semiyear** — Semiannual
  - **Quarter** — Quarterly

  - **Month** — Monthly
  - **Semimonth** — 1st and 16th of each month
  - **Ten Days** — 1st, 11th, and 21st of each month
  - **Week** — Weekly
  - **Weekday** — Daily, ignoring weekend days
  - **Day** — Daily
  - **Hour** — Hourly
  - **Minute** — Every minute
  - **Second** — Every second

  Note the following conditions when selecting your time interval:

  - When the time interval variable does not contain a valid SAS date, time, or datetime format, the time interval variable is interpreted as a sequential variable. Thus, there is no automatic detection of the time interval.
  - When your time ID variable uses a valid SAS date format, you cannot specify a time interval shorter than **Day**.
  - When your time ID variable uses a valid SAS time format, you cannot specify a time interval longer than **Hour**.
  - Automatic time interval detection is not supported for a time ID variable with a valid SAS time format. You must specify either **Hour**, **Minute**, or **Second**.

- **Accumulation** — specifies how the data set observations are accumulated within each time interval. The time interval is specified by the Specify an Interval property in the **TS Data Preparation** node. The Accumulation property is particularly useful when there are gaps in the input data or when there are multiple input observations that coincide with a particular time period.

  The accumulation methods available are as follows:

  - **Total** — Observations are accumulated based on the total sum of the values in a time interval.
  - **Average** — Observations are accumulated based on the average of the values in a time interval.
  - **Minimum** — Observations are accumulated based on the minimum value of a time interval.
  - **Median** — Observations are accumulated based on the median value of a time interval.
  - **Maximum** — Observations are accumulated based on the maximum value of a time interval.
  - **First** — Observations are accumulated based on the first value in a time interval.
  - **Last** — Observations are accumulated based on the last value in a time interval.

- **Similarity Measure** — specifies the measure that is used to determine the similarity between each time series. You can choose from **Squared Deviation**, **Absolute Deviation**, **Mean Squared Deviation**, and **Mean Absolute Deviation**.
- **Set Missing Value** — specifies how missing values are interpreted in the accumulated time series.

  You can choose from the following options:

  - **Missing**
  - **Zero**
  - **Average**
  - **Previous Value**
  - **Next Value**

- **Sequence Sliding** — specifies the sliding of the target sequence with respect to the input sequence.

  You can choose from the following options:

  - **None** — No sequence sliding is performed, and the input time series is compared directly to the target sequence.
  - **Index** — Index sequence sliding uses the time index variable, and the input time series is compared to the target sequence by observation index.
  - **Season** — Season sequence sliding uses the seasonal index variable, and the input time series is compared to the target sequence by seasonal index.

- **Normalization** — specifies the sequence normalization that is applied to the working target sequence.

  The following settings are available:

  - **None** — Normalization is not applied.
  - **Standard** — Standard normalization is applied.
  - **Absolute** — Absolute normalization is applied.

- **Scale** — specifies the scaling of the working input sequence with respect to the working target sequence. Scaling is performed after normalization.

  The following settings are available:

  - **None** — Normalization is not applied.
  - **Standard** — Standard normalization is applied.
  - **Absolute** — Absolute normalization is applied.

- **Compression Options** — Click the ... button to the right of the Compression Options property to open the Compression Options window and specify whether the target sequence will be compressed. Compression is performed on the target sequence with respect to the input sequence. Compression of the target sequence is identical to expansion of the input sequence, and vice versa.

  - **Compress** — specifies the sliding sequence (global) and warping (local) compression range of the target sequence with respect to the input sequence. The compression limits are defined based on the length of the target sequence and are imposed on the target sequence.

    If you select **None** or **Seasonal** for the Sequence Sliding property, then the global compression properties are ignored. To disallow local compression, set both **Local Absolute Compression** and **Local Compression Percent** to 0.

  - **Global Absolute Compression** — specifies the sliding sequence compression range of the target sequence. The value specified here must be an integer between 0 and 10,000, where 0 implies no compression.

  - **Global Compression Percent** — specifies the sliding sequence compression as a percentage of the length of the target sequence. The value specified here must be between 0 and 100, where 0 implies no compression and 100 implies the maximum allowable compression.
  - **Local Absolute Compression** — specifies the warping compression range of the target sequence. The value specified here must be an integer between 0 and 10,000, where 0 implies no compression.
  - **Local Compression Percent** — specifies the warping compression as a percentage of the length of the target sequence. The value specified here must be between 0 and 100, where 0 implies no compression and 100 implies the maximum allowable compression.

- **Expansion Options** — Click the ... button to the right of the Expansion Options property to open the Expansion Options window and specify whether the target sequence will be expanded. Expansion of the target sequence is the same as compression of the input sequence, and vice versa.
  - **Expansion** — specifies the sliding sequence (global) and warping (local) expansion range of the target sequence with respect to the input sequence. The expansion limits are defined based on the length of the target sequence and are imposed on the target sequence.

    If you select **None** or **Seasonal** for the Sequence Sliding property, then the global expansion properties are ignored. To disallow local expansion, set both **Local Absolute Compression** and **Local Compression Percent** to 0.
  - **Global Absolute Expansion** — specifies the sliding sequence expansion range of the target sequence. The value specified here must be an integer between 0 and 10,000, where 0 implies no expansion.
  - **Global Expansion Percent** — specifies the sliding sequence expansion as a percentage of the length of the target sequence. The value specified here must be between 0 and 100, where 0 implies no expansion and 100 implies the maximum allowable expansion.
  - **Local Absolute Expansion** — specifies the warping expansion range of the target sequence. The value specified here must be an integer between 0 and 10,000, where 0 implies no expansion.
  - **Local Expansion Percent** — specifies the warping expansion as a percentage of the length of the target sequence. The value specified here must be between 0 and 100, where 0 implies no expansion and 100 implies the maximum allowable expansion.

Time Series Similarity Train Properties: Clustering Options

- **Hierarchical Clustering** — specifies if the **TS Similarity** node will produce hierarchical clusters.

  You can select from the following options:
  - **Default** — Output varies based on the presence of a target variable. When the input data set contains a target variable, selecting **Default** does not perform hierarchical clustering. When the input data set does not contain a target variable, selecting **Default** does perform hierarchical clustering.
  - **Yes** — Always creates hierarchical clusters.
  - **No** — Never creates hierarchical clusters.
- **Number of Clusters** — specifies the number of clusters created.

- **Include Targets** — specifies if the exported distance matrix will contain target variables.

Time Series Similarity Report Properties

- **Similarity Plot Maximum** — specifies the maximum number of time series similarity plots.
- **Preference of Similarity Plot** — specifies which series is plotted with the target series. You can choose either the **Most Similar** series or the **Least Similar** series.
- **Output Data Set** — specifies which type of data is exported to the successor node. Select **Distance Matrix** to export the distance matrix. Select **Clustering Segment** to export the time series clustering segments and to perform hierarchical clustering. Note that hierarchical clustering is performed only if the conditions described in the **Hierarchical Clustering** property are met. If you specify **Default**, then the time sequences that are used to compute the similarity measures are exported.

Time Series Similarity Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Time Series Similarity Results

You can open the Results window of the **TS Similarity** node after a successful run by clicking **Results** in the Run Status window, or by right-clicking the node in the diagram workspace and selecting **Results** from the pop-up menu. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **TS Similarity** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **TS Similarity** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.

- **SAS Results**
  - **Log** — the SAS log of the **TS Similarity** node run.
  - **Output** — the SAS output of the **TS Similarity** node run.
  - **Flow Code** — the SAS code used to produce the output that the **Time Series** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **TS Similarity** node does not generate PMML code.
- **Model**
  - **TSID Map Table** — displays a table that shows the input variable grouping that is associated with each time series ID.
- **Plots**
  - **Distance Map** — displays a grid with all the time series IDs on both axes and the distance between each ID to every other ID in the grid. A distance of 0 denotes that the time series are identical, as evidenced by the diagonal of this graph.

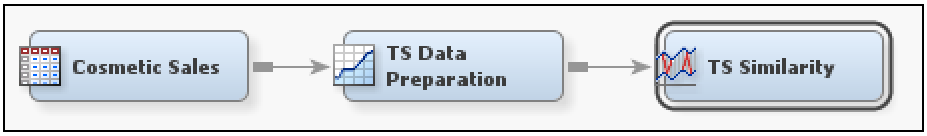
    Note: The distance map is not created when the number of cells in the map exceeds 10000.
- **Table** — displays a table that contains the underlying data used to produce a chart. The **Table** feature is dimmed and unavailable in the menu unless a results chart is open and selected.
- **Plot** — opens the Graph wizard to modify an existing Results plot or create a Results plot of your own. The Plot feature is dimmed and unavailable in the menu unless a Results chart or table is open and selected.

Time Series Similarity Node Examples

Example 1: Similarity Analysis without a Target Variable

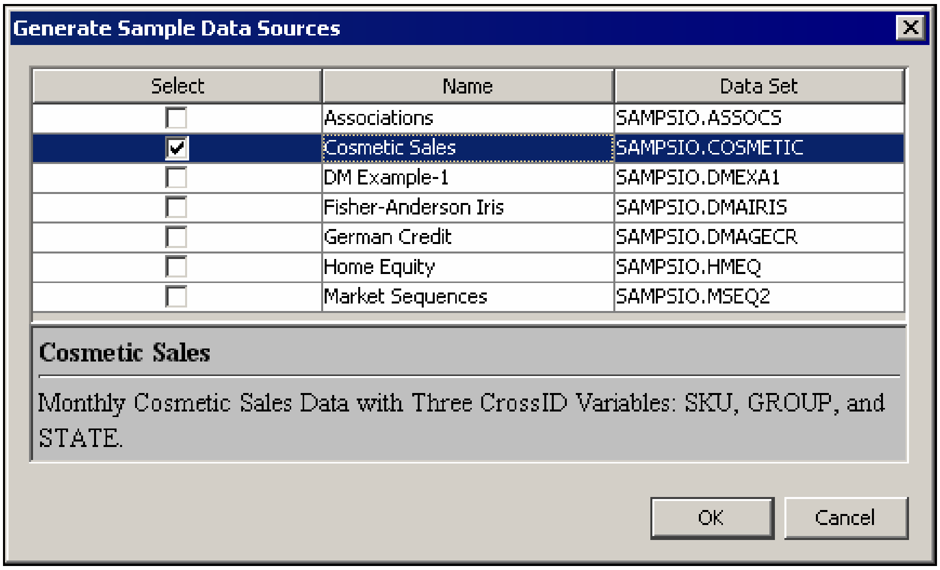
This example illustrates how to perform a time series similarity analysis without a specified target variable. In essence, similarity analysis without a target variable becomes a clustering problem. If you do not specify any target variable, the **TS Similarity** node generates a distance matrix among all input time series and performs basic hierarchical clustering. The Output Data Set property can be used to export either the distance matrix or the clustering segments.

Your completed process flow diagram will resemble the one shown below.



This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the menu bar. In the Generate Sample Data Sources window, select only the Cosmetic Sales data set, as shown below.



Generate Sample Data Sources

Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

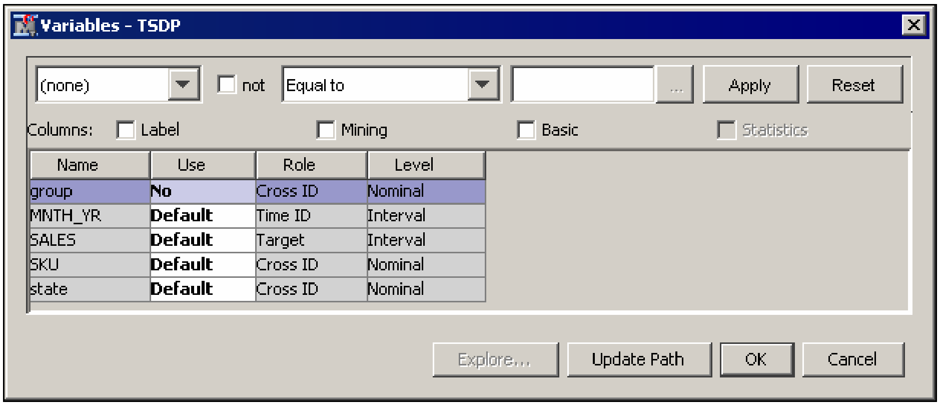
Cosmetic Sales

Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

2. Next, drag the Cosmetic Sales data set into your diagram workspace.

   From the **Time Series** tab, select the **TS Data Preparation** node and drag it into your diagram workspace. Connect the Cosmetic Sales data set to the **TS Data Preparation** node.

3. Right-click the **TS Data Preparation** node and select **Edit Variables** from the menu. This opens the Variables window, where you can set the use status for each variable. For this example, you want to set the use status for the group variable to **No**, as shown below.



Variables - TSDP

(none) □ not Equal to

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Use	Role	Level
group	No	Cross ID	Nominal
MNTH_YR	Default	Time ID	Interval
SALES	Default	Target	Interval
SKU	Default	Cross ID	Nominal
state	Default	Cross ID	Nominal

   Notice that there are three Cross ID variables for the Cosmetic Sales data set. The **TS Data Preparation** node creates a separate time series for each unique combination of the values of the Cross ID variable.

   In the Cosmetic Sales data set, the SKU variable has 5 values, the **state** variable has 5 values, and the group variable has 3 values. If you used all three Cross ID variables, then you would have 75 different time series in this example. After excluding the group variable, there are 25 time series in the similarity analysis.

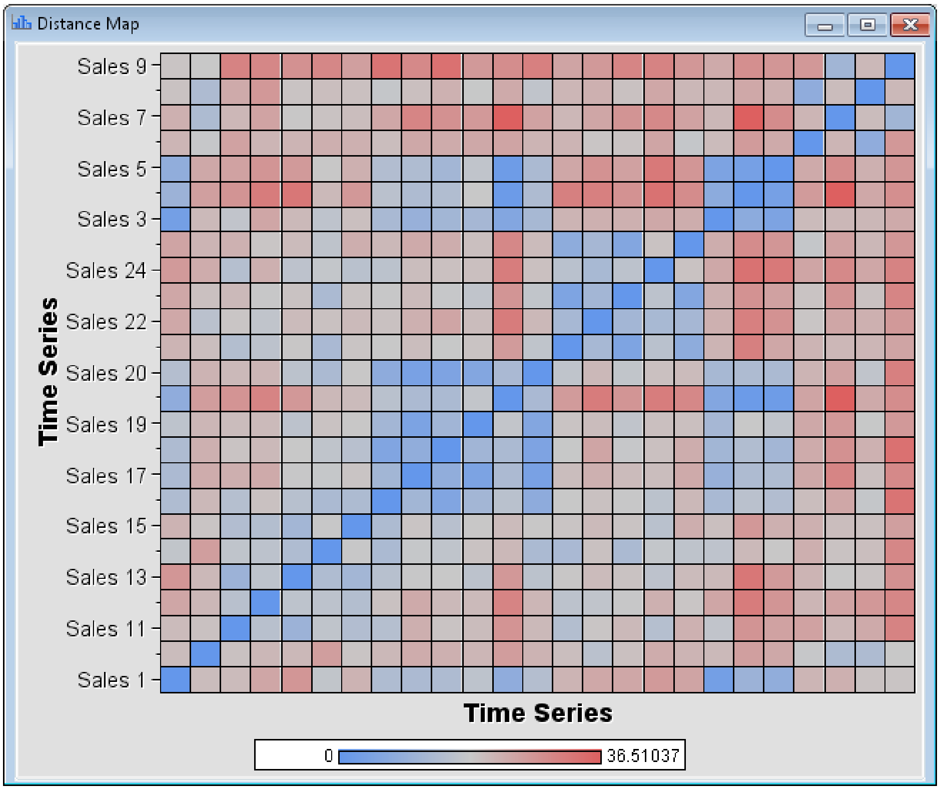
4. Select the Time Series Data Preparation node, and change the value of the Transpose property to **Yes**. Change the value of the Keep Variable Role property to **No**.

5. From the **Time Series** tab, select the **TS Similarity** node and drag it into your diagram workspace. Connect the **TS Data Preparation** node to the **TS Similarity** node.

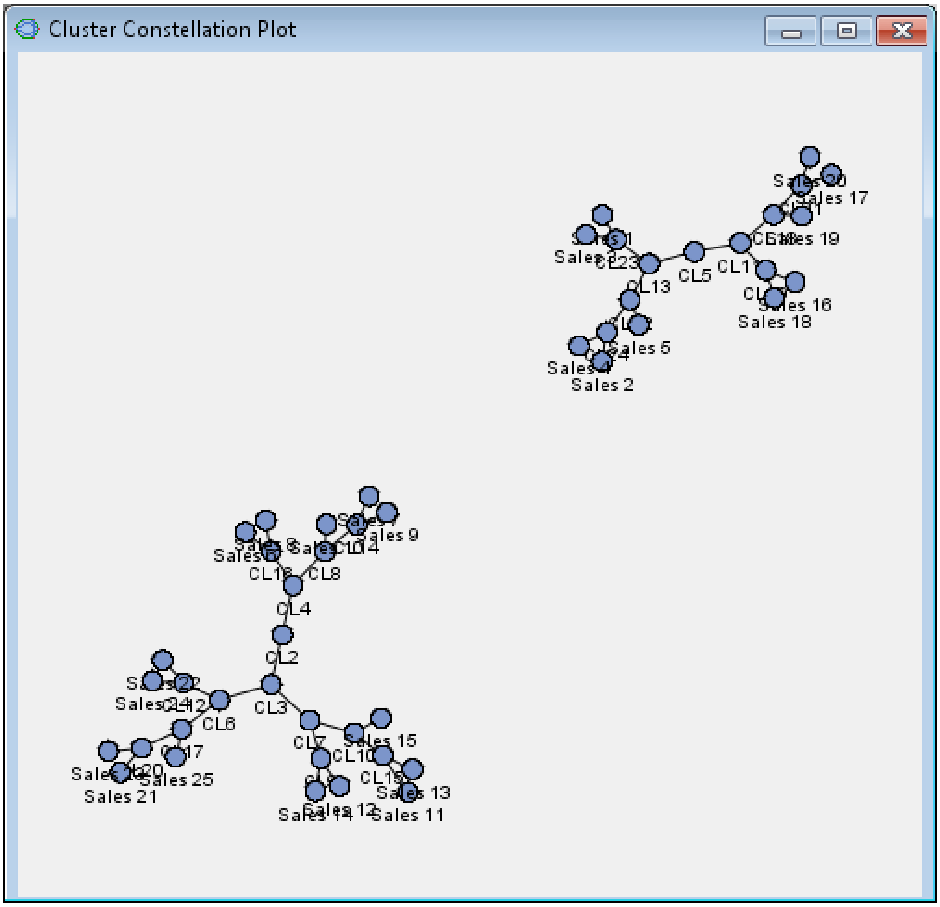
   This example uses the default properties for the **TS Similarity** node, so right-click the **TS Similarity** node and click **Run** on the pop-up menu. In the Confirmation window, click **Yes**.

6. After a successful run of the **TS Similarity** node, click **Results** in the Run Status window.

   The first result that you should notice is the Distance Map. Maximize the Distance Map window. The Distance Map lists the time series IDs along both axes and provides a visual display of how similar one time series is to every other time series. Blue indicates that the time series are similar and red indicates that they are dissimilar.

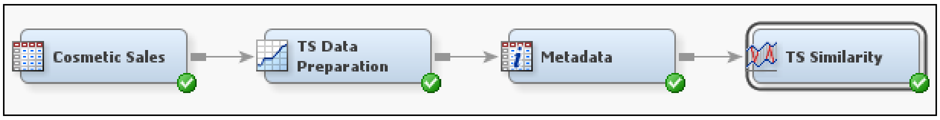
   Next, minimize the Distance Map and maximize the Cluster Constellation Plot window. This window shows a constellation plot of the identified clusters and their constituent time series IDs. Two clusters of the time series are identified in this example, as shown below.



Example 2: Similarity Analysis with Target Variables

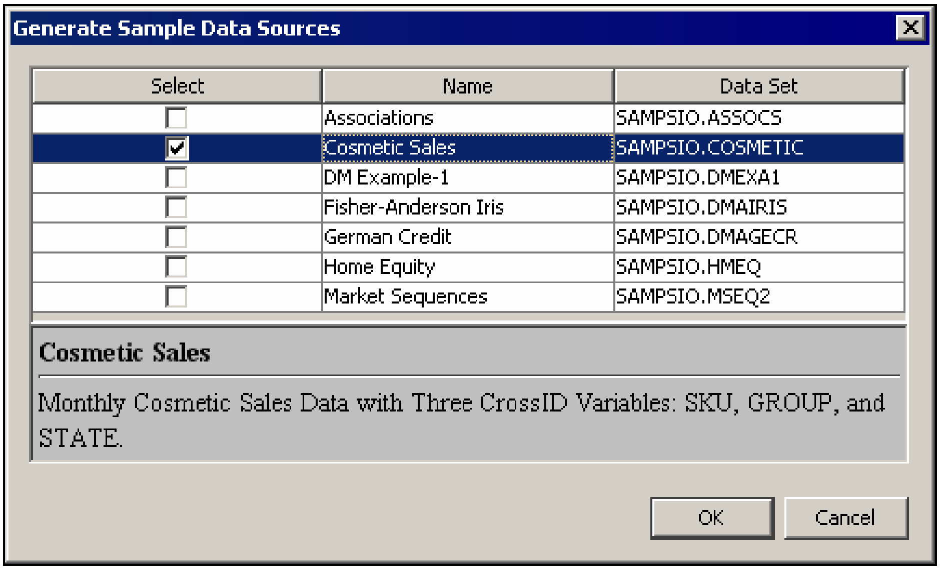
The **TS Similarity** node can be used to identify the most and least similar time series to one or more reference time series. This example uses the same data set as the previous example. However, you need to use the **Metadata** node to designate two time series as targets.

The completed process flow diagram is shown below.



This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the menu bar. In the Generate Sample Data Sources window, select only the Cosmetic Sales data set, as shown below.



Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Cosmetic Sales

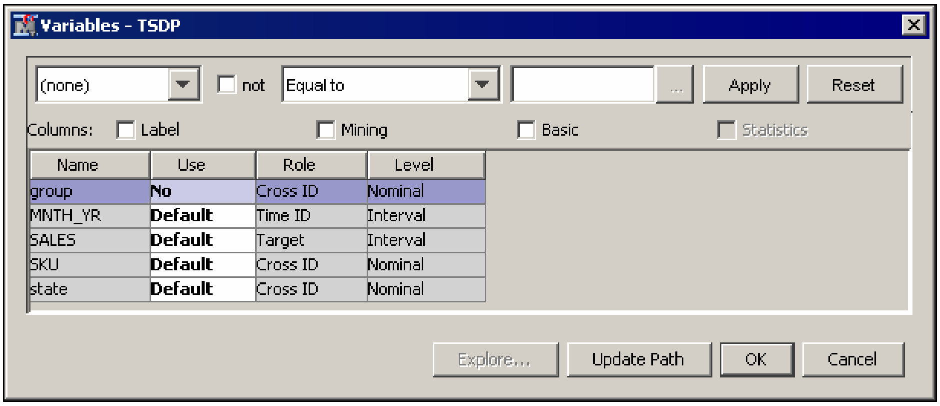
Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

Click **OK**.

2. Next, drag the Cosmetic Sales data set into your diagram workspace.

   From the **Time Series** tab, select the **TS Data Preparation** node and drag it into your diagram workspace. Connect the Cosmetic Sales data set to the **TS Data Preparation** node.

3. Right-click the **TS Data Preparation** node and click **Edit Variables** on the menu. This opens the Variables window, where you can set the use status for each variable. For this example, you want to set the use status for the group variable to **No**, as shown below.



Name	Use	Role	Level
group	No	Cross ID	Nominal
MNTH_YR	Default	Time ID	Interval
SALES	Default	Target	Interval
SKU	Default	Cross ID	Nominal
state	Default	Cross ID	Nominal

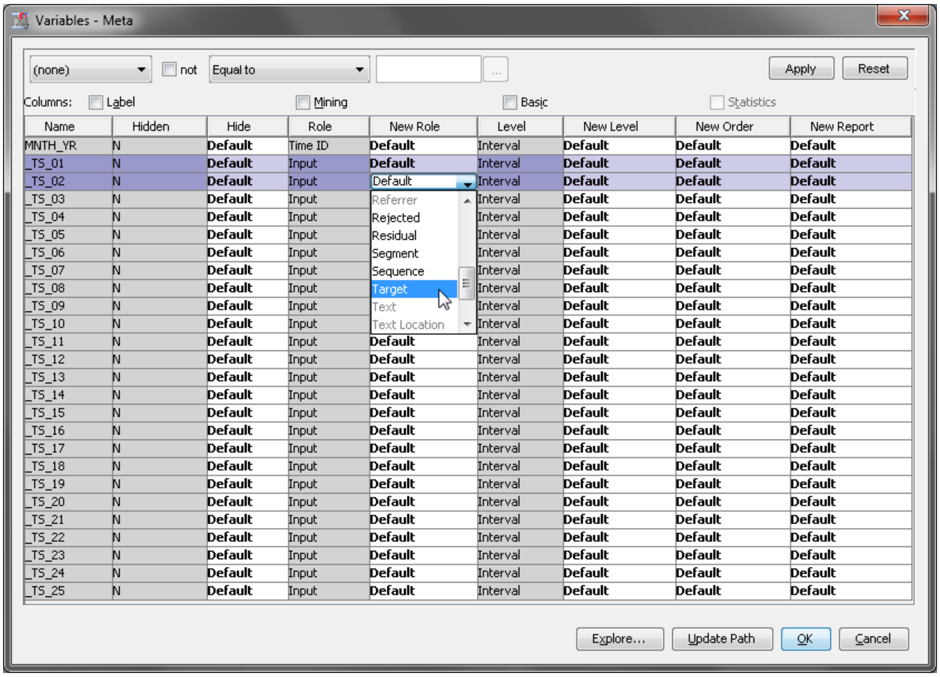
Notice that there are three Cross ID variables for the Cosmetic Sales data set. The **TS Data Preparation** node creates a separate time series for each unique combination of variable values for each Cross ID variable.

In the Cosmetic Sales data set, the SKU variable has 5 values, the state variable has 5 values, and the group variable has 3 values. If you used all three Cross ID variables, then you would have 75 different time series in this example. After excluding the group variable, there are 25 time series in the similarity analysis.

4. Select the Time Series Data Preparation node, and change the value of the Transpose property to **Yes**. Change the value of the Keep Variable Role property to **No**.

5. From the **Utility** tab, select the **Metadata** node and drag it into the diagram workspace. Connect the **TS Data Preparation** node to the **Metadata** node. For this example, you will set the Role of _TS_01 and _TS_02 to **Target**.

In the Variables subgroup of the Train properties group, click the ... for the Transaction property. Select both _TS_01 and _TS_02 by holding down the Ctrl key and clicking both variables. Click anywhere in the New Role column of those two variables and click **Target**, as shown below.



Name	Hidden	Hide	Role	New Role	Level	New Level	New Order	New Report
MNTH_YR	N	Default	Time ID	Default	Interval	Default	Default	Default
_TS_01	N	Default	Input	Default	Interval	Default	Default	Default
_TS_02	N	Default	Input	Default	Interval	Default	Default	Default
_TS_03	N	Default	Input		Interval	Default	Default	Default
_TS_04	N	Default	Input		Interval	Default	Default	Default
_TS_05	N	Default	Input		Interval	Default	Default	Default
_TS_06	N	Default	Input		Interval	Default	Default	Default
_TS_07	N	Default	Input		Interval	Default	Default	Default
_TS_08	N	Default	Input		Interval	Default	Default	Default
_TS_09	N	Default	Input		Interval	Default	Default	Default
_TS_10	N	Default	Input		Interval	Default	Default	Default
_TS_11	N	Default	Input	Default	Interval	Default	Default	Default
_TS_12	N	Default	Input	Default	Interval	Default	Default	Default
_TS_13	N	Default	Input	Default	Interval	Default	Default	Default
_TS_14	N	Default	Input	Default	Interval	Default	Default	Default
_TS_15	N	Default	Input	Default	Interval	Default	Default	Default
_TS_16	N	Default	Input	Default	Interval	Default	Default	Default
_TS_17	N	Default	Input	Default	Interval	Default	Default	Default
_TS_18	N	Default	Input	Default	Interval	Default	Default	Default
_TS_19	N	Default	Input	Default	Interval	Default	Default	Default
_TS_20	N	Default	Input	Default	Interval	Default	Default	Default
_TS_21	N	Default	Input	Default	Interval	Default	Default	Default
_TS_22	N	Default	Input	Default	Interval	Default	Default	Default
_TS_23	N	Default	Input	Default	Interval	Default	Default	Default
_TS_24	N	Default	Input	Default	Interval	Default	Default	Default
_TS_25	N	Default	Input	Default	Interval	Default	Default	Default

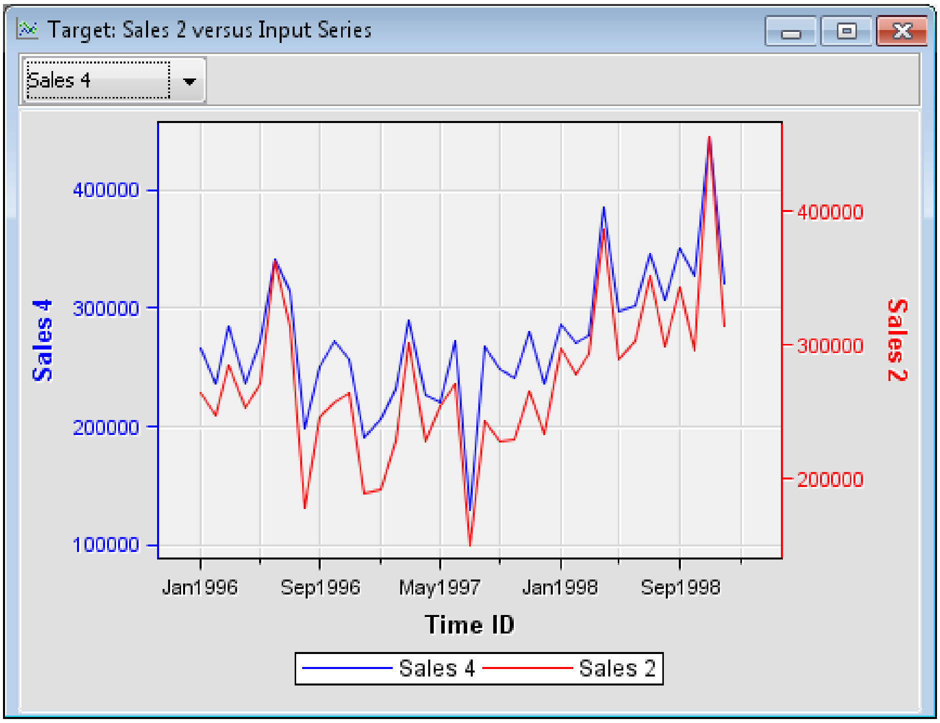
6. From the **Time Series** tab, select the **TS Similarity** node and drag it into your diagram workspace. Connect the **Metadata** node to the **TS Similarity** node.

   This example uses the default properties for the **TS Similarity** node, so right-click the **TS Similarity** node and click **Run** on the pop-up menu. In the Confirmation window, click **Yes**.

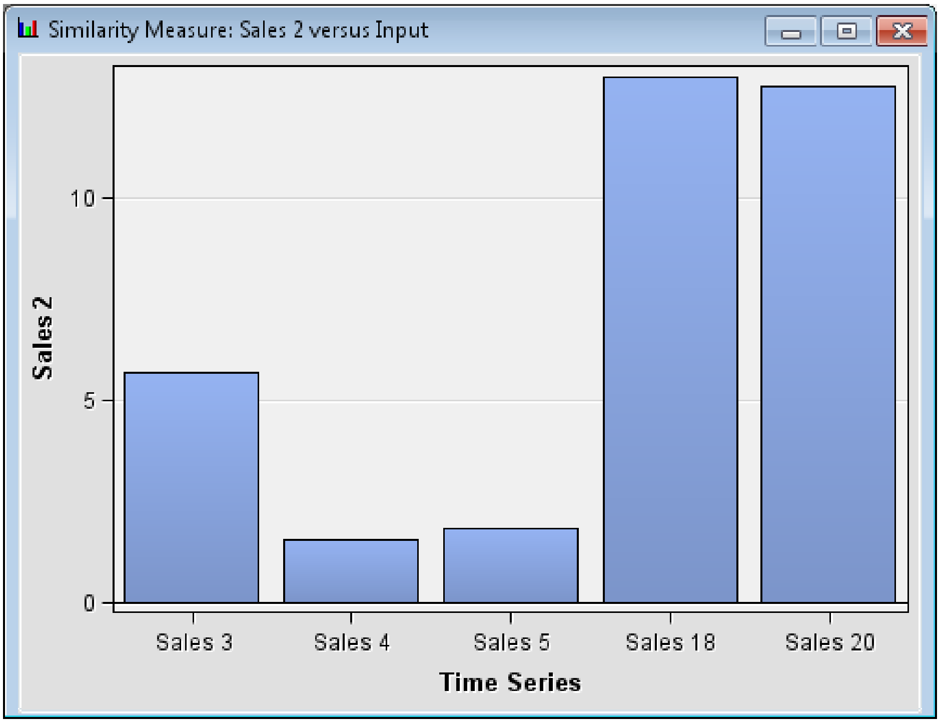
7. After a successful run of the Time Series Similarity node, click **Results** in the Run Status window. Notice that the output generated from this run of the Time Series Similarity node is different from that generated in the previous example. The distance matrix contains rows for only _TS_01 and _TS_02, but compares them to every other time series.

   There are also two graphics present here that were not present in the previous example. The first is a plot of the target series against the input series. The second is a bar chart comparing the similarity measure of each input time series to the target time series.

   By default, the plot of the target series against the input series generates a graph for the five most similar time series. You can select which of those five you want to view using the drop-down menu in the upper left corner of the window.



Also, those five time series are the same time series that are shown in the bar chart.



To set how many input series will be displayed, use the Similarity Plot Maximum property in the Report properties group. Close the Results window.

8. Select the **TS Similarity** node and find the Preference of Similarity Plot property in the Report properties group. Change the value for this property to **Least Similar**. Rerun the **TS Similarity** node. Now, the **TS Similarity** node searches for the five time series that are least similar to the _TS_01 and _TS_02.

When the **TS Similarity** node has successfully run, click **Results** in the **Run Status** window. Notice now that the plots of the input series and the target series rarely coincide and appear substantially different. Close the Results window.

Chapter 83

Time Series Exponential Smoothing Node

Time Series Exponential Smoothing Node

Overview of the Time Series Exponential Smoothing Node

The **TS Exponential Smoothing** node generates forecasts and some outputs that are useful for data mining. The node uses exponential smoothing models that have optimized smoothing weights for time series data and transaction data.

The **TS Exponential Smoothing** node offers the following forecasting models:

- Simple (single) exponential smoothing
- Double (Brown) exponential smoothing
- Linear (Holt) exponential smoothing
- Damped trend exponential smoothing
- Additive seasonal exponential smoothing
- Multiplicative seasonal exponential smoothing
- Winters multiplicative method
- Winters additive method

For more details about these methods, please see the SAS online Help for the SAS/ETS procedure PROC ESM. The **TS Exponential Smoothing** node also provides modelers with the ability to detect and replace outliers, to export some distance matrices, and to extend input time series to the future values.

Time Series Exponential Smoothing Node Data Set Requirements

The **TS Exponential Smoothing** node does not require a preceding **TS Data Preparation** node.

The **TS Exponential Smoothing** node requires that the following roles be assigned to variables in the **Input Data Source** node:

- **TimeID** — You must define a single numeric interval time ID variable. The time ID provides either timestamp information or sequential information. When the time ID variable does not contain a valid SAS date, time, or datetime format, the values of the time ID variable are interpreted as sequential numbers.
- **CrossID** (optional) — You can define any number of optional cross ID variables. A cross ID variable must have a non-interval measurement. A cross ID variable represents a cross-sectional dimension to the time series data.

Note that when a CrossID is available, the **TS Exponential Smoothing** node analyzes only one target variable each run.

Time Series Exponential Smoothing Node Properties

Time Series Exponential Smoothing General Properties

The following general properties are associated with the **TS Exponential Smoothing** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **TS Exponential Smoothing** node that is added to a diagram has a Node ID of TSEM. The second **TS Exponential Smoothing** node added to a diagram has a Node ID of TSEM2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Time Series Exponential Smoothing window. The Imported Data — Time Series Exponential Smoothing window contains a list of the ports that provide data sources to the **TS Exponential Smoothing** node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Time Series Exponential Smoothing window. The Exported Data — Time Series Exponential Smoothing window contains a list of the output data ports that the **TS Exponential Smoothing** node creates data for when it runs. Click the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Time Series Exponential Smoothing Train Properties

The following train properties are associated with the **TS Exponential Smoothing** node:

- **Variables** — Use the Variables property of the **TS Exponential Smoothing** node to specify the properties of each variable that you want to use in your data source. Click the ... button to the right of the Variables property to open a variables table. You can set the variable status to either **Use** or **Don't Use** in the table, and you can select which variables you want included in reports.
- **Specify an Interval** — specifies a valid SAS time interval, which is the frequency of the accumulated time series. The default **Automatic** setting enables the SAS Enterprise Miner software to heuristically detect and configure the time series interval. You can manually configure your time interval. For example, if the input data set consists of quarterly observations, then you can specify **Quarter** as your Specify an Interval value. Alternatively, you can specify **Year** as your Specify an Interval value to output yearly accumulated time series. For more information, see the **Accumulation** property. The following interval values are available for selection:

  - **Automatic** — The interval length is determined from the data.
  - **Year** — Yearly
  - **Semiyear** — Semiannual
  - **Quarter** — Quarterly
  - **Month** — Monthly
  - **Semimonth** — 1st and 16th of each month
  - **Ten Days** — 1st, 11th, and 21st of each month
  - **Week** — Weekly
  - **Weekday** — Daily, ignoring weekend days
  - **Day** — Daily
  - **Hour** — Hourly
  - **Minute** — Every minute
  - **Second** — Every second

  Note the following conditions when selecting your time interval:

  - When the time interval variable does not contain a valid SAS date, time, or datetime format, the time interval variable is interpreted as a sequential variable. Thus, there is no automatic detection of the time interval is not available.

  - When your time ID variable uses a valid SAS date format, you cannot specify a time interval shorter than **Day**.
  - When your time ID variable uses a valid SAS time format, you cannot specify a time interval longer than **Hour**.
  - Automatic time interval detection is not supported for a time ID variable with a valid SAS time format. You must specify either **Hour**, **Minute**, or **Second**.
- **Accumulation** — specifies how the data set observations are accumulated within each time interval. The time interval is specified by the **Specify an Interval** property in the **TS Data Preparation** node. The **Accumulation** property is particularly useful when there are gaps in the input data or when there are multiple input observations that coincide with a particular time period.

  The accumulation methods available are as follows:

  - **Total** — Observations are accumulated based on the total sum of the values in a time interval.
  - **Average** — Observations are accumulated based on the average of the values in a time interval.
  - **Minimum** — Observations are accumulated based on the minimum value of a time interval.
  - **Median** — Observations are accumulated based on the median value of a time interval.
  - **Maximum** — Observations are accumulated based on the maximum value of a time interval.
  - **First** — Observations are accumulated based on the first value in a time interval.
  - **Last** — Observations are accumulated based on the last value in a time interval.
- **Seasonality** — specifies the length of the seasonal cycle in terms of the number of observations per cycle. For example, if you set the value of **Seasonality** to **3**, then each group of three observations forms a seasonal cycle. If you click **Default**, then either no seasonality is used or the seasonality implied by the time interval is used. That is, if the Specify an Interval property in the **TS Data Preparation** node is set to **Month**, then a value of 12 is implied for **Seasonality**. This option is applicable only for seasonal forecasting models.
- **Forecasting Method** — specifies the forecasting model that is used.
  - **Best** — chooses the best model based on the criteria specified in the Best Model Selection property group.
  - **Simple** — uses single exponential smoothing.
  - **Double** — uses double exponential smoothing, also called Brown exponential smoothing.
  - **Linear** — uses linear exponential smoothing, also called Holt exponential smoothing.
  - **Damped Trend** — uses damped trend exponential smoothing.
  - **Additive Seasonal** — uses additive seasonal exponential smoothing.
  - **Additive Winters** — uses the Winters additive method.
  - **Multiplicative Seasonal** — uses multiplicative seasonal exponential smoothing.
  - **Multiplicative Winters** — uses the Winters multiplicative method.

- **Forecast Lead** — specifies the number of periods ahead to forecast. This value is relative to the Forecast Back value and the last observation in the input or accumulated time series, and not the last nonmissing observation. Thus, if a series has missing values at the end, then the actual number of forecasts that are computed is greater than this value.
- **Forecast Back** — specifies the number of observations before the end of the data where the multistep forecasts are to begin.
- **Forecast Sum Start** — specifies the starting forecast lead (or horizon) that is used to begin the summation of the forecasts specified by the Forecast Lead property. The Forecast Sum Start value must be less than the value specified in **Forecast Lead**. That is, if **Forecast Sum Start** is set to **2** and **Forecast Lead** is set to **6**, the forecast summation will include the second forecast term through the sixth forecast term.
- **Significance Level** — specifies the significance level that is used to compute the confidence limits of the forecast. The value specified here must be between 0 and 1.

Time Series Exponential Smoothing Train Properties: Input Time Series

- **Forecast Input Time Series** — exports the data with forecast values only for the input variables. This setting is useful for time series regression models and enables you to predict one time series based on the other time series.
- **Extended Value** — specifies which forecast values are appended to the actual data. This property is available only if you select **Yes** for the Forecast Input Time Series property.
  - **Predicted Value** — appends the predicted values to the exported data set.
  - **Lower CI Value** — appends the lower confidence limit values to the exported data set.
  - **Upper CI Value** — appends the upper confidence limit values to the exported data set.

Time Series Exponential Smoothing Train Properties: Best Model Selection

- **Selection Criterion** — specifies the model selection criterion that is used to determine the best forecasting model.

  The selection criteria are as follows:

  - **Mean Square Error**
  - **Sum of Square Error**
  - **Schwarz's Bayesian Information Criterion**
  - **Root Mean Square Error**
  - **R-Square**
  - **Maximum Symmetric Percent Error**
  - **Median Absolute Percent Error**
  - **Median Absolute Error Percent of Standard Deviation**
  - **Median Absolute Predicted Percent Error**
  - **Median Absolute Predicted Percent Error**
  - **Median Absolute Symmetric Percent Error**

  - **Median Relative Absolute Error**
  - **Mean Error**
  - **Minimum Absolute Error Percent of Standard Deviation**
  - **Minimum Error**
  - **Minimum Percent Error**
  - **Minimum Predicted Percent Error**
  - **Minimum Relative Error**
  - **Minimum Symmetric Percent Error**
  - **Mean Percent Error**
  - **Mean Predicted Percent Error**
  - **Mean Relative Absolute Error**
  - **Mean Relative Error**
  - **Mean Symmetric Percent Error**
  - **Random Walk R-Square**
  - **Mean Absolute Symmetric Percent Error**
  - **Unbiased Mean Square Error**
  - **Unbiased Root Mean Square Error**

- **Model Candidates** — Click the ... button to the right of the Model Candidates property to open the Model Candidates window. The Model Candidates window enables you to select which models to compare. The **TS Exponential Smoothing** node compares only the models that you have selected **Yes** for.

  Furthermore, if the chosen time interval has no implicit seasonality, then the seasonal models are not compared. For example, the yearly time interval has no seasonality. Thus, even if you select **Yes** for the **Additive Seasonal Model**, it will not be used.

  The available models are as follows:

  - **Simple Exponential Smoothing**
  - **Double Exponential Smoothing**
  - **Linear Exponential Smoothing**
  - **Damped Trend Exponential Smoothing**
  - **Additive Seasonal Model**
  - **Multiplicative Seasonal Model**
  - **Additive Winters Method**
  - **Multiplicative Winters Method**

Time Series Exponential Smoothing Train Properties: Outliers in Exported Data

- **Smooth Outliers** — click **Yes** to detect any outliers, replace those values, and export the smoothed data set. Click **No** to leave outliers in the data set.
- **Outlier Replacement** — determines how outliers are replaced. If you select **Predicted Value**, then the value of the outlier is replaced with the predicted value

from its forecasting model. If you select **Missing**, then the value of the outlier is left as missing values in the exported data.

Time Series Exponential Smoothing Train Properties: Output Options

- **Output Data Type** — specifies which data is exported to any successor nodes.
  - **Default** — exports the time series data and the forecast values, including target variables.
  - **Kullback-Leibler Distance** — exports a Kullback-Leibler distance matrix among forecast distributions at a specific future time point or over the sum of the forecasting lead points. The specific point is set in the Clustering Lead Point property, which is described below.
  - **Coordinated Forecast Data** — exports a data set that contains only forecast values to be used for clustering time series based on forecasts.
  - **Similarity Input Data** — exports a data set that contains only forecast values like Coordinated Forecast Data, but it is formatted for input to the **TS Similarity** node.
- **Lead Point for KLD** — specifies a clustering time point within the forecasting lead that is used to compute the Kullback-Leibler distance among forecasts. Set this value to 0 to calculate the Kullback-Leibler distance among forecasts summed over all the forecasting lead points. This property is available only if you select **Kullback-Leibler Distance** for the Output Data property.

Time Series Exponential Smoothing Report Properties

The following report property is associated with the **TS Exponential Smoothing** node:

- **Plot Length** — specifies the length of the time series plot. The default value of 3 means that if the total number of observations exceeds 20,000, then the plot displays three times the forecasting lead, with a minimum of six data points.

Time Series Exponential Smoothing Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Time Series Exponential Smoothing Results

You can open the Results window of the **TS Exponential Smoothing** node after a successful run by clicking **Results** in the Run Status window, or by right-clicking the node in the diagram workspace and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the **TS Exponential Smoothing** node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the **TS Exponential Smoothing** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the **TS Exponential Smoothing** node run.
  - **Output** — the SAS output of the **TS Exponential Smoothing** node run.
  - **Flow Code** — the SAS code that is used to produce the output that the **TS Exponential Smoothing** node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the **TS Exponential Smoothing** node does not generate PMML code.
- **Model**
  - **Parameter Estimate** — displays a table of the parameter estimates that were computed for each time series ID. This table indicates the variable name, transformation applied, forecasting model used, parameter name, and various parameter estimates.
  - **Fit Statistics** — displays a table of the fit statistics that were computed for each time series ID.
  - **TSID Map Table** — displays a table that shows the input variable grouping that is associated with each time series ID.
  - **Outlier Table** — displays a table of the outlier observations for each time series ID.
  - **Time Series Metadata Table** — displays a table that contains metadata information about the TimeID variable.
- **Plots**
  - **Forecast Summation Histogram** — displays a histogram distribution of forecast, forecast upper bound, and forecast lower bound summation.
  - **Forecast Histogram** — displays a histogram distribution of forecasts at each step ahead.

  - **Multiple Forecast Comparison Plot** — displays a single plot that contains all forecast series over the specified future time points.
  - **Forecast Comparison Plot** — displays a single forecast series so that you can observe the forecast trend.
  - **Forecast Summary** — displays plots of forecast summary statistics for each time series.
- **Forecasting Plots** — A menu item is generated for each time series ID and enables you to view the original time series and the forecasting values for each time series ID.
- **Table** — displays a table that contains the underlying data used to produce a chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph wizard to modify an existing Results plot or create a Results plot of your own. The **Plot** menu item is dimmed and unavailable unless a Results chart or table is open and selected.

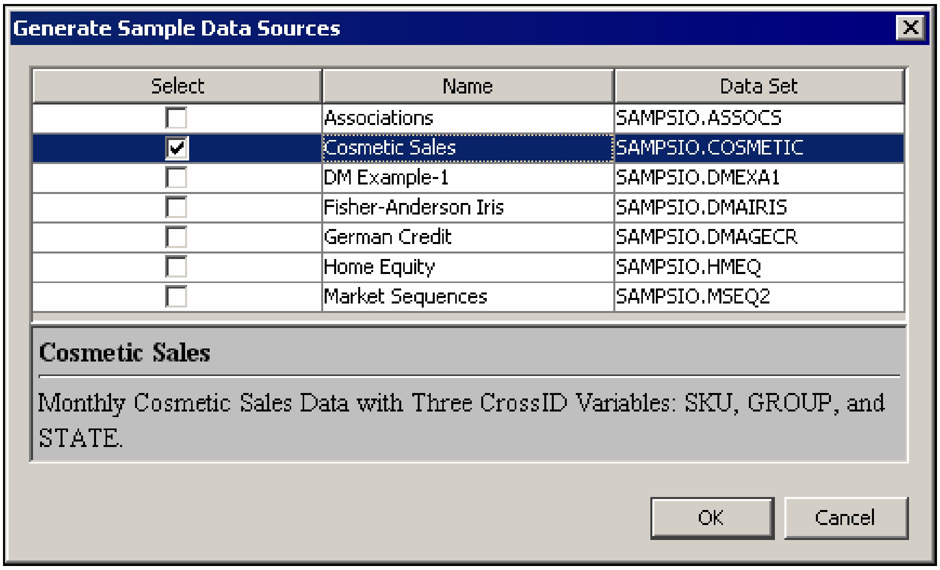
Time Series Exponential Smoothing Node Examples

Example 1: Using the Time Series Exponential Smoothing Node

This example illustrates many of the core features of the **TS Exponential Smoothing** node. This example uses the Cosmetic Sales data set in the SAMPSIO library.

This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the menu bar. In the Generate Sample Data Sources window, select only the Cosmetic Sales data set, as shown below.



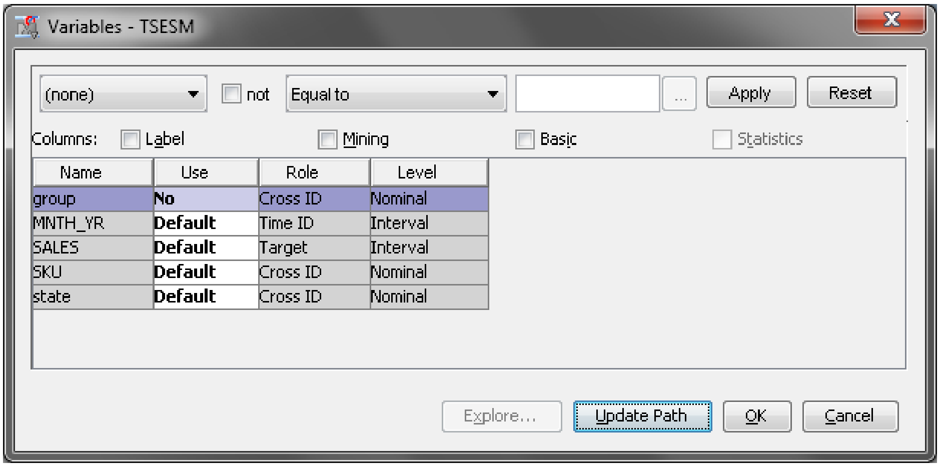
Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Cosmetic Sales

Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

2. Next, drag the Cosmetic Sales data set into your diagram workspace. From the **Time Series** tab, select the **TS Exponential Smoothing** node and drag it into your diagram workspace. Connect the Cosmetic Sales data set to the **TS Exponential Smoothing** node.

3. Right-click the **TS Exponential Smoothing** node and select **Edit Variables** from the menu. This opens the Variables window, where you can set the use status for each variable. For this example, you want to set the use status for the group variable to **No**, as shown below.



Variables - TSESM

(none) □ not Equal to Apply Reset

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Use	Role	Level
group	No	Cross ID	Nominal
MNTH_YR	Default	Time ID	Interval
SALES	Default	Target	Interval
SKU	Default	Cross ID	Nominal
state	Default	Cross ID	Nominal

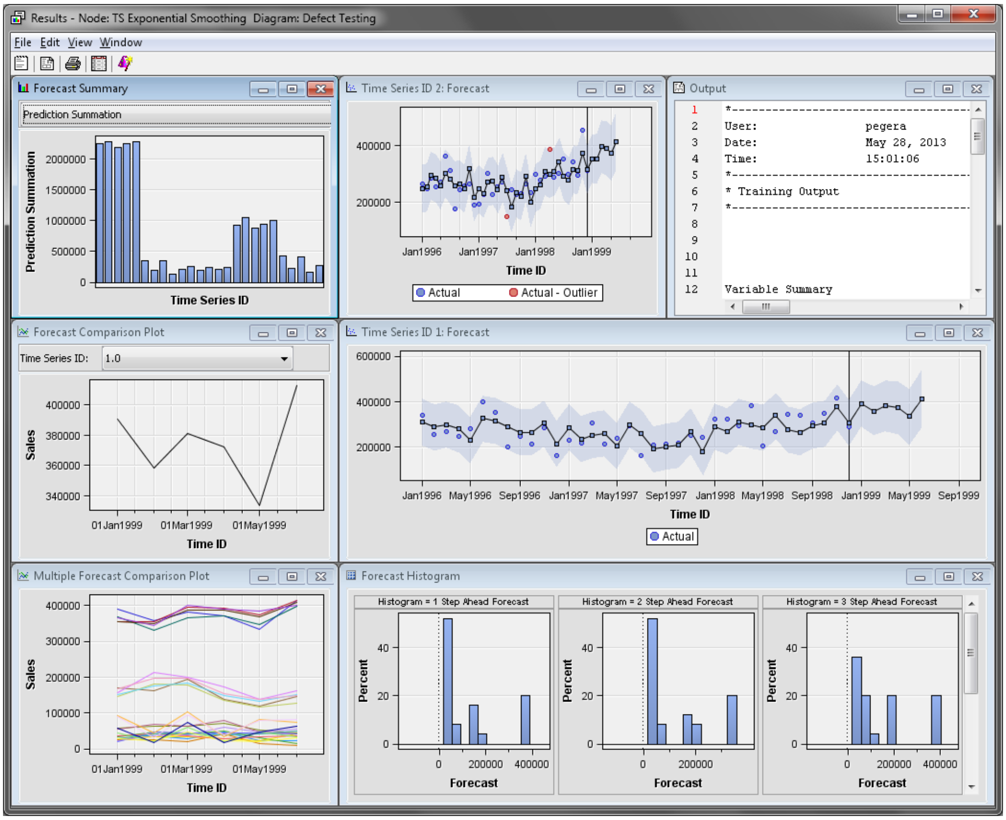
Notice that there are three Cross ID variables for the Cosmetic Sales data set. The **TS Data Preparation** node creates a separate time series for each unique combination of variable values for each Cross ID variable.

In the Cosmetic Sales data set, the SKU variable has 5 values, the state variable has 5 values, and the group variable has 3 values. If you used all three Cross ID variables, then you would have 75 different time series in this example. After excluding the group variable, there are 25 time series in the similarity analysis.

4. The default node properties for the **TS Exponential Smoothing** node are adequate for this example.

   Right-click the **TS Exponential Smoothing** node and click **Run** on the menu. In the Confirmation window, click **Yes** to run the process flow diagram.

5. After the process flow diagram has run successfully, click **Results** in the Run Status window. This opens the Results window.

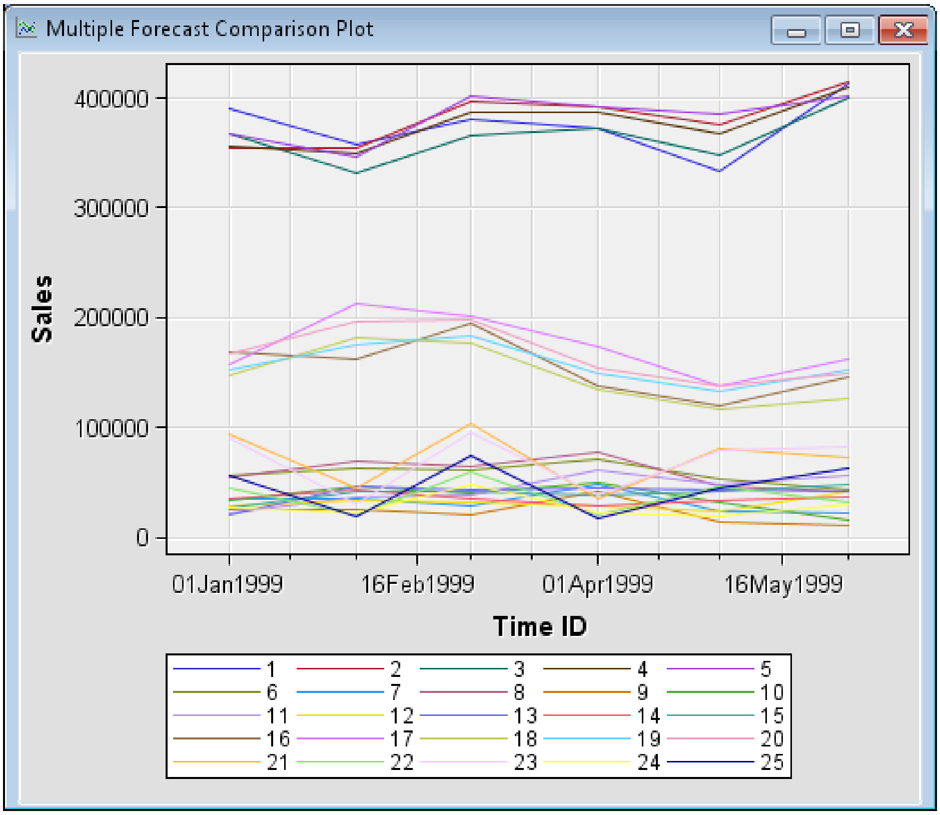


Time Series ID: 1.0

```
*------------------------------------------
User:                   pegera
Date:                   May 28, 2013
Time:                   15:01:06
*------------------------------------------
* Training Output
*------------------------------------------




Variable Summary
```

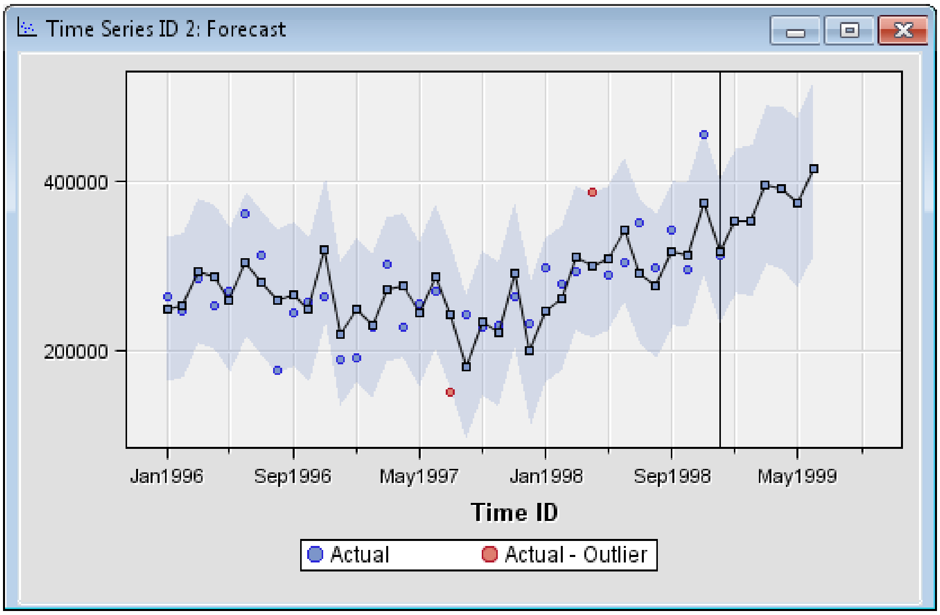
First, notice the Forecast Summary window. The drop-down menu enables you to compare various summary statistics across time series. For example, if you select **Minimum Value of Series** from the drop-down menu, then the Forecast Summary window displays the minimum value for each time series.

The Multiple Forecast Comparison Plot window displays plots of the forecast values for all time series at the future time points. This graph can help you obtain an approximate idea about the clustering of the forecast time series.



The Forecast Histogram window contains a histogram of the forecast distribution for each forecast time step.

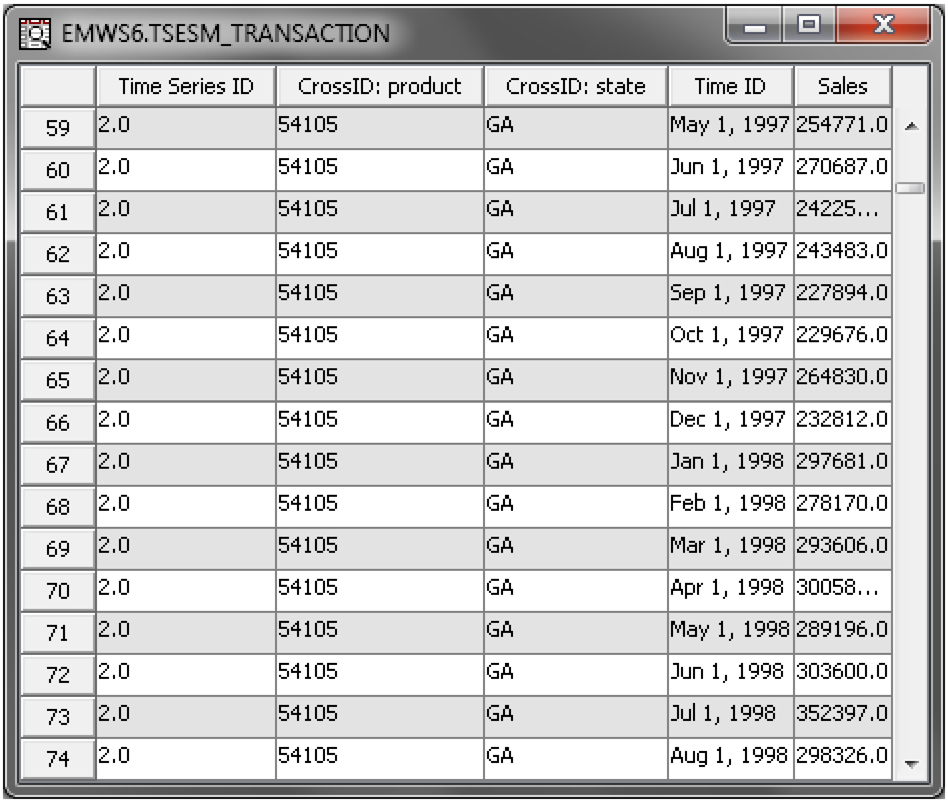
The Time Series ID n: Forecast windows display a detailed display of each time series with actual values, forecast values, and forecast value confidence intervals. The graph displays actual values that fall outside of the forecast value confidence intervals as outliers.



In the Time Series ID window, the square points indicate the predicted time series, and the circular points indicate the actual time series data. The shaded region indicates the confidence interval (determined by the Significance Level property) around each predicted value, and the red dots indicate any outliers. The vertical line represents the end of the data set, and any points after this line are forecast values. The number of data points that are forecast are determined by the Forecast Lead property.

6. Close the Results window. In the Outliers in Exported Data property group, change the value of the Smooth outliers property to **Yes**. Ensure that the Outlier Replacement property is set to **Predicted Value**. Rerun the **TS Exponential Smoothing** node. Click **OK** in the Run Status window after the node has successfully run.

   Select the ellipses next to the Exported Data property for the **TS Exponential Smoothing** node. Notice that only a transaction data set exists. Select the transaction data set and click **Browse**.



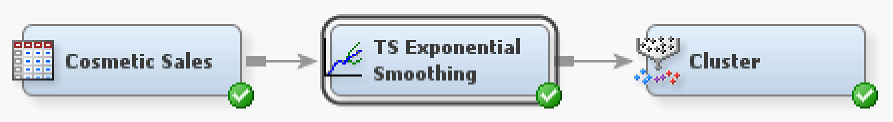
EMWS6.TSESM_TRANSACTION

	Time Series ID	CrossID: product	CrossID: state	Time ID	Sales
59	2.0	54105	GA	May 1, 1997	254771.0
60	2.0	54105	GA	Jun 1, 1997	270687.0
61	2.0	54105	GA	Jul 1, 1997	24225...
62	2.0	54105	GA	Aug 1, 1997	243483.0
63	2.0	54105	GA	Sep 1, 1997	227894.0
64	2.0	54105	GA	Oct 1, 1997	229676.0
65	2.0	54105	GA	Nov 1, 1997	264830.0
66	2.0	54105	GA	Dec 1, 1997	232812.0
67	2.0	54105	GA	Jan 1, 1998	297681.0
68	2.0	54105	GA	Feb 1, 1998	278170.0
69	2.0	54105	GA	Mar 1, 1998	293606.0
70	2.0	54105	GA	Apr 1, 1998	30058...
71	2.0	54105	GA	May 1, 1998	289196.0
72	2.0	54105	GA	Jun 1, 1998	303600.0
73	2.0	54105	GA	Jul 1, 1998	352397.0
74	2.0	54105	GA	Aug 1, 1998	298326.0

Two outliers, the observations in rows 61 and 70, are replaced with their predicted values in the Time Series ID: 2 series. The outliers can be left as missing values and handled with missing value imputation techniques.

Example 2: Clustering Forecast Data

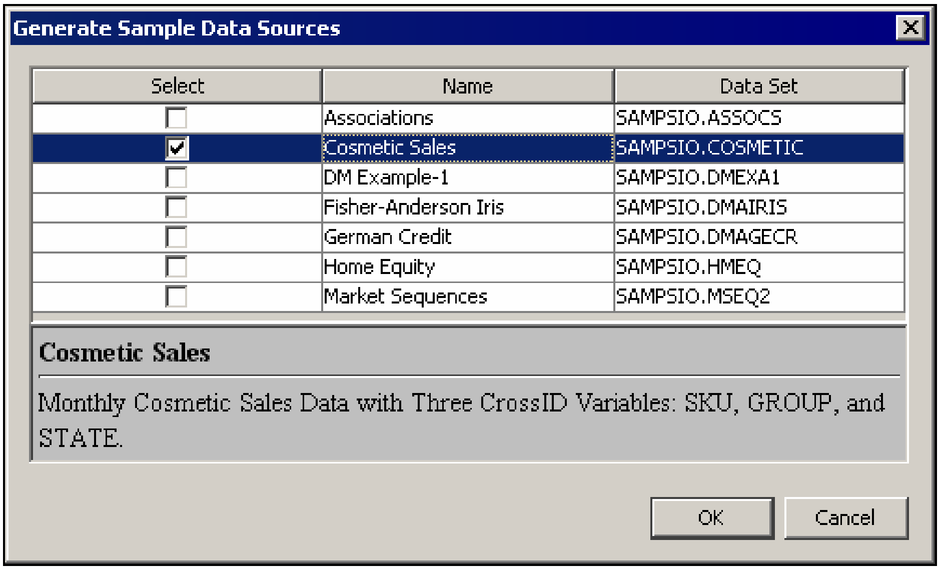
This example illustrates the Output Data Type property of the **TS Exponential Smoothing** node. This example uses the Cosmetic Sales data set. Your completed process flow diagram will resemble the image below.



This example assumes that you have already created and opened both a project and a diagram to contain this example. For information about how to create a project, see

Creating a New Project on page 226 . For information about how to create a diagram, see Create a New Project Diagram on page 230 .

1. First, you need to generate the Cosmetic Sales data set. To do so, select **Help** ⇨ **Generate Sample Data Sources** from the menu bar. In the Generate Sample Data Sources window, select only the Cosmetic Sales data set, as shown below.



Generate Sample Data Sources

Select	Name	Data Set
□	Associations	SAMPSIO.ASSOCS
☑	Cosmetic Sales	SAMPSIO.COSMETIC
□	DM Example-1	SAMPSIO.DMEXA1
□	Fisher-Anderson Iris	SAMPSIO.DMAIRIS
□	German Credit	SAMPSIO.DMAGECR
□	Home Equity	SAMPSIO.HMEQ
□	Market Sequences	SAMPSIO.MSEQ2

Cosmetic Sales

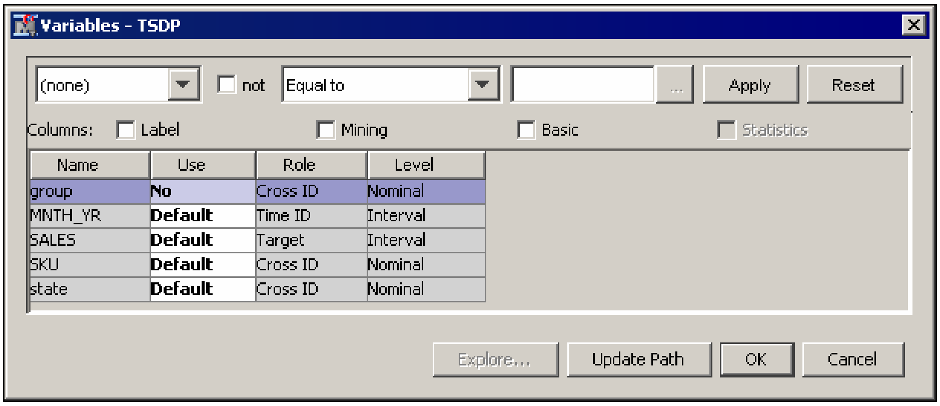
Monthly Cosmetic Sales Data with Three CrossID Variables: SKU, GROUP, and STATE.

Click **OK**.

2. Next, drag the Cosmetic Sales data set into your diagram workspace. Set the value of the Role property to **Raw**. Alternatively, you can set the value of Role to **Train**.

   From the **Time Series** tab, select the **TS Exponential Smoothing** node and drag it into your diagram workspace. Connect the Cosmetic Sales data set to the **TS Exponential Smoothing** node.

3. Right-click the **TS Exponential Smoothing** node and select **Edit Variables** from the menu. This opens the Variables window, where you can set the use status for each variable. For this example, you want to set the use status for the group variable to **No**, as shown below.



Variables - TSDP

Name	Use	Role	Level
group	**No**	Cross ID	Nominal
MNTH_YR	**Default**	Time ID	Interval
SALES	**Default**	Target	Interval
SKU	**Default**	Cross ID	Nominal
state	**Default**	Cross ID	Nominal

Notice that there are three Cross ID variables for the Cosmetic Sales data set. The **TS Data Preparation** node creates a separate time series for each unique combination of variable values for each Cross ID variable.

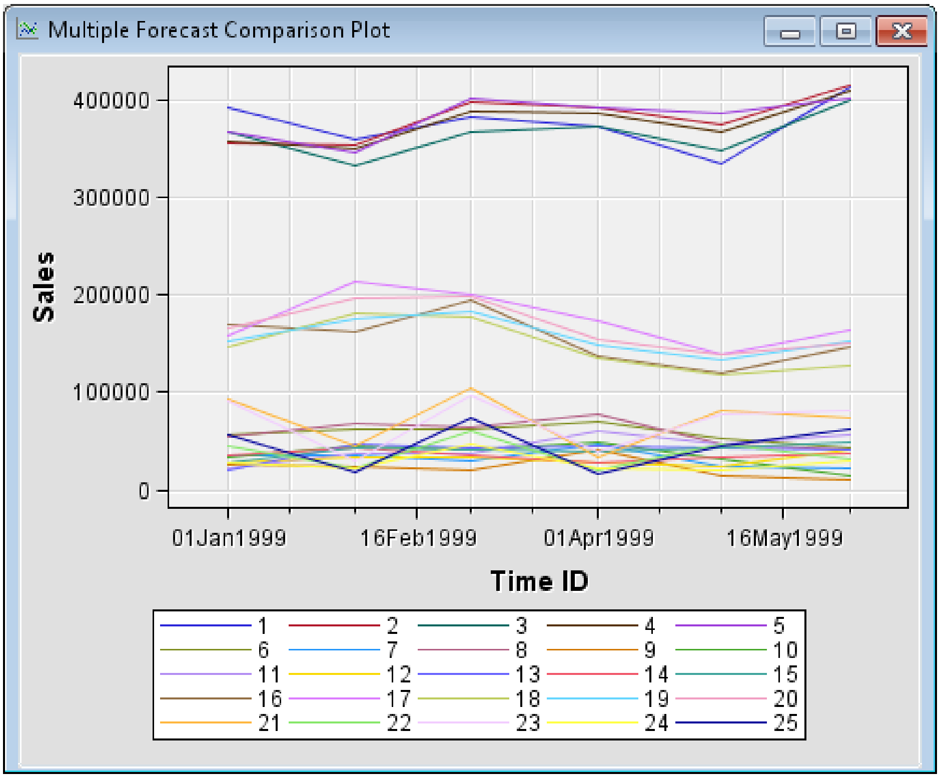
In the Cosmetic Sales data set, the SKU variable has 5 values, the state variable has 5 values, and the group variable has 3 values. If you used all three Cross ID variables, then you would have 75 different time series in this example. After excluding the group variable, there are 25 time series in the similarity analysis.

4. For the **TS Exponential Smoothing** node, set the Output Data Type property to **Coordinated Forecast Data**.

5. From the **Explore** tab, drag the **Cluster** node into the diagram workspace. This example uses the default property settings. Connect the **TS Exponential Smoothing** node to the **Cluster** node.

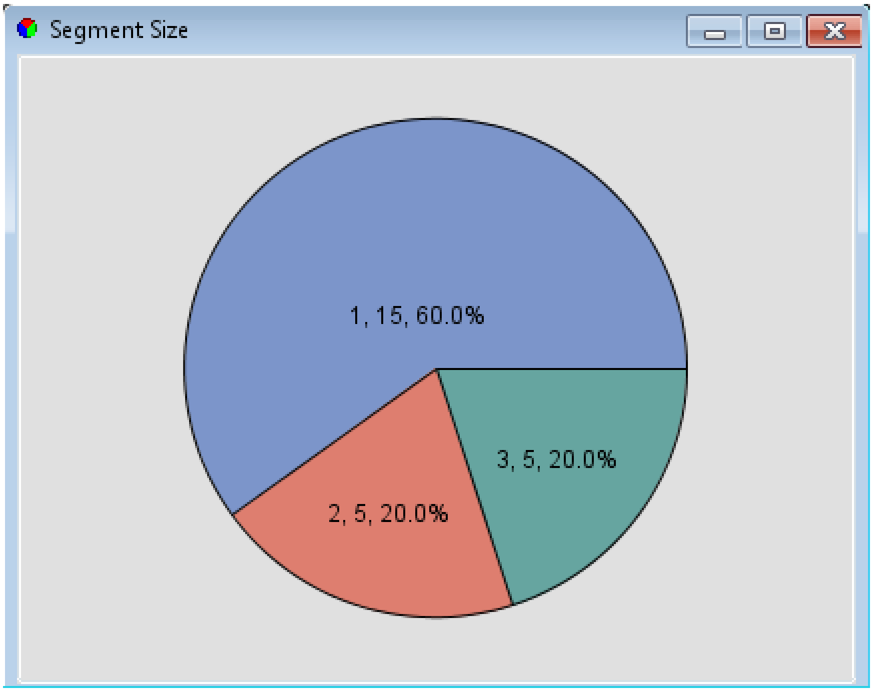
6. Right-click the **Cluster** node and select **Run** from the menu. In the Confirmation window, select **Yes** to run the process flow diagram. In the Run Status window, click **OK**.

7. Right-click the **TS Exponential Smoothing** node and select **Results**. This opens the Results window for the **TS Exponential Smoothing** node. Notice that the Multiple Forecast Comparison Plot window contains three distinct groups of time series, as shown below.



8. Close the Results window. Select on the **Cluster** node in the diagram workspace, and select the ellipses next to the Imported Data property. Select the training data set and click **Browse**. Notice that the input data set for the **Cluster** node contains a variable for forecast time point and that there is an observation for each time series. Close the training data set window and the Imported Data window.

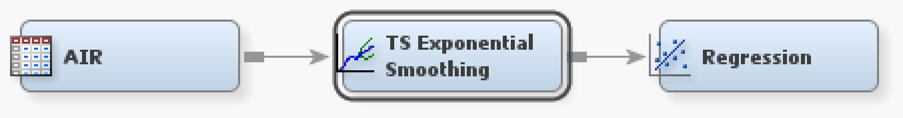
9. Right-click the **Cluster** node, and select **Results** from the drop-down menu. In the Segment Size window, you can see that the **Cluster** node identified three clusters of data. The resultant segments **1**, **2**, and **3** contain 15, 5, and 5 time series, respectively.



Note: To see the numbers and percentages in the graph, select **Edit** ⇨ **Graph Properties** from the main menu. In the Properties — Pie window, enable the **Value** and **Percentage** options.

Example 3: Forecast Input Time Series

The **TS Exponential Smoothing** node can be used to forecast input time series, which can be time varying covariates. This example uses the AIR data set located in the SAMPSIO library. The **TS Exponential Smoothing** node forecasts input time series instead of a target variable. The exported data can be used in regression analysis. Your completed process flow diagram will resemble the image below.



This example assumes you have already created and opened a project diagram. For information about how to create a project diagram, see Create a New Project Diagram on page 230 .

Use the following steps to add the SAMPSIO.AIR data set to your project:

1. In the SAS Enterprise Miner project navigator, right-click **Data Sources** and select **Create Data Source**. This opens the Data Source wizard. The default selection for metadata source is **SAS Table**. Click **Next**.

2. In the **Table** field, type `SAMPSIO.AIR`. Click **Next**.

3. The Table Information window provides summary information about the SAMPSIO.AIR data set. Click **Next** to advance to the Metadata Advisor Options window.

4. In the Metadata Advisor Options window, click **Advanced**. Click **Next**.

5. In the Column Metadata window make the following variable role assignments:

   - Set the role of the variable Dust to **Target**.

- Set the role of the variables Day and Hour to **Rejected**.

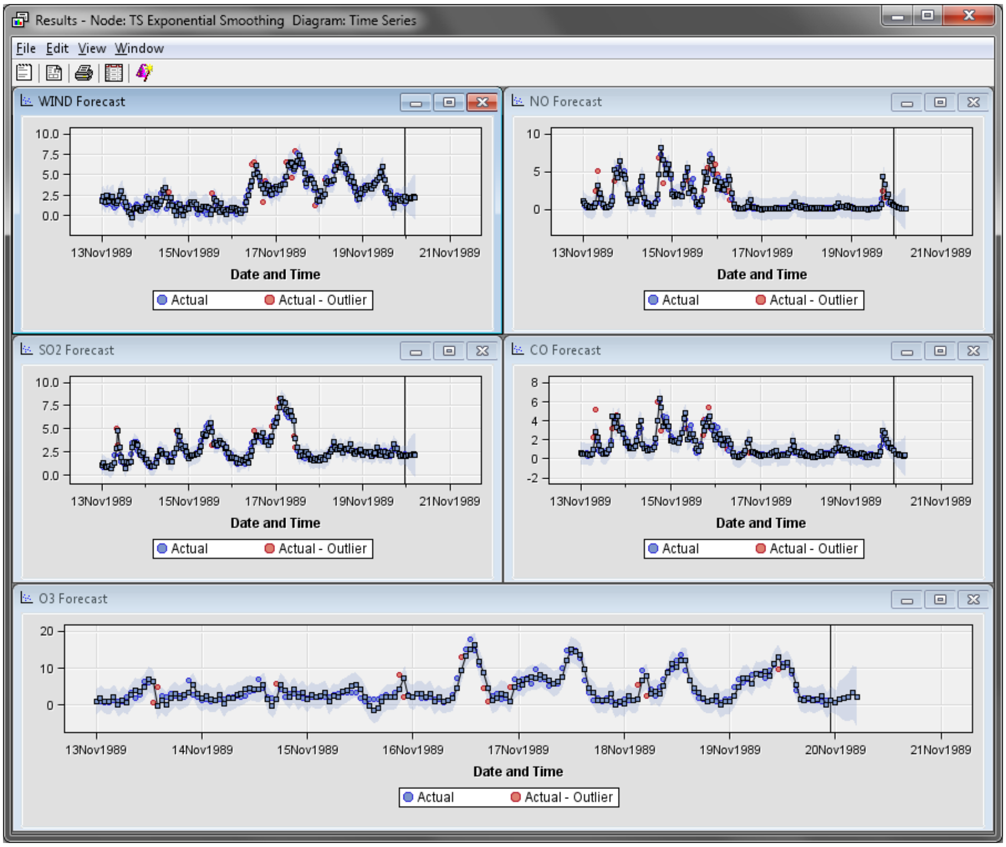
Click **Next**.

6. In the Create Sample window, click **Next**.
7. In the Data Source Attributes window, click **Next**.
8. In the Summary window, click **Finish**.

The AIR data source should appear as a data source in the SAS Enterprise Miner project navigator.

Follow the steps below to create and run your process flow diagram.

1. Drag the AIR data set onto your diagram workspace.
2. From the **Time Series** tab, drag the **TS Exponential Smoothing** node onto the diagram workspace. Set the Specify an Interval property to **Hour** and the Forecast Input Time Series property to **Yes**.
3. Connect the AIR data set to the **TS Exponential Smoothing** node. Right-click the **TS Exponential Smoothing** node and select **Run**. In the Confirmation window, click **Yes**.
4. In the Run Status window, click **Results**. The **TS Exponential Smoothing** node created five input time series forecast plots, shown below. By default, only three plots are shown. You can view the other two by selecting **View** ⇨ **Forecasting Plots**.



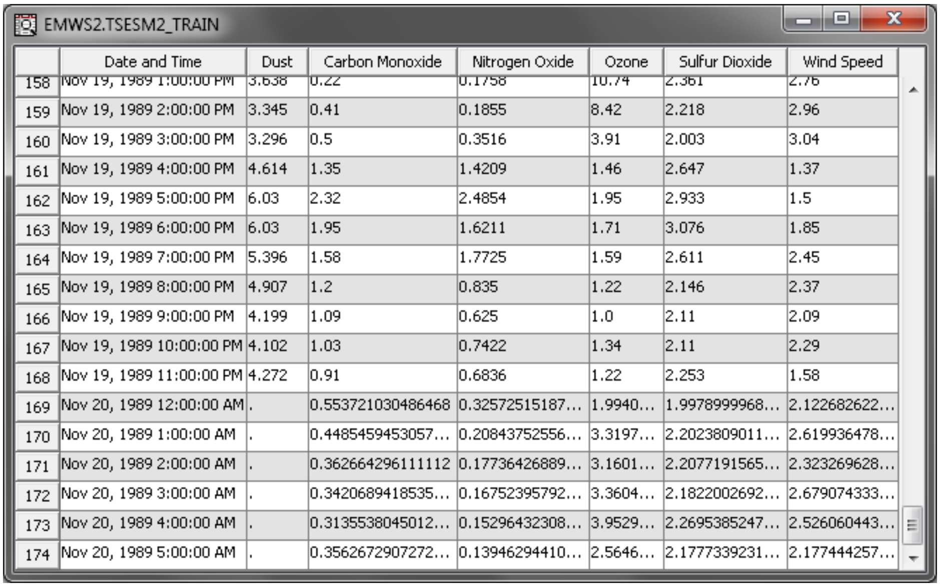
Close the Results window.

5. From the **Model** tab, drag the **Regression** node onto the diagram workspace. This example uses the default settings for the **Regression** node. Connect the **TS**

Exponential Smoothing node to the **Regression** node. The five input time series are used as time-dependent covariates in the **Regression** node.

6. Right-click the **Regression** node and select **Run** from the drop-down menu. In the Confirmation window, select **Yes** to run the process flow diagram. After the process flow diagram has run successfully, click **OK** in the Run Status window.

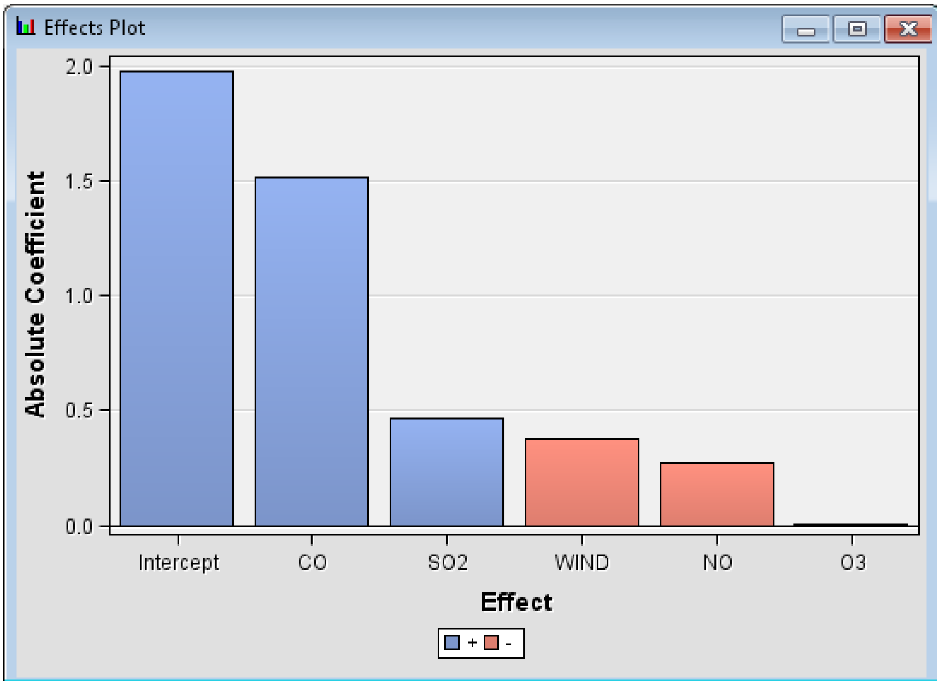
7. Select the **Regression** node in the diagram workspace and click the ellipses next to the Imported Data property. Select the training data set and click **Browse**. Notice that the target variable Dust contains six missing values for the final six observations. Also notice the variables Carbon Monoxide, Nitrogen Oxide, Ozone, Sulfur Dioxide, and Wind Speed contain six predicted values for the final six observations. There are six missing and predicted values because you specified **6** in the Forecast Lead property of the **TS Exponential Smoothing** node.



	Date and Time	Dust	Carbon Monoxide	Nitrogen Oxide	Ozone	Sulfur Dioxide	Wind Speed
158	Nov 19, 1989 1:00:00 PM	3.638	0.22	0.1758	10.74	2.361	2.76
159	Nov 19, 1989 2:00:00 PM	3.345	0.41	0.1855	8.42	2.218	2.96
160	Nov 19, 1989 3:00:00 PM	3.296	0.5	0.3516	3.91	2.003	3.04
161	Nov 19, 1989 4:00:00 PM	4.614	1.35	1.4209	1.46	2.647	1.37
162	Nov 19, 1989 5:00:00 PM	6.03	2.32	2.4854	1.95	2.933	1.5
163	Nov 19, 1989 6:00:00 PM	6.03	1.95	1.6211	1.71	3.076	1.85
164	Nov 19, 1989 7:00:00 PM	5.396	1.58	1.7725	1.59	2.611	2.45
165	Nov 19, 1989 8:00:00 PM	4.907	1.2	0.835	1.22	2.146	2.37
166	Nov 19, 1989 9:00:00 PM	4.199	1.09	0.625	1.0	2.11	2.09
167	Nov 19, 1989 10:00:00 PM	4.102	1.03	0.7422	1.34	2.11	2.29
168	Nov 19, 1989 11:00:00 PM	4.272	0.91	0.6836	1.22	2.253	1.58
169	Nov 20, 1989 12:00:00 AM	.	0.553721030486468	0.32572515187...	1.9940...	1.9978999968...	2.122682622...
170	Nov 20, 1989 1:00:00 AM	.	0.4485459453057...	0.20843752556...	3.3197...	2.2023809011...	2.619936478...
171	Nov 20, 1989 2:00:00 AM	.	0.362664296111112	0.17736426889...	3.1601...	2.2077191565...	2.323269628...
172	Nov 20, 1989 3:00:00 AM	.	0.3420689418535...	0.16752395792...	3.3604...	2.1822002692...	2.679074333...
173	Nov 20, 1989 4:00:00 AM	.	0.3135538045012...	0.15296432308...	3.9529...	2.2695385247...	2.526060443...
174	Nov 20, 1989 5:00:00 AM	.	0.3562672907272...	0.13946294410...	2.5646...	2.1777339231...	2.177444257...

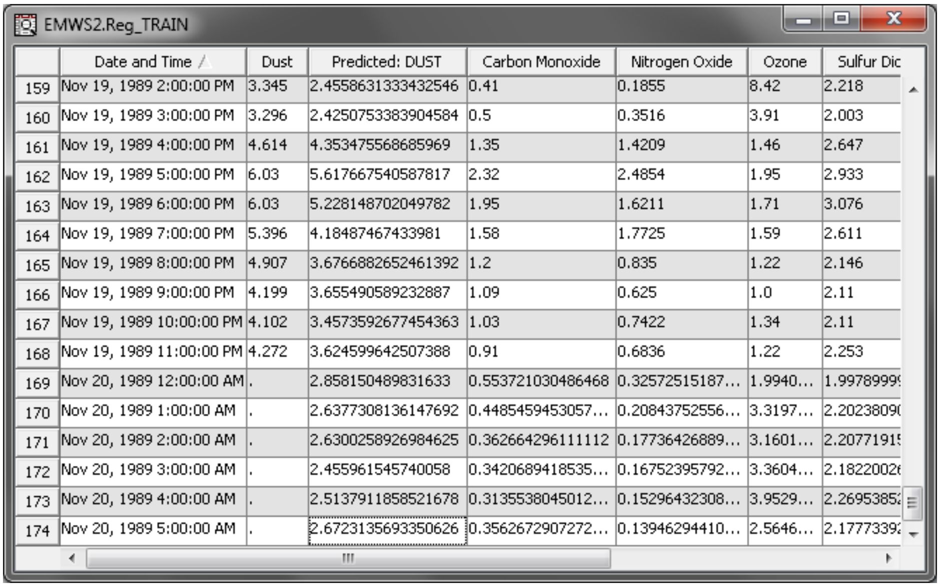
Close the Train data window and the Imported Data window.

8. The **Regression** node created coefficient estimates for all input variables and a predictive regression model. Right-click the **Regression** node and select **Results** to open the Results window. In the Results window, you can see coefficient estimates for all input variables and the predictive regression model.



Close the Results window.

9. Select the **Regression** node in the diagram workspace, and click the ellipses next to the Exported Data property. Select the training data set and click **Browse**. Notice that the **Regression** node has predicted values for the missing observations.


EMWS2.Reg_TRAIN

	Date and Time	Dust	Predicted: DUST	Carbon Monoxide	Nitrogen Oxide	Ozone	Sulfur Dic
159	Nov 19, 1989 2:00:00 PM	3.345	2.4558631333432546	0.41	0.1855	8.42	2.218
160	Nov 19, 1989 3:00:00 PM	3.296	2.4250753383904584	0.5	0.3516	3.91	2.003
161	Nov 19, 1989 4:00:00 PM	4.614	4.353475568685969	1.35	1.4209	1.46	2.647
162	Nov 19, 1989 5:00:00 PM	6.03	5.617667540587817	2.32	2.4854	1.95	2.933
163	Nov 19, 1989 6:00:00 PM	6.03	5.228148702049782	1.95	1.6211	1.71	3.076
164	Nov 19, 1989 7:00:00 PM	5.396	4.18487467433981	1.58	1.7725	1.59	2.611
165	Nov 19, 1989 8:00:00 PM	4.907	3.6766882652461392	1.2	0.835	1.22	2.146
166	Nov 19, 1989 9:00:00 PM	4.199	3.655490589232887	1.09	0.625	1.0	2.11
167	Nov 19, 1989 10:00:00 PM	4.102	3.4573592677454363	1.03	0.7422	1.34	2.11
168	Nov 19, 1989 11:00:00 PM	4.272	3.624599642507388	0.91	0.6836	1.22	2.253
169	Nov 20, 1989 12:00:00 AM	.	2.858150489831633	0.553721030486468	0.32572515187...	1.9940...	1.9978999
170	Nov 20, 1989 1:00:00 AM	.	2.6377308136147692	0.4485459453057...	0.20843752556...	3.3197...	2.2023809
171	Nov 20, 1989 2:00:00 AM	.	2.6300258926984625	0.362664296111112	0.17736426889...	3.1601...	2.2077191
172	Nov 20, 1989 3:00:00 AM	.	2.455961545740058	0.3420689418535...	0.16752395792...	3.3604...	2.1822002
173	Nov 20, 1989 4:00:00 AM	.	2.5137911858521678	0.3135538045012...	0.15296432308...	3.9529...	2.2695385
174	Nov 20, 1989 5:00:00 AM	.	2.6723135693350626	0.3562672907272...	0.13946294410...	2.5646...	2.1777339

Close the Train window and the Exported Data window.

Part 18

Batch Processing

Chapter 84
SAS Enterprise Miner 12.3 Batch Processing Help

SAS Enterprise Miner 12.3 Batch Processing Help

Overview of SAS Enterprise Miner 12.3 Batch Processing

SAS Enterprise Miner 12.3 batch processing is a macro-based interface to the SAS Enterprise Miner 12.3 client / server environment that operates without running the SAS Enterprise Miner GUI (Graphical User Interface). Batch processing supports the building, running, and reporting of SAS Enterprise Miner 12.3 process flow diagrams. The same diagram may be run from either the SAS Enterprise Miner 12.3 GUI or from a batch job. The results may be viewed in the SAS Enterprise Miner 12.3 GUI, or integrated into a reporting SAS program.

SAS Enterprise Miner 12.3 batch processing code is not designed to be submitted to SAS Enterprise Miner through the SAS Enterprise Miner GUI Program Editor. Instead, SAS Enterprise Miner 12.3 batch processing code should be submitted in a SAS batch job or submitted through the base SAS Program Editor window.

All SAS Enterprise Miner 12.3 actions have batch interfaces. SAS Enterprise Miner 12.3 produces batch code for process flows built in the GUI, or process flow diagrams can be manually coded by experienced SAS Enterprise Miner users. The macro interface used for batch processing in SAS Enterprise Miner 12.3 is compatible with all SAS Enterprise Miner 12.3 file structures and SAS language capabilities. These are the tools users need to automate creation and execution of a data mining analysis.

With batch processing, you can

- Schedule processor-intensive SAS Enterprise Miner process flow diagrams for off-peak processing hours.
- Automate daily, weekly, or monthly SAS Enterprise Miner process flow diagram runs and model training.
- Automate event-driven SAS Enterprise Miner process flow diagram runs and model training.
- Automate regular data integration jobs for Enterprise Miner.
- Create data mining templates for analysts and business users

The batch processing tool is intended for use by statisticians and programmers who have strong experience in writing SAS code and building SAS Enterprise Miner models.

Components of SAS Enterprise Miner 12.3 Batch Processing

When you construct process flow diagrams in the GUI interface to Enterprise Miner, SAS configures many settings in the background. To replace these necessary configurations, batch coders must create definition data sets that provide SAS Enterprise Miner 12.3 with the relational information and individual node settings that are required for a valid process flow diagram. The following SAS definition data sets are the necessary components that you need to create a working SAS Enterprise Miner 12.3 process flow diagram when you use batch processing:

Data Source Data Set	The data source that you want to use in your process flow diagram.
Workspace Data Set	The values of the configuration properties in your workspace.
Nodes Data Set	The nodes that are used in your process flow diagram.
Actions Data Set	The actions to be taken by each node in your process flow diagram.
Connections Data Set	The connections that indicate directional data flow from predecessor nodes to successor nodes.
Node Properties Data Set	The functional properties of each individual node in the process flow diagram.

After you code the component data sets, it is not necessary to rebuild the component data sets for every successive run. You need only to rebuild component data sets if you change something in your process flow diagram, such as adding or subtracting nodes, changing node connections, or changing node configuration properties. When that happens, you need to rebuild only the component data set or data sets that are affected by your changes.

The %EM5BATCH Macro

The %EM5BATCH macro is stored in your SAS Enterprise Miner 12.3 installation, in the folder **SASROOT/dmine/sasmacro**, where SASROOT represents the root directory of the SAS installation on a computer.

Appendix 1 contains tables that describe the variables and manual formatting required for each of the five component definition data sets if you plan to manually write the batch processing code.

The general form of the %EM5BATCH macro is:

```
%EM5BATCH (operation, parameters);
```

Each parameter value is the name of a SAS data set. Each parameter value data set is encoded in name/value pair format.

The available operations are

EXECUTE	Submit the flow stream to SAS for program execution on page 1349
PURGEPROJECT	Deletes the directory tree from the entire last project (no parameter)
PURGEWORKSPACE	Deletes the last used workspace within the last project (no parameter).

The %EM5BATCH macro follows the general form of:

```
%EM5BATCH(execute,
   workspace=workspace,
   nodes=nodes,
   connect=connect,
   nodeproperties=nodeproperties,
   actions=actions
    );
```

More detailed examples of macro usage for build and execute operations is illustrated in sequential examples that appear in Creating the Batch Processing Code on page 1335 .

Batch Processing Code Data Set Caching

The %EM5BATCH macro caches the last data sets that were specified by creating the macro variables. If component data sets are not explicitly specified in the %EM5BATCH macro, it retrieves the most recent table that was specified for each component data set.

For example, if you submit the statement

```
%EM5BATCH(execute,
      workspace=work.workspace,
      nodeprops=work.nodeprops,
      action=work.actions,
      nodes=work.nodes,
      connect=work.connect,
      datasources=
      );
```

and then you submit the following statement

```
%EM5BATCH(execute,
      nodeprops=work.nodeprops,
      action=work.actions
      );
```

The %EM5BATCH macro will use the previously defined work.workspace, work.nodes, and work.connect arguments.

If you don't want to use the previously defined arguments, you can either use PROC DATASETS to delete the old data sets that you do not want to be used, or submit the macro with blank arguments to prevent the most recent data from being used, as shown below:

```
%EM5BATCH(execute,
      workspace=work.workspace,
      nodeprops=work.nodeprops,
      action=work.actions,
      nodes=,
      connect=,
      datasources=
      );
```

SAS Enterprise Miner Data Sources

EMDS

SAS Enterprise Miner reads data sources from an EMDS (Enterprise Miner Data Sources) library. Data sources are not the actual training data, but are the metadata that defines the source data. The source data must exist in some allocated libname (for example, **SAMPSIO**). The source data libname allocation and target profile information should be defined in either the SAS Enterprise Miner server startup code (preferred), or in the SAS Enterprise Miner project startup code.

EMLDS

SAS Enterprise Miner creates an EMLDS (Enterprise Miner Local Data Sources) library in your project. Each project will have its own EMLDS library.

EMGDS

You can define the EMGDS (Enterprise Miner Global Data Sources) library in your SAS Enterprise Miner server startup code. The libname statement in the server startup code may resemble

```
libname EMGDS "/projects/global_datasources";
```

The EMGDS library supports

- data source definitions that are shared between multiple users and in multiple projects
- the creation of data sources by external processes such as nightly data integration jobs or query applications. These external processes can create data source definitions that can be used by SAS Enterprise Miner GUI users.

You can overwrite the assignment of the EMGDS library by including a different EMGDS libname in your project startup code. For example, you could include the following in the code:

```
libname EMGDS "projects/userID/my_global_datasources";
```

Using the EMDS Library

SAS Enterprise Miner assigns the EMDS library as concatenation of the EMLDS and EMGDS libraries:

```
libname EMDS (EMLDS EMGDS);
```

If you run SAS Enterprise Miner 12.3 with the GUI, then you cannot write to the EMGDS library. Because of the EMDS libname concatenation, any changes that you make to a global data source are written to the EMLDS library. Therefore, changes to these data sources are made within the project. This enables you to modify a global data source without affecting the work of other users who are using the same data source.

You can remove the changes that you make to a global data source by either deleting the local changes, or by creating a new project.

The %EMDS Macro

The %EMDS macro supports creation of SAS Enterprise Miner 12.3 data source definitions and management of table and column metadata. Data source definitions contain the metadata required for data mining, such as table names, locations, variable roles, and so on. The %EMDS macro is very useful for automating routine data source registration from batch data preparation jobs. Consider a SAS code job that extracts data from relational tables in a data warehouse, and then creates a table ready for predictive modeling. You can use the %EMDS macro to create the data source definitions that are needed for either the SAS Enterprise Miner 12.3 GUI (Graphic User Interface) or for the SAS Enterprise Miner batch processing environment.

You can create data source definitions in any directory, but creating them in the EMGDS location is one way to make them accessible by Enterprise Miner. The source data libname should be defined in either the server startup code or in the project code.

You can use the %EMDS macro

- to automate metadata creation when you run external data integration jobs or query applications.
- to create temporary data source definitions when you run the %EM5BATCH macro.

The general form of the %EMDS macro is:

```
%EMDS (option, setting);
```

You can specify the following options with the %EMDS macro:

Use This Option...	To Specify...	Default Value
data=	input data set	&syslast
rootLibrary=	target libname	EMGDS
target=	target variable	
freq=	frequency variable	

Use This Option...	To Specify...	Default Value
cost=	cost variable	
id=	id variable	
General Data Source Information		
name=	display name of the data source	
userid=	user ID of the user who generates the data source	
tablerole	modeling role of the table	RAW
Advisor Options *Note:* Advisor Options only apply when adviseMode=ADVANCED		
adviseMode=	basic or advanced options	BASIC
applyIntervalLevelLowerLimit	whether to apply the intervalLevelLowerLimit option	Y
intervalLevelLowerLimit	lower limit for interval variables	20
applyMaxClassLevels	whether to apply the maxClassLevels options	Y
maxClassLevels	maximum number of class levels before a variable is rejected	20
identifyEmptyColumns	whether to identify columns	Y
maxPercentMissing	maximum percent missing values allowed before a variable is rejected	20
Information Only — Not used for modeling		
segment=	segment variable	
samplerate=	sampling rate	
useexternal=	whether to use external data	

The following code is one way to use the %EMDS macro:

```
filename code catalog "sashelp.emutil.em_loadutilmacros.source";
%include code;
```

```
libname EMGDS "/projects/global_datasources";
%emds(data=mylib.mydata,
      target=sometarget);
```

Note: to view the data source in the SAS Enterprise Miner GUI when you run the sample code above, you must have the EMGDS library assigned in either your server startup code or in your project startup code.

The following code is a more detailed example of the %EMDS macro:

```
%EMDS(data=MyLib.MyData,
      target=purchase,
      id=customerid,
      userid=chrobi
      tablerole=train
      AdviseMode=advanced,
      ApplyMaxClassLevels=Y,
      MaxClassLevels=25,
      ApplyIntervalLevelLowerLimit=Y,
      IntervalLowerLimit=0,
      segmentvariable=customergroup
      );
```

You can also create target profile definitions for a data source by using the %EMTP macro. See "%EMTP Macro" on page 214 in the Target Profiler documentation for more information.

The %EMTP Macro

You can use the %EMTP macro to create target profile definitions for a data source. Use this macro when you want to create decision processing models. Decision processing models are weighted by the values of decision alternatives, such as profit matrices, loss matrices, cost elements, or prior probabilities.

The general form of the %EMTP macro is:

```
%EMTP (component=setting);
```

You can specify the following components with the %EMDS macro:

Use This Option...	To Specify...	Default Value
data=	input data	
target=	target variable's name	
columnsmeta=	columns metadata set	
decdata=	name of the decision data set (optional)	WORK.DECDATA
decmeta=	name of the decision metadata set (optional)	WORK.DECDATA
dectype=	type of decision (LOSS or PROFIT) (optional)	

Use This Option...	To Specify...	Default Value
numdec=	number of decisions (optional)	

The following example code enables the %EMTP macro and specifies the input data set, the target variable name, and the columnsmeta data set:

```
filename code catalog "sashelp.emutil.emtp.source";
%include code;
libname EMGDS "/projects/global_datasources";
%EMTP(data=mylib.mydata,
      target=sometarget,
      columnsmeta=emds.mydata_cm
      );
```

A slightly more complicated example usage of the %EMTP macro enables the %EMTP macro and specifies the input data set, the target variable, the name of the decision data set, the type of the decision data set, the columnsmeta data set, and the number of decisions:

```
filename code catalog "sashelp.emutil.emtp.source";
      %include code;
      libname EMGDS "/projects/global_datasources";
      %EMTP(data=mylib.fraud,
          target=fraud,
          decdata=mylib.decdata,
          dectype=loss,
          columnsmeta=emds.mydata_cm,
          numdec=2
          );
```

You should use SAS code if you want to modify values of cells in the decdata decision tables created by the %EMTP macro.

Major Steps to Create Batch Processing Code

The definition data sets are an integral part of batch processing model code, combined with statements that define the path to your definition data sets and instruct SAS to run your code. For a typical batch processing input stream, use the following coding sequence:

1. Define the library path to your definition data sets on page 1336
2. Define your Data Source (Data Source Data Set) on page 1336
3. Define your project workspace properties (Workspace Data Set) on page 1336
4. Define the nodes used in your process flow diagram (Nodes Data Set) on page 1340
5. Define the actions to be taken by the nodes (Actions Data Set) on page 1341
6. Define the connections between nodes (Connections Data Set) on page 1342
7. Define the properties of each node (Node Properties Data Set) on page 1343
8. Completed Batch Code on page 1346
9. Execute the process flow diagram on page 1349

Notes about the Batch Processing Code Examples

- All code in this document is SAS code.
- Data set contents are coded in definition statements that use SAS DATALINES input.
- You can include comments in the code between the definition statements.
- Comments and definition statements cannot be combined -- each line must be either a definition statement or a comment.
- Any line whose first nonblank character is an asterisk (*), a slash (/), or an exclamation mark (!) is treated as a comment.

The following restrictions apply to batch processing code that is submitted to the SAS Enterprise Miner 12.3 code editor. SAS Enterprise Miner 12.3 batch processing code is not designed to be submitted to Enterprise Miner through the SAS Enterprise Miner GUI Program Editor. Instead, SAS Enterprise Miner 12.3 batch processing code should be submitted in a SAS batch job or submitted through the base SAS Program Editor window.

Creating the Batch Processing Code

Overview

The following example code is a generic template for a batch processing input stream. The steps to create each section of code are listed beneath the generic example.

```
/* Define the Library Path to your Definition Data Sets*/
   libname library-name-for-data-set-storage 'path-to-library-location'

/* Define the Data Source Data Set */
   libname library-name-for-data-mining-source-data 'path-to-library-location';
   -options;
   %let data=
        [library-name-for-data-mining-source-data].[data-mining-source-data-set-name]
   %let target= [target-variable-name]
   %let role= [target-variable-name]

/* Define the Workspace Data Set*/
   libname library-name-for-data-set-storage;

/* Define the Nodes Data Set*/
   libname library-name-for-data-set-storage;
/* Define the Actions Data Set*/
   libname library-name-for-data-set-storage;
/* Define the Connections Data Set*/
   libname library-name-for-data-set-storage;
/* Define the Node Properties Data Set*/
   libname library-name-for-data-set-storage;
/* Execute the Batch Processing flow */
   %em5batch(execute,
             workspace=[workspace-data-set-name],
             nodes=[nodes-data-set-name],
             connect=[connection-data-set-name],
             nodeprops=[node-properties-data-set-name]),
             action=[actions-data-set-name]
             );
   run;
```

The generic batch processing code steps are explained in detail below.

Define the library path to your definition data sets

Start creating your batch processing code by defining your library path. The library path tells SAS where the library that contains your definition data sets is located.

```
/* Define the Library Path to the Definition Data Sets */

      libname BoxerTest 'c:\EMBoxer';
```

Define the data source data set (Data Source Data Set)

Continue creating your batch processing code by defining the data source that you want to mine. If your data source is located in a library other than the one which contains your definition data sets, you will need to make a LIBNAME statement to define the path to your data source as well.

For example, to define the Home Equity data set 'HMEQ' from the SAS 'SAMPSIO' samples library as your data source, you could code:

```
/* Define the Data Source Data Set */

libname SAMPSIO 'c:\Program Files\SAS 9.4\dmine\sample';
options fmtsearch=(sampsio.emfmt);
```

Use a LIBNAME statement to define the path to your data source. You can use a two-level LIBNAME statement or a one-level LIBNAME statement and a %LET statement to specify the data source file. You also use %LET statements to declare any required variable model roles and variable measurements. The example uses a binary target variable named BAD.

Define the project workspace properties (Workspace Data Set)

Next you must define the batch processing project workspace properties. The project workspace properties portion of your code describes the SAS Enterprise Miner project that you are creating, and the properties use terms that SAS understands. SAS needs to know your project's name, location, password information, and other important information such as the number of threads to use during runtime and which methods to use during debugging.

After the workspace property data set is built, you do not need to regenerate it for each run. You can save your workspace property data set and reference it as needed for future runs.

```
PROPERTY=property VALUE=value; output;
```

where **`property`** is the name of the workspace property as defined. For example, in the following table PROJECTLOCATION is a property name. **`Value`** is the value to be used for that property.

Note: Properties in bold are required properties and may not remain blank. For example, the value in the required property PROJECTLOCATION is **`c:\Boxer`**

Table 84.1 Project Workspace Properties, Values, and Descriptions

Property Name	Example Value	Description
PROJECTLOCATION	value=c:\Boxer	The local path to your SAS Enterprise Miner 12.3 project files.

Property Name	Example Value	Description
PROJECTNAME	value=BoxerTest	The name of your SAS Enterprise Miner 12.3 project.
WORKSPACENAME	value=EMWSnn	A unique name for your workspace, which is EMWS concatenated with a two-digit number.
EDITMODE	value= R | M	Batch processing permits you to create, modify, and execute actions in a process flow. When you change an existing flow, choose one option: • **R**: Replace clears the existing workspace and replaces it with your new configuration. • **M**: Merge (Default Setting) simply adds the new configuration to the existing diagram.
FORCERUN	Boolean values (Y or N)	The FORCERUN property allows you to control whether the workspace within a flow automatically reruns, even if the flow has executed before. • **Y**: (Default Setting) Sets the macro variable EM_FORCERUN to Y, and on subsequent process flow runs, all nodes within the flow will rerun whether they need to or not. • **N**: Sets the macro variable EM_FORCERUN to N, and does not force all nodes within the flow to rerun on subsequent flow executions.
USERID	value=userID	Batch processing uses USERID when generating reports. Your batch processing reports will display the User ID you submit with your batch processing code. If you do not submit a User ID, SAS generates one for you.

Property Name	Example Value	Description
SESSIONID	value=[user definable]	The SESSIONID helps manage user access and prevents multiple users from opening the same session concurrently. If you do not specify SESSIONID, SAS generates one for you.
SUMMARYDATASET	value=library.dataset	A two-level libref pointing to the library and report cross-reference table that is the summary data set.
NUMTASKS	value= SINGLE | HARDWARE | MAX | n *Note:* There is no upper bound for this setting. If you configure a value that is larger than the number of CPUs on your server, you will degrade computing performance.	On multiprocessor machines, you can configure the number of threads that are allocated for control functions and node execution. • **SINGLE**: Default mode; same as 0; all nodes and control functions run in one SAS thread. • **HARDWARE**: The number of threads launched is based on the number of processors on the server. • **MAX**: The number of branches in a flow determines the number of threads launched. Five branches in a flow will start five sessions. • **0**: All nodes and control functions run in one SAS thread; same as SINGLE. • **1**: Control functions run in one thread and all nodes run synchronously in a separate SAS thread. • **2**: Control functions run in one thread and two threads run parallel execution of nodes. • **4**: Control functions run in one thread and four threads run parallel execution of nodes.

Property Name	Example Value	Description
SASCMD	sas –config d:\users\robie \sas91.cfg	SASCMD is the command line invocation of SAS. This invocation sets a pointer to the location of the SAS config file on your local machine.
DEBUG	value= METHOD | WINDOW | LOG | OUTPUT | SOURCE | ALL (more than one may be specified)	• **METHOD**: logs the run-time methods used. • **WINDOW**: run-time logging of class identification. • **LOG**: reroutes node run-time logs from the node folder to the SAS log. • **OUTPUT**: reroutes node run-time output from the node folder to the SAS log. • **SOURCE**: dumps extra information to the log by displaying normally hidden steps. • **ALL**: perform all of the above plus create list dumps and display values of key variables.
UNLOCK	value=Y	When UNLOCK=Y, the workspace is unlocked and monitoring files are deleted. Useful when testing on a workspace that crashed or had a screen control language (SCL) dump in the last run.

Here is an example of a SAS project workspace definition for a workspace data set called "BoxerWorkspace":

- The project is stored on your local PC
- You want to call the project "BoxerTest"
- You want to call the Project Workspace Name "EMWS99"
- You want to call the summary data set "BoxerSummary"
- Your server has two processors
- You want to log run-time methods:

```
/*  Define the Workspace Data Set  */


data BoxerWorkspace;
length property $64 value $100;
```

```
      property= 'PROJECTLOCATION'; value= 'c:\Boxer';       output;
      property= 'PROJECTNAME';     value= 'BoxerTest';    output;
      property= 'WORKSPACENAME';   value= 'EMWS99';       output;
      property= 'SUMMARYDATASET';  value= 'BoxerSummary'; output;
      property= 'NUMTASKS';          value= 'SINGLE';       output;
      property= 'DEBUG';           value= 'LOG OUTPUT';   output;

      run;
```

This code snippet creates a SAS data set called BoxerWorkspace in the BoxerTest library, or using two-level libnames, a SAS Data set called BoxerTest.BoxerWorkspace.

Define the nodes used in your process flow diagram (Nodes Data Set)

Next, build the nodes data set by defining them in your batch processing code. The nodes data set describes the SAS Enterprise Miner node components in terms that SAS understands. SAS needs to know unique node IDs, the node component name, and your description of the node.

Once the nodes data set is built, it does not need to be regenerated for each run unless you want to change the node settings. You can save your nodes data set and reference it as needed for future runs.

The node definitions are a section of DATALINES input, listing one node per statement, in the form

```
ID component description=
```

where

	Description	Type	Length
ID	A unique ID that you choose that begins with a dollar sign. For example, $1	C	12
component	The component name of the node. Node component names for each node are given in Appendix II on page 1431 .	C	32
description	Your description of the tool. For example, SAMPSIO Home Equity Data	C	64

Here is an example of the node definition code using the Data Source, Sample, Partition, Regression, Tree, and Model Comparison nodes:

```
/* Create Nodes Data Set  */

     data BoxerNodes;
     length id $12 component $32 description $64;
     id='$1';  component='DataSource';   description='HomeEquity';    output;
     id='$2';  component='Sample';       description='Sample';        output;
     id='$3';  component='Partition';    description='Partition';     output;
     id='$4';  component='Regression';   description='Reg';           output;
     id='$5';  component='DecisionTree'; description='Tree';          output;
     id='$6';  component='ModelCompare'; description='ModelCompare';  output;

     run;
```

This code snippet creates a SAS data set called BoxerNodes in the BoxerTest library, or using two-level libnames, a SAS data set called BoxerTest.BoxerNodes.

Define the actions to be taken by the nodes (Action Data Set)

After defining the nodes in your process flow diagram, you must define the node actions. The action data set tells SAS which node you want to issue the run command to.

After the action data set is built, you do not need to regenerate it for each run unless you want to run a different node or generate a new report. You can save your action data set and reference it as needed for future runs.

The node actions are a section of DATALINES input, listing one action per statement, in the form

```
ID action
```

where

	Description	Type	Length
ID	The unique ID (beginning with a dollar sign) that is assigned to the node in the nodes data set on page 1340 . For example, $1	C	12
action	There are three node actions : run, update, or report. Assign the appropriate action to the identified node.	C	12

The following example refers to the node definition data set that was created in the previous step, where $1 refers to the Data Source node, $2 refers to the Sample node, $3 refers to the Partition node, $4 refers to the Regression node, $5 refers to the DecisionTree node, and $6 refers to the Model Comparison node. The code instructs SAS to run the Model Comparison node in the flow. As in the SAS Enterprise Miner GUI, running a node in your batch processing flow also runs the predecessor nodes. It is not necessary to issue run instructions to each node in your batch processing process flow diagram.

```
/* Create the Actions to Run Data Set */

    data BoxerActions;
    length id $12 action $40;
    input id action;
    cards;
       $6 run;
    run;
```

This code snippet creates a SAS data set called BoxerActions In the BoxerTest library, or using two-level libnames, a SAS Data set called BoxerTest.BoxerActions.

Define the connections between nodes (Connections Data Set)

After you define the node actions in your process flow diagram, you should define the node connection information for SAS. The connections data set tells SAS how the data flows between each of nodes in your batch processing model.

Once the connections data set is built, it does not need to be regenerated for each run unless you want to change the node connections. You can save your connections data set and reference it as needed for future runs.

The node actions are a section of DATALINES input, listing two nodes per statement, in the form

```
from to
```

where

	Description	Type	Length
from	The unique ID that is assigned to the precursor node of the nodes data set. For example, $1	C	12
to	The unique ID that is assigned to the successor node of the nodes data set. For example, $2	C	12

Nodes that are not connected in the connections data set will not run in your batch processing code. The example below shows that the Data Source, Sampling, Partition, Regression, Decision Tree, and Model Comparison nodes that are identified in the Node Definition Data Set example are connected to each other.

The example below illustrates how to create a data set that will connect nodes.

```
/* Create the Node Connections Data Set */

    data BoxerConnect;
    length from to $12;
    input from to;
    cards;
      $1 $2
      $2 $3
      $3 $4
      $3 $5
      $4 $6
      $5 $6
    ;
    run;
```

This code snippet creates a SAS Node Connection definition data set called BoxerConnect. BoxerConnect is a member of the BoxerTest library. Using two-level libnames, you could call the Node Connections definition data set BoxerTest.BoxerConnect.

Define the properties of each node (Node Properties Data Set)

After you define the node definitions, actions, and connections in your process flow diagram, set the properties for each node in your batch processing code. The node properties data set instructs SAS how you want each node configured when it is run. Node properties vary from node to node. Simple nodes have few properties, while complex nodes (such as modeling nodes) may have many properties you want to set. It is only necessary to declare property settings that vary from the default setting. Node properties that are not declared in the node properties data set assume default settings. If you want to run a node using entirely default settings you do not need to specify anything for the node in the properties data set. Nodes that are connected in your batch processing flow but are not included in the node property data set are run with the node default settings.

The Properties panel is useful for visualizing and understanding the available properties for a node, but you will need to use the Batch Processing names for nodes and properties. The Batch Processing facility requires specific node and node property name which often differ from the GUI. The Batch Processing names for SAS Enterprise Miner 12.3 nodes are found in the tables in the Batch Processing Help Appendix: “Batch Processing Component Names by Node Type” on page 1431 .

Once the node properties data set is built, it does not need to be regenerated for each run unless you want to change the configuration of the property settings of one of the nodes in your process flow diagram. You can save your node properties data set and reference it as needed for future runs.

The node actions are a section of DATALINES input, listing one node property per statement, in the form

```
id property=propertyname value=propertyvalue
```

where

	Description	Type	Length
ID	The unique ID (beginning with a dollar sign) that is assigned to the node in the nodes data set. For example, $1	C	12
propertyname	The name of the node property as defined by the node tool. For example, PORT_DATA_DATA	C	32
propertyvalue	The value to be used for that property. For example, SAMPSIO.HMEQ	C	64

Here is an example code for the node properties data set. The code follows the flow using a Data Source node, a Sample node, a Partition node, a Regression node, a

DecisionTree node, and a Model Comparison node. Notice that in the Node Properties code, only the node properties that vary from default settings are declared. If a node is run completely in its default state, it does not require its own node properties data set.

The Data Source node uses the input data set SAMPSIO.HMEQ. The temporary column metadata file TESTMETA.HMEQ is created using the EMCMETA macro. The property PORT_DATA_COLUMNMETA is configured to use the TESTMETA.HMEQ data set.

The Data Source node is connected to a Sample node, the Sample node is connected to a Partition node, the Partition node connects to (1) a Regression node and (2) a DecisionTree node, and the Regression and DecisionTree nodes both connect to a single Model Comparison node.

```
/* Create Temporary Metadata File with the EMCMETA Macro   */

     %FILENAME(emcmeta,testmeta,sas,perm=yes,subdir=testsrc);
      %INCLUDE testmeta;

    /* Create Node Properties Data Set */
       data BoxerNodeprops;

    length id $12 property $32 value $64;
    /* Data Source Node Properties */

    id= '$1';    property='port_data_data';        value="&data";       output;
    id= '$1';    property='port_data_columnmeta'; value='testmeta'; output;
    id= '$1';    property='role';                 value='train';        output;


    /* Sample Node Properties */

    id= '$2';    property='Method';             value='FirstN';    output;
    id= '$2';    property='SizeType';             value='Computed';  output;
    id= '$2';    property='SizePercent';         value='15';        output;
    id= '$2';    property='SizeObs';             value='5';            output;
    id= '$2';    property='Alpha';                 value='.1';        output;
    id= '$2';    property='Pvalue';             value='.05';        output;
    id= '$2';    property='AdjustFreq';         value='Y';            output;
    id= '$2';    property='StratifyCriterion';  value='Optimal';   output;
    id= '$2';    property='IgnoreSmallStrata';  value='Y';            output;


    /* Partition Node Properties */

    id= '$3';    property='Method';            value='Random';    output;
    id= '$3';    property='TrainPct';        value='70';           output;
    id= '$3';    property='ValidatePct';    value='10';        output;
    id= '$3';    property='TestPct';        value='20';        output;


    /* Regression Node Properties */

    id= '$4';    property='LinkFunction';         value='Probit';    output;
    id= '$4';    property='MinResourceUse';     value='N';         output;
    id= '$4';    property='ModelSelection';     value='Stepwise';  output;
    id= '$4';    property='SelectionCriterion'; value='VERROR';    output;
    id= '$4';    property='Sequential';         value='Y';            output;
```

```
id= '$4';    property='SlEntry';             value='0.025';     output;
id= '$4';    property='SlStay';             value='0.14';      output;
id= '$4';    property='Start';                 value='1';         output;
id= '$4';    property='Stop';                 value='10';        output;
id= '$4';    property='Rule';                 value='Multiple';  output;
id= '$4';    property='MaxStep';             value='10';        output;
id= '$4';    property='StepOutput';         value='Y';         output;
id= '$4';    property='OptimizationTechnique'; value='DBLDOG'; output;
id= '$4';    property='ModelDefaults';         value='N';           output;
id= '$4';    property='MaxIterations';         value='55';        output;
id= '$4';    property='MaxFunctionCalls';   value='1005';        output;
id= '$4';    property='MaxTime';             value='51452';     output;
id= '$4';    property='ConvDefaults';       value='N';            output;
id= '$4';    property='AbsFValue';             value='5';            output;
id= '$4';    property='AbsFTime';             value='4';            output;
id= '$4';    property='AbsGValue';             value='0.01';         output;
id= '$4';    property='AbsGTime';             value='10';        output;
id= '$4';    property='AbsXValue';             value='1E-6';        output;
id= '$4';    property='AbsXTime';             value='5';            output;
id= '$4';    property='FConvValue';         value='15';        output;
id= '$4';    property='FConvTimes';         value='6';            output;
id= '$4';    property='GConvValue';         value='2E-5';        output;
id= '$4';    property='GConvTimes';         value='5';            output;
id= '$4';    property='ClParm';             value='Y';            output;
id= '$4';    property='CovB';                 value='Y';            output;
id= '$4';    property='Simple';             value='Y';            output;
id= '$4';    property='Details';             value='Y';            output;
id= '$4';    property='PrintDesignMatrix';     value='Y';            output;


/* Decision Tree Node Properties */

id= '$5';    property='Criterion';         value='CHISQ';     output;
id= '$5';    property='SigLevel';         value='0.991';     output;
id= '$5';    property='LeafSize';         value='13';           output;
id= '$5';    property='SplitSize';         value='15';           output;
id= '$5';    property='MinCatSize';     value='11';           output;
id= '$5';    property='MaxBranch';         value='57';        output;
id= '$5';    property='MaxDepth';         value='5';           output;
id= '$5';    property='NRules';         value='18';           output;
id= '$5';    property='NSurrs';         value='5';           output;
id= '$5';    property='MissingValue';     value='BIGBRANCH'; output;
id= '$5';    property='UseVarOnce';     value='Y';            output;
id= '$5';    property='Subtree';         value='LARGEST';   output;
id= '$5';    property='Nsubtree';         value='1';           output;
id= '$5';    property='AssessMeasure';     value='MISC';      output;
id= '$5';    property='AssessPercentage'; value='0.5';        output;
id= '$5';    property='VarSelection';     value='N';           output;
id= '$5';    property='NodeSample';     value='32766';     output;
id= '$5';    property='Exhaustive';     value='1000';      output;
id= '$5';    property='Kass';             value='N';           output;
id= '$5';    property='KassApply';         value='After';     output;
id= '$5';    property='Depth';             value='N';            output;
id= '$5';    property='Inputs';         value='Y';           output;
id= '$5';    property='NumInputs';         value='7';           output;
id= '$5';    property='NodeRole';         value='INPUT';     output;
```

```
    /* Model Comparison Properties */

    id= '$6';    property='DecileBin';               value='25';        output;
    id= '$6';    property='LiftEpsilon';           value='2e-5';      output;
    id= '$6';    property='ProfitEpsilon';           value='2e-4';      output;
    id= '$6';    property='ScoreDistBin';            value='15';        output;
    id= '$6';    property='RocEpsilon';            value='1e-4';      output;
    id= '$6';    property='ModelSelection; value='ManualSelection';  output;
    id= '$6';    property='SelectionStatistic';     value='PROFITLOSS'; output;


    run;
```

This code snippet creates a SAS Node Properties component data set called BoxerNodeprops. This node properties component data set is stored in the BoxerTest library. Using two-level libnames, you could call the SAS Node Properties component data set BoxerTest.BoxerNodeprops.

Completed Batch Code

The batch code below is assembled from the previous example steps. This example is a complete batch processing flow that was created using data step coding and the %EM5BATCH macro.

```
/* Define the Library Path to the Definition Data Sets */

libname BoxerTest 'c:\EMBoxer';


/* Define the Data Source Data Set */

libname SAMPSIO 'c:\Program Files\SAS 9.4\dmine\sample';
options fmtsearch=(sampsio.emfmt);
%let data=SAMPSIO.HMEQ;
%let target=BAD;
%let binary=BAD;
run;


/*  Define the Workspace Data Set  */

data BoxerWorkspace;
length property $64 value $100;
property= 'PROJECTLOCATION'; value= 'c:\Boxer';        output;
property= 'PROJECTNAME';     value= 'BoxerTest';    output;
property= 'WORKSPACENAME';   value= 'EMWS99';        output;
property= 'SUMMARYDATASET';  value= 'BoxerSummary'; output;
property= 'NUMTASKS';           value= 'SINGLE';         output;
property= 'DEBUG';             value= 'LOG OUTPUT';   output;
run;


/* Create Nodes Data Set  */

data BoxerNodes;
```

```
length id $12 component $32 description $64;
id='$1';  component='DataSource';   description='HomeEquity';   output;
id='$2';  component='Sample';       description='Sample';         output;
id='$3';  component='Partition';    description='Partition';    output;
id='$4';  component='Regression';   description='Reg';            output;
id='$5';  component='DecisionTree'; description='Tree';            output;
id='$6';  component='ModelCompare'; description='ModelCompare'; output;
run;


/* Create the Actions to Run Data Set */

data BoxerActions;
length id $12 action $40;
input id action;
cards;
  $6 run;
run;


/* Create the Node Connections Data Set */

data BoxerConnect;
length from to $12;
input from to;
cards;
  $1 $2
  $2 $3
  $3 $4
  $3 $5
  $4 $6
  $5 $6
;
run;


/* Create Temporary Metadata File with the EMCMETA Macro   */

%FILENAME(emcmeta,testmeta,sas,perm=yes,subdir=testsrc);
%INCLUDE testmeta;


/* Create Node Properties Data Set */
/* Remember it is only necessary to specify property settings */
/* that are NOT in default setting for a node. EM assumes all */
/* properties are in default status unless specified in the   */
/* Node Properties Data Set.                                  */

data BoxerNodeprops;
length id $12 property $32 value $64;


/* Data Source Node Properties */

id= '$1';    property='port_data_data';         value="&data";       output;
id= '$1';    property='port_data_columnmeta'; value='testmeta'; output;
```

```
id= '$1';    property='role';                  value='train';    output;


/* Sample Node Properties */

id= '$2';    property='Method';         value='FirstN';    output;
id= '$2';    property='SizeType';     value='Computed';  output;
id= '$2';    property='SizePercent'; value='15';        output;
id= '$2';    property='SizeObs';     value='5';            output;
id= '$2';    property='Alpha';         value='.1';        output;
id= '$2';    property='Pvalue';         value='.05';        output;
id= '$2';    property='AdjustFreq';  value='Y';            output;
id= '$2';    property='StratifyCriterion'; value='Optimal'; output;
id= '$2';    property='IgnoreSmallStrata'; value='Y';     output;


/* Partition Node Properties */

id= '$3';    property='Method';        value='Random';    output;
id= '$3';    property='TrainPct';    value='70';           output;
id= '$3';    property='ValidatePct'; value='10';        output;
id= '$3';    property='TestPct';        value='20';        output;


/* Regression Node Properties */

id= '$4';    property='LinkFunction';     value='Probit';    output;
id= '$4';    property='MinResourceUse'; value='N';         output;
id= '$4';    property='ModelSelection'; value='Stepwise';  output;
id= '$4';    property='SelectionCriterion'; value='VERROR'; output;
id= '$4';    property='Sequential';         value='Y';        output;
id= '$4';    property='SlEntry';         value='0.025';     output;
id= '$4';    property='SlStay';         value='0.14';      output;
id= '$4';    property='Start';             value='1';         output;
id= '$4';    property='Stop';             value='10';        output;
id= '$4';    property='Rule';             value='Multiple';  output;
id= '$4';    property='MaxStep';         value='10';        output;
id= '$4';    property='StepOutput';         value='Y';         output;
id= '$4';    property='OptimizationTechnique';     value='DBLDOG';    output;
id= '$4';    property='ModelDefaults';         value='N';       output;
id= '$4';    property='MaxIterations';         value='55';        output;
id= '$4';    property='MaxFunctionCalls';        value='1005';        output;
id= '$4';    property='MaxTime';         value='51452';     output;
id= '$4';    property='ConvDefaults';           value='N';        output;
id= '$4';    property='AbsFValue';         value='5';        output;
id= '$4';    property='AbsFTime';         value='4';        output;
id= '$4';    property='AbsGValue';         value='0.01';         output;
id= '$4';    property='AbsGTime';         value='10';        output;
id= '$4';    property='AbsXValue';         value='1E-6';        output;
id= '$4';    property='AbsXTime';         value='5';        output;
id= '$4';    property='FConvValue';         value='15';        output;
id= '$4';    property='FConvTimes';         value='6';        output;
id= '$4';    property='GConvValue';         value='2E-5';        output;
id= '$4';    property='GConvTimes';         value='5';        output;
id= '$4';    property='ClParm';         value='Y';        output;
id= '$4';    property='CovB';             value='Y';        output;
```

```
id= '$4';    property='Simple';         value='Y';        output;
id= '$4';    property='Details';         value='Y';        output;
id= '$4';    property='PrintDesignMatrix';     value='Y';        output;


/* Decision Tree Node Properties */

id= '$5';    property='Criterion';         value='CHISQ';     output;
id= '$5';    property='SigLevel';         value='0.991';     output;
id= '$5';    property='LeafSize';         value='13';           output;
id= '$5';    property='SplitSize';         value='15';           output;
id= '$5';    property='MinCatSize';         value='11';       output;
id= '$5';    property='MaxBranch';         value='57';        output;
id= '$5';    property='MaxDepth';         value='5';           output;
id= '$5';    property='NRules';         value='18';           output;
id= '$5';    property='NSurrs';         value='5';           output;
id= '$5';    property='MissingValue';     value='BIGBRANCH'; output;
id= '$5';    property='UseVarOnce';     value='Y';           output;
id= '$5';    property='Subtree';         value='LARGEST';   output;
id= '$5';    property='Nsubtree';         value='1';           output;
id= '$5';    property='AssessMeasure';     value='MISC';      output;
id= '$5';    property='AssessPercentage';    value='0.5';   output;
id= '$5';    property='VarSelection';     value='N';           output;
id= '$5';    property='NodeSample';     value='32766';     output;
id= '$5';    property='Exhaustive';     value='1000';      output;
id= '$5';    property='Kass';             value='N';           output;
id= '$5';    property='KassApply';         value='After';     output;
id= '$5';    property='Depth';             value='N';           output;
id= '$5';    property='Inputs';         value='Y';           output;
id= '$5';    property='NumInputs';         value='7';           output;
id= '$5';    property='NodeRole';         value='INPUT';     output;


/* Model Comparison Properties */

id= '$6';    property='DecileBin';             value='25';        output;
id= '$6';    property='LiftEpsilon';         value='2e-5';      output;
id= '$6';    property='ProfitEpsilon';         value='2e-4';      output;
id= '$6';    property='ScoreDistBin';         value='15';        output;
id= '$6';    property='RocEpsilon';         value='1e-4';      output;
id= '$6';    property='ModelSelection;         value='ManualSelection'; output;
id= '$6';    property='SelectionStatistic';    value='PROFITLOSS';   output;

run;
```

Execute the process flow diagram

The final step in building your batch processing code is to provide execution instructions for SAS to process.

The library path statement that you created in Step 1 points SAS to the location of the batch processing component data sets that you constructed. The %EM5BATCH macro with the EXECUTE option cannot be saved as a data set and must be run each time you wish to execute a batch process flow.

Here is the batch processing macro execution code for the example batch process flow we have been building for this example:

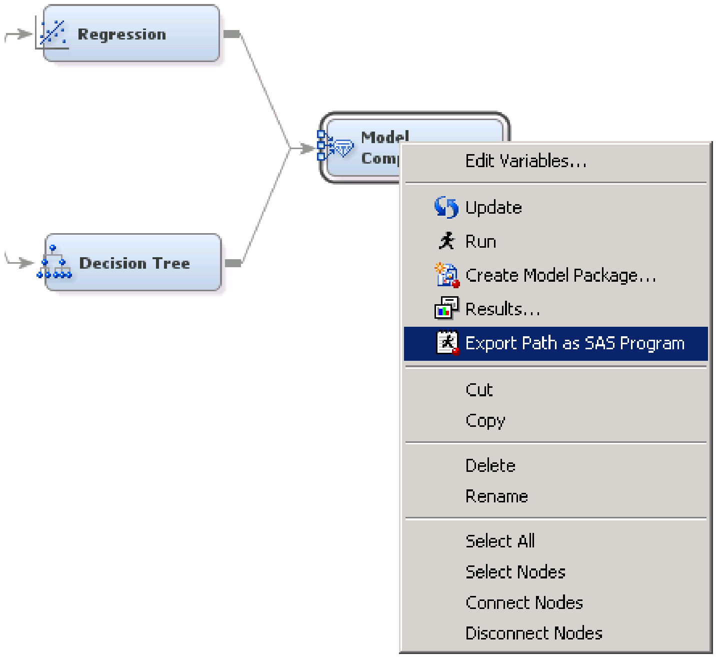
```
%EM5BATCH(execute,
          workspace=BoxerTest.BoxerWorkspace,
          nodes=BoxerTest.BoxerNodes,
          action=BoxerTest.BoxerActions
          connect=BoxerTest.BoxerConnect,
          nodeprops=BoxerTest.BoxerNodeprops)
    ;
    run;
```

Running SAS Enterprise Miner Batch Code

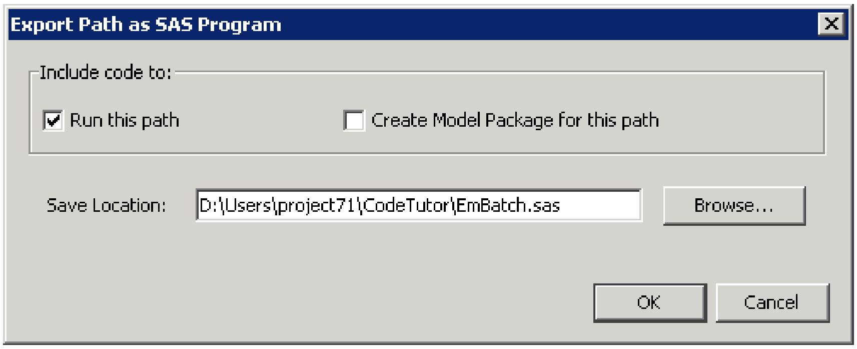
Overview

After building your batch processing code, you can run it outside of SAS Enterprise Miner. Batch code can be run through the SAS Display Management System, SAS Enterprise Guide, or from the command line. If desired, automated execution of batch processing code can be configured through additional SAS software or through standard clients on supported operating systems. In all cases, SAS Enterprise Miner must be licensed on the system where the batch code is executed.

Generally, batch is created by creating a SAS Enterprise Miner process flow diagram and exporting that diagram as SAS code. Experienced uses can write custom batch processing programs. To export a process flow diagram as SAS code, right click the terminal node and select **Export Path as SAS Program**.



In the Export Path as SAS Program window, the batch processing code is saved in the directory specified by the **Save Location** field.

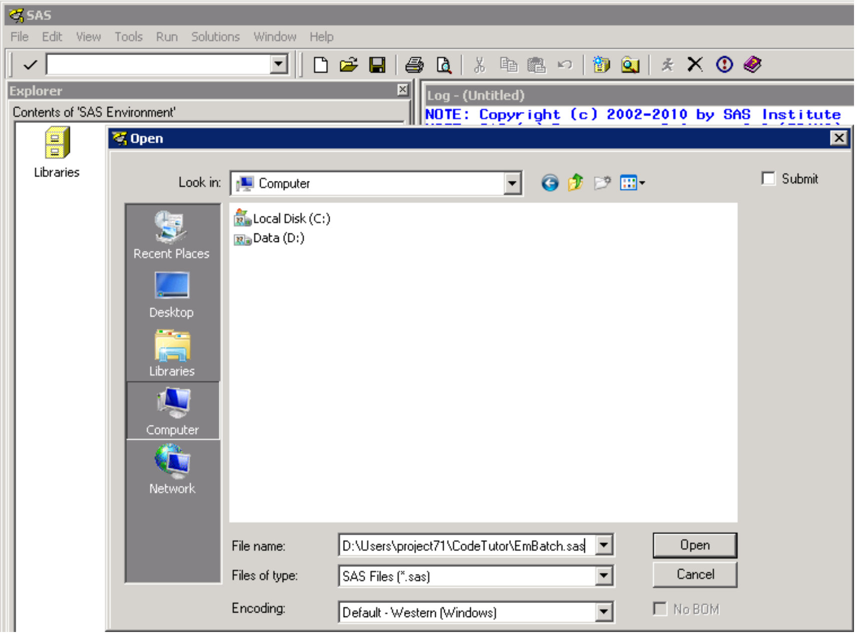


☑ Run this path   □ Create Model Package for this path

Save Location: D:\Users\project71\CodeTutor\EmBatch.sas

Running SAS Enterprise Miner Batch Code from the SAS Display Management System

Before you can run your exported code, you must close the SAS Enterprise Miner project that was used to create the code. To execute the batch processing code in the SAS Display Management System, select **File** ⇨ **Open** and navigate to your batch processing program.



NOTE: Copyright (c) 2002-2010 by SAS Institute

Look in: Computer

□ Submit

File name: D:\Users\project71\CodeTutor\EmBatch.sas

Files of type: SAS Files (*.sas)

Encoding: Default - Western (Windows)

□ No BOM

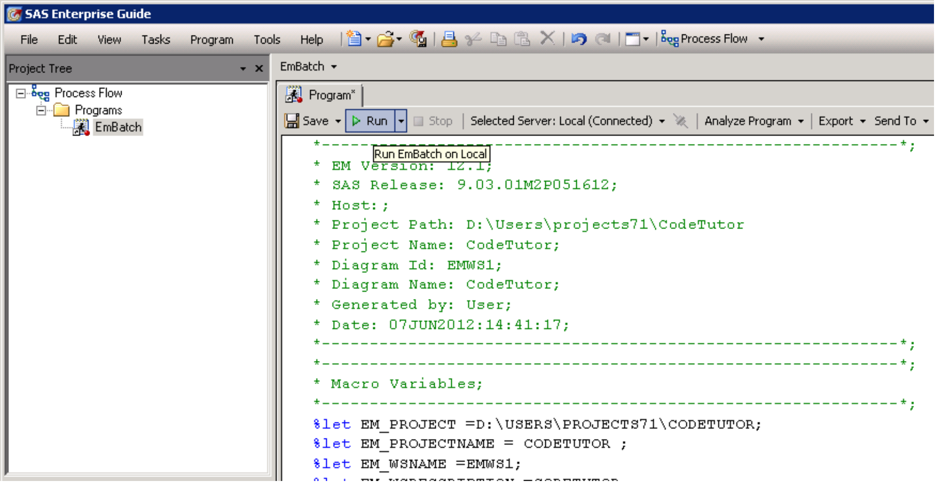
The batch processing program is submitted like a typical SAS program, by selecting **Run** ⇨ **Submit**. Depending on your batch code, the output is sent either to a temporary SAS work library or to a predefined location. In either case, the exact location of all output is available at the bottom of the SAS Log produced by the batch code in lines similar to the following:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS
```

Running SAS Enterprise Mine Batch Code from SAS Enterprise Guide

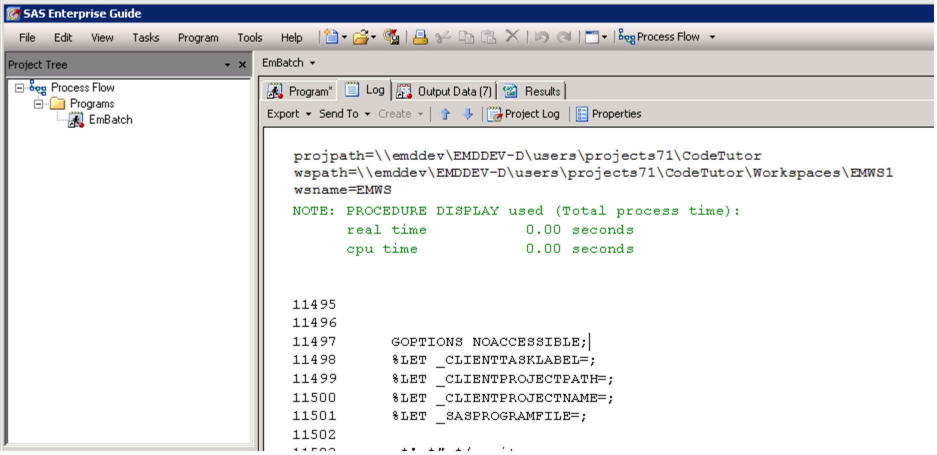
Before you can run your exported code, you must close the SAS Enterprise Miner project that was used to create the code. To execute the batch processing code in SAS Enterprise Guide, select **File** ⇨ **Open** and navigate to your batch processing program.

Run the batch processing program by selecting **File** ⇨ **Run**, **Prgraom** ⇨ **Run**, or by clicking the **Run** button in the **Program** tab.



```
*------------------------------------------------------------*;
* EM Version: 12.1;
* SAS Release: 9.03.01M2P051612;
* Host: ;
* Project Path: D:\Users\projects71\CodeTutor
* Project Name: CodeTutor;
* Diagram Id: EMWS1;
* Diagram Name: CodeTutor;
* Generated by: User;
* Date: 07JUN2012:14:41:17;
*------------------------------------------------------------*;
*------------------------------------------------------------*;
* Macro Variables;
*------------------------------------------------------------*;
%let EM_PROJECT =D:\USERS\PROJECTS71\CODETUTOR;
%let EM_PROJECTNAME = CODETUTOR ;
%let EM_WSNAME =EMWS1;
```

After the program executes, the **Log**, **Output Data**, and **Results** tabs appear. Several tables are generated on the SAS Enterprise Guide process flow. While these new tabs and tables contain information pertinent to the SAS Enterprise Miner batch program, none contain the actual results of the batch program run. If the run completes successfully, the log must be consulted to find the location of the results.



```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


11495
11496
11497          GOPTIONS NOACCESSIBLE;
11498          %LET _CLIENTTASKLABEL=;
11499          %LET _CLIENTPROJECTPATH=;
11500          %LET _CLIENTPROJECTNAME=;
11501          %LET _SASPROGRAMFILE=;
11502
```

Running SAS Enterprise Miner Batch Code from the Command Line

Exported batch processing code can be run from the command line using the SYSIN option. First navigate to the SAS Foundation directory where the primary SAS executable is located. Once there, start SAS using the SYSIN option followed by the fully qualified name of the batch processing code file. The example code below runs the batch processing code without starting a graphical interface on Windows.

```
Z:\>cd "E:\Program Files\SASHome\SASFoundation\9.4\"
E:\Program Files\SASHome\SASFoundation\9.4>sas.exe
-SYSIN D:\Users\projects71\CodeTutor\EmBatch.sas
```

On most Unix platforms, the example code below executes the SAS Enterprise Miner batch processing code without starting a graphical interface.

```
$ cd /install/cfgsas1/SASHome/SASFoundation/9.4/
$ ./sas -SYSIN /users/project71/CodeTutor/EmBatch.sas
```

On both Windows and Unix, a log file detailing the attempted run is created in the directory from where SAS command is invoked. Consult this log file to determine if the batch code executed as expected. The location of any output is included near the end of the log file, just as in the SAS Log in the SAS Display Management System or SAS Enterprise Guide. Note that the exact location of your SAS installation and exported SAS Enterprise Miner batch programs are most likely different from those in the given examples.

Direct questions regarding the location of SAS system applications and automating SAS Enterprise Miner batch processing code to the appropriate IT administrator or technical support group.

See Also

Part II: "Batch Processing Code Examples" on page 1355

Part III: "SAS Enterprise Miner Batch Processing Appendix" on page 1429

Chapter 85
Batch Processing Code Examples

Batch Processing Code Examples

Overview of SAS Enterprise Miner Batch Processing Code Examples

The batch processing code examples in this document should be completed sequentially. The batch processing examples illustrate data mining tasks in a client/server environment. The data mining tasks include creating and modifying SAS Enterprise Miner projects, project workspaces, and process flow diagrams. In the example process flows you create new data sources as well as use existing ones, create and modify target profiles, add modeling nodes, compare data models, and generate data mining reports. As you complete successive examples, you will learn how batch processing code can be created and modified using both the SAS Program Editor and the SAS Enterprise Miner 12.3 Graphic User Interface (GUI).

The examples also discuss the SAS Enterprise Miner 12.3 project file structures to see how their contents change during successive data mining process flow runs. The examples show the location of the DataSource, Reports, and Workspaces folders in a SAS Enterprise Miner project folder, as well as the files within those folders.

The final example begins with a GUI process flow diagram that you build and run, then retrieve exported batch processing code from the Results window. Then, you exit SAS Enterprise Miner 12.3 and use the SAS Program Editor to modify your exported batch processing code to create a new project and generate reports. After you run your batch code and generate the reports, you modify the code to change the reports that you created. Finally, you use the SAS Enterprise Miner 12.3 GUI to open and view the project and reports that you created and modified in the batch processing environment.

Restrictions to Batch Processing Code Submitted to the SAS Enterprise Miner Code Editor

The SAS Enterprise Miner 12.3 batch processing code in these examples should be submitted in a SAS batch job or submitted through the Program Editor that is a part of base SAS. The SAS Enterprise Miner 12.3 GUI includes a Program Editor window, but you can not submit batch processing code though any part of the SAS Enterprise Miner 12.3 GUI, including the Program Editor window.

Batch Processing Code Data Set Caching

The %EM5BATCH macro caches the last data sets that were specified by creating the macro variables. If component data sets are not explicitly specified in the %EM5BATCH macro, it retrieves the most recent table that was specified for each component data set.

For example, if you submit the statement

```
%EM5BATCH(execute,
      workspace=work.workspace,
      nodeprops=work.nodeprops,
      action=work.actions,
      nodes=work.nodes,
      connect=work.connect,
      datasources=
      );
```

and then you submit the following statement

```
%EM5BATCH(execute,
      nodeprops=work.nodeprops,
      action=work.actions
      );
```

The %EM5BATCH macro will use the previously defined work.workspace, work.nodes, and work.connect arguments.

If you don't want to use the previously defined arguments, you can either use PROC DATASETS to delete the old data sets that you do not want to be used, or submit the macro with blank arguments to prevent the most recent data from being used, as shown below:

```
%EM5BATCH(execute,
      workspace=work.workspace,
      nodeprops=work.nodeprops,
      action=work.actions,
      nodes=,
      connect=,
      datasources=
      );
```

Examples

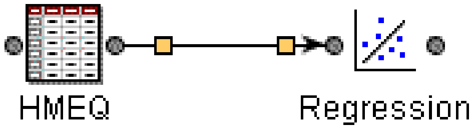
Example 1: Simple Regression Example Using a Temporary Project Space and Workspace

Example 1 uses SAS Enterprise Miner batch processing to run a simple data mining process flow in a temporary workspace and temporary project space. The data source is created from data that is not associated with an existing SAS Enterprise Miner project. After the batch processing code runs, output data structures and results can viewed in the EMProject folder that is found in the temporary Work library.

Example 1 is purely a batch processing code example and does not require the SAS Enterprise Miner GUI to create it. However, it is helpful to visualize the equivalent of the GUI process flow diagram that performs the same tasks as the batch processing code examples. An overview of the process flow diagram to be created and an overview of the node properties follows to help new users. It is not necessary to construct the process flow diagram in the GUI for the purposes of this example.

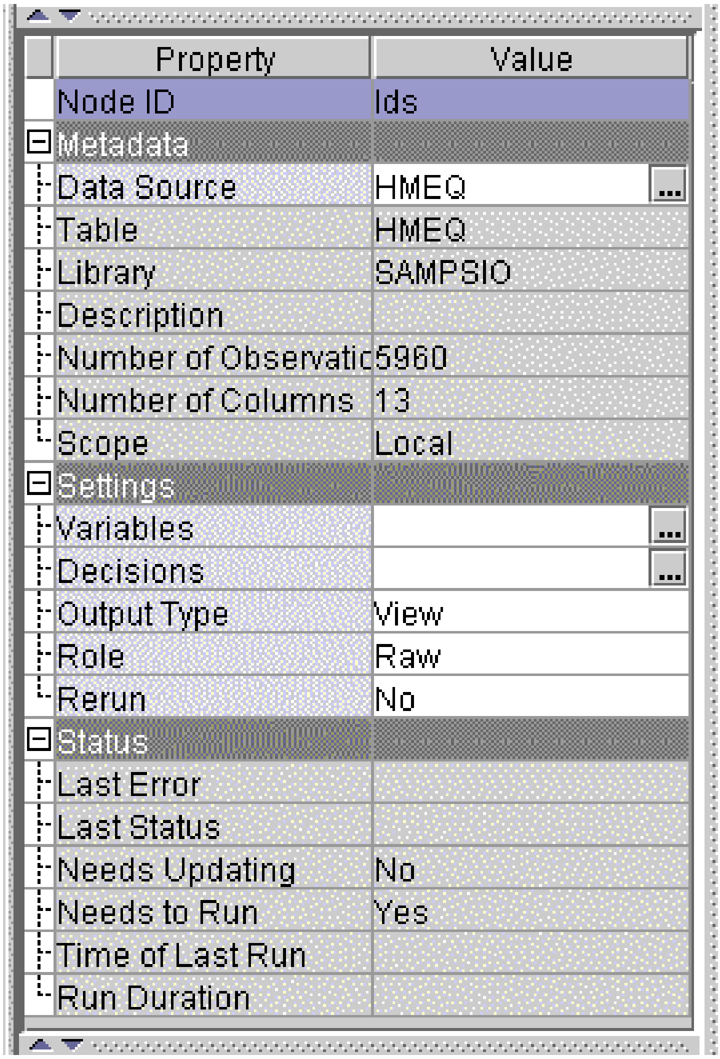
Example 1 Diagram Overview

If the Example 1 process flow were created in the SAS Enterprise Miner GUI, it would look like the following process flow diagram:



Example 1 IDS Node Properties Overview:

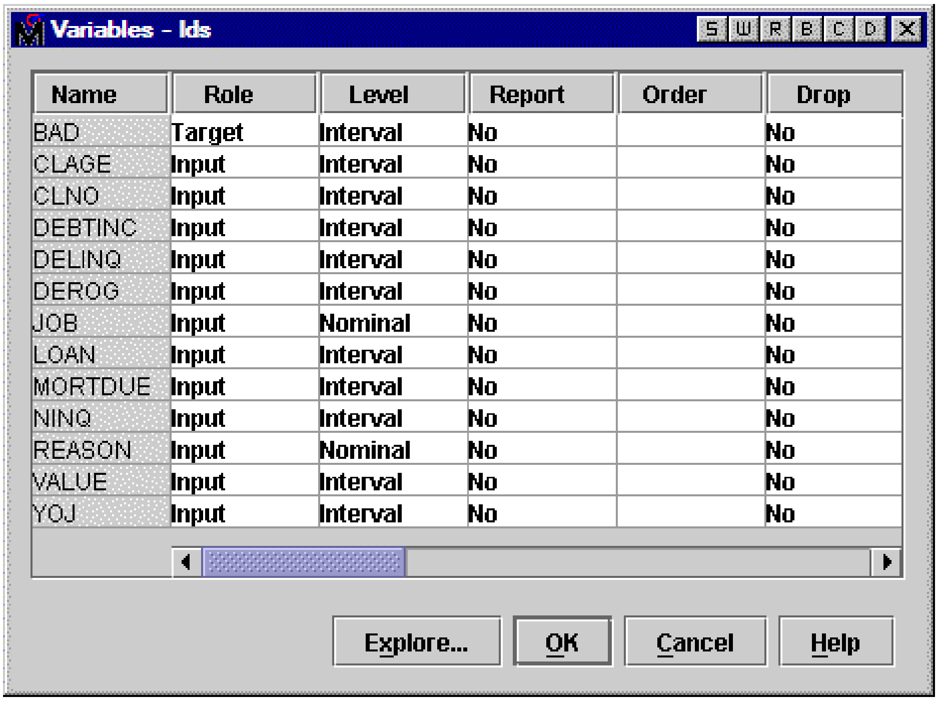
If the Example 1 process flow were created in the SAS Enterprise Miner GUI, the properties panel for the HMEQ data source would look similar to the following:



Property	Value
Node ID	Ids
⊟ Metadata	
Data Source	HMEQ ...
Table	HMEQ
Library	SAMPSIO
Description	
Number of Observatic	5960
Number of Columns	13
Scope	Local
⊟ Settings	
Variables	...
Decisions	...
Output Type	View
Role	Raw
Rerun	No
⊟ Status	
Last Error	
Last Status	
Needs Updating	No
Needs to Run	Yes
Time of Last Run	
Run Duration	

Example 1 Data Source Properties

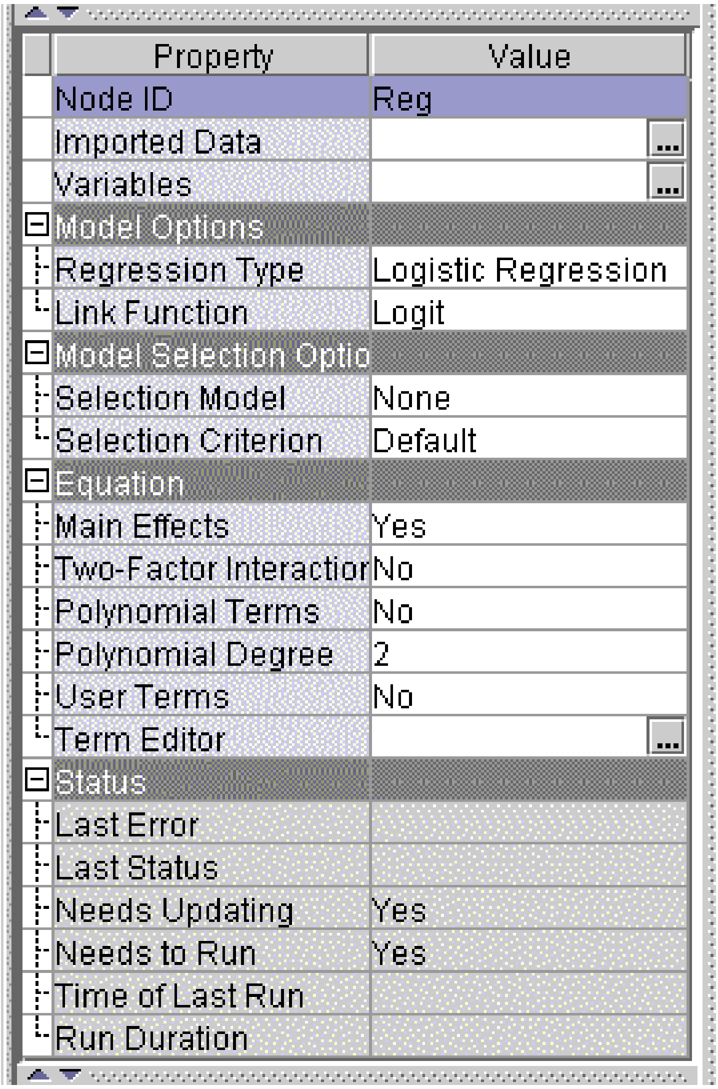
In the Column Metadata subgroup, click the ... icon to the right of the Variables property to display a variables table that shows the HMEQ variables. You can see the BAD variable set as the target variable.



Name	Role	Level	Report	Order	Drop
BAD	Target	Interval	No		No
CLAGE	Input	Interval	No		No
CLNO	Input	Interval	No		No
DEBTINC	Input	Interval	No		No
DELINQ	Input	Interval	No		No
DEROG	Input	Interval	No		No
JOB	Input	Nominal	No		No
LOAN	Input	Interval	No		No
MORTDUE	Input	Interval	No		No
NINQ	Input	Interval	No		No
REASON	Input	Nominal	No		No
VALUE	Input	Interval	No		No
YOJ	Input	Interval	No		No

Example 1 Regression Node Properties Overview:

If the Example 1 process flow were created in the SAS Enterprise Miner GUI, the properties panel for the Regression node would appear similar to what is shown below. Because the Regression node in the process flow is run using the default settings, it is not necessary to declare any of the Regression node properties in the batch processing code.



Property	Value
Node ID	Reg
Imported Data	...
Variables	...
⊟ Model Options	
Regression Type	Logistic Regression
Link Function	Logit
⊟ Model Selection Optio	
Selection Model	None
Selection Criterion	Default
⊟ Equation	
Main Effects	Yes
Two-Factor Interactior	No
Polynomial Terms	No
Polynomial Degree	2
User Terms	No
Term Editor	...
⊟ Status	
Last Error	
Last Status	
Needs Updating	Yes
Needs to Run	Yes
Time of Last Run	
Run Duration	

Example 1 Regression Properties

The settings for the GUI process flow diagram displayed above are re-created by the Example 1 batch processing code below.

Example 1 Output Location:

The Example 1 batch processing code creates a temporary project in the EmProject folder, located within the SAS Temporary Files folder. Example 1 in this document was run under Windows XP, where the default SAS temporary project path is `C:\Documents and Settings\username\Local Settings\Temp\SAS Temporary Files\_TD1772\EmProject`

Your temporary project path will vary according to your operating environment and SAS installation defaults. The folder 'username' and the folder '_TD1772' will have different names if you run the example. You can find your temporary project path in the SAS output log after you run your batch processing code.

Example 1 Batch Processing Code:

The Example 1 batch processing code does not use a data source or metadata from an existing project. The data source is identified and the metadata is created through the SAS Enterprise Miner batch processing environment. The target variable is defined by using the EM_VARIABLEATTRIBUTES property to modify the default metadata.

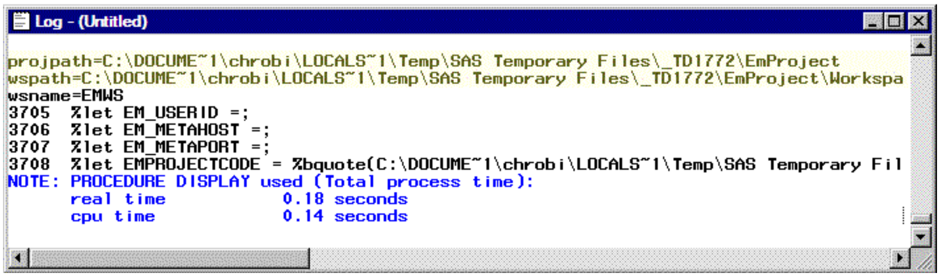
The examples that follow should be submitted in a SAS batch job or submitted through the Program Editor in base SAS.

```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner Batch Processing Example 1***     ;
* Use %let statements to initialize vars for summary data set ;
* name and library, specify single thread execution, use the  ;
* Replace edit mode.                                          ;
*------------------------------------------------------------*;
%let EM_PROJECT = ;
%let EM_PROJECTNAME = ;
%let EM_WSNAME = ;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = SINGLE;
%let EM_EDITMODE = R;
%let EM_DEBUG = ;
%let EM_ACTION = RUN;

*------------------------------------------------------------*;
* Create Workspace data set and populate the property values  ;
* using the variables created by the %let statements above    ;
*------------------------------------------------------------*;
data workspace;
length property $64 value $100;

property= 'PROJECTLOCATION'; value= "<&EM_PROJECT>";     output;
property= 'PROJECTNAME';     value= "<&EM_PROJECTNAME>"; output;
property= 'WORKSPACENAME';   value= "<&EM_WSNAME>";      output;
property= 'SUMMARYDATASET';  value= "<&EM_SUMMARY>";     output;
property= 'NUMTASKS';        value= "<&EM_NUMTASKS>";    output;
property= 'EDITMODE';        value= "<&EM_EDITMODE>";    output;
property= 'DEBUG';           value= " ";               output;

run;

*------------------------------------------------------------*;
* Create Nodes data set                                       ;
* Create GUI text labels of "Regression" for the Regression   ;
* node and "HMEQ" for the Input Data node                     ;
*------------------------------------------------------------*;
data nodes;
length id $12 component $32 description $64;
id= "Reg"; component="Regression"; description= "Regression"; output;
id= "Ids"; component="DataSource"; description= "HMEQ";       output;

run;

*------------------------------------------------------------*;
* Specify Variable Attributes for the Data Source in lib WORK ;
* Assign the HMEQ variable 'BAD' the role of target variable  ;
* and store the information in Ids_VariableAttribute          ;
*------------------------------------------------------------*;
data WORK.Ids_VariableAttribute;

length Variable $64 AttributeName $32 AttributeValue $64;
Variable="BAD"; AttributeName="ROLE"; AttributeValue="TARGET"; Output;
```

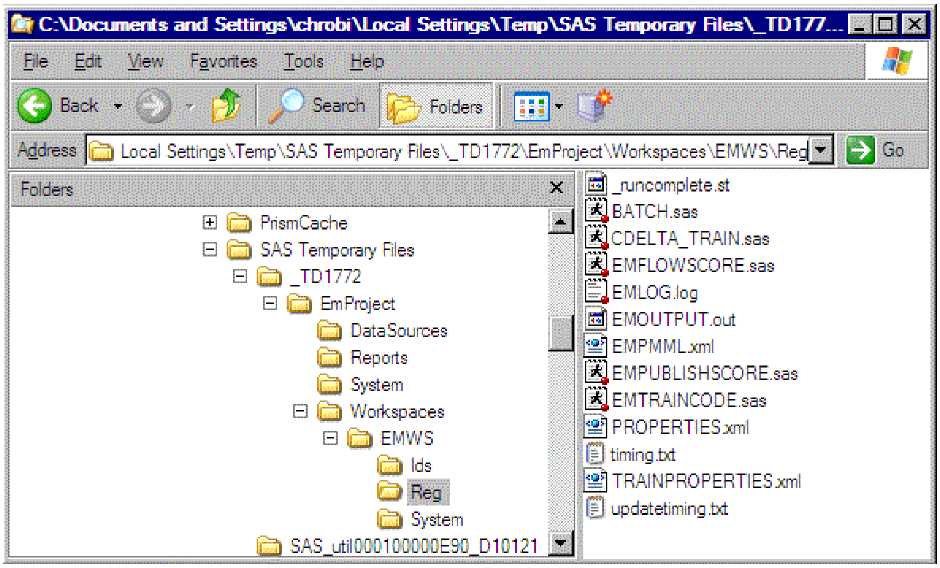
```
run;

*------------------------------------------------------------*;
* Create node properties data set for the Data Source node    ;
* Specify the HMEQ data set from the SAMPSIO library as the   ;
* data source. Read Ids_VariableAttribute in the WORK library ;
* so EM_VARIABLEATTRIBUTES can define BAD as the target       ;
* variable. The Regression node uses all default property     ;
* settings, so no declarations are required for the Regression;
* node.                                                       ;
*------------------------------------------------------------*;
data nodeprops;
length id $12 property $32 value $64;
id= "Ids"; property="PORT_DATA_DATA";        value= "SAMPSIO.HMEQ"; output;
id= "Ids"; property="EM_VARIABLEATTRIBUTES"; value= "WORK.Ids_VariableAttribute";
output;

run;

*------------------------------------------------------------*;
* Create Connections data set                                 ;
* Data Source connects to Regression.                         ;
*------------------------------------------------------------*;
data connect;
length from to $12;
input from to;
cards;
   Ids Reg
   ;

run;

*------------------------------------------------------------*;
* Create Actions to run the process flow                      ;
* Run the flow from the Regression node     Ids --> Reg.      ;
*------------------------------------------------------------*;
data actions;
length id $12 action $40;

id="Reg";  action='run';  output;

run;

*------------------------------------------------------------*;
* Execute the Actions                                         ;
* Use the %EM5BATCH macro and call the component data sets    ;
* that were created from the DATA step code in this example   ;
*------------------------------------------------------------*;
%EM5BATCH(execute,
    workspace=workspace,
    nodes=nodes,
    connect=connect,
    nodeprops=nodeprops,
    action=actions
    );
```

```
run;
```

After you run the Example 1 batch processing code, examine the SAS log. Find the path to your SAS Temporary Files directory where the Example 1 project structure was created. The display below uses a yellow background to call attention to the temporary project and workspace paths. Your SAS log will not show yellow backgrounds behind the project and workspace paths.



```
projpath=C:\DOCUME~1\chrobi\LOCALS~1\Temp\SAS Temporary Files\_TD1772\EmProject
wspath=C:\DOCUME~1\chrobi\LOCALS~1\Temp\SAS Temporary Files\_TD1772\EmProject\Workspa
wsname=EMWS
3705  %let EM_USERID =;
3706  %let EM_METAHOST =;
3707  %let EM_METAPORT =;
3708  %let EMPROJECTCODE = %bquote(C:\DOCUME~1\chrobi\LOCALS~1\Temp\SAS Temporary Fil
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           0.18 seconds
      cpu time            0.14 seconds
```



You can view the contents of your temporary project by using a file utility. Note that the DataSources folder is empty since the batch processing code did not use a pre-existing SAS Enterprise Miner data source. Because the batch processing code did not create reports, the Reports folder is empty. The Workspaces folder contains the EMWS workspace where you can browse the various node outputs from Example 1.

The output files in the node folders under a project contain a variety of information.

For example, the EMOUTPUT.out file in the SAS Temporary Files_TD1772\EMWS\Reg folder contains the following information about the results of running the Regression node:

- Variable Summary
- Predicted and Decision Variables
- Analysis of Variance
- Model Fit Statistics
- Target Fit Statistics
- Analysis of Maximum Likelihood Estimates

- Type 3 Analysis of Effects
- Regression Model Information
- Assessment Score Rankings
- Assessment Score Distribution

Example 2: Simple Regression Example Using an Existing Project and Creating a Workspace

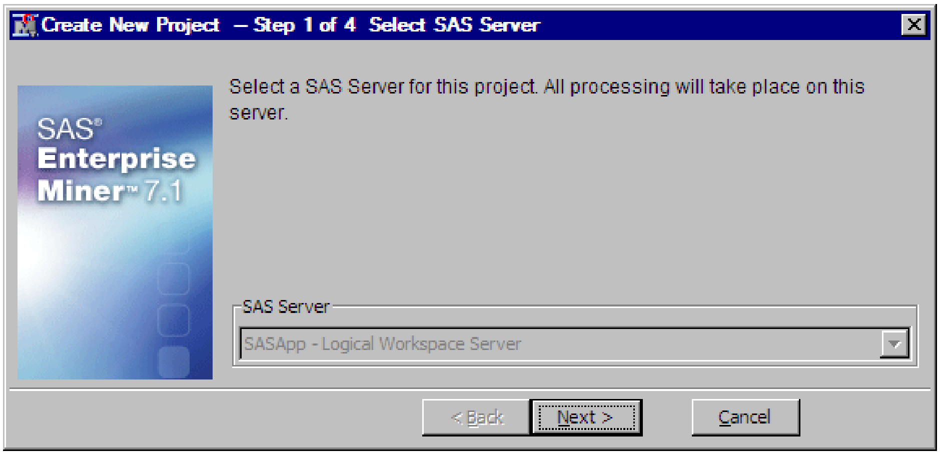
Examples 2 through 6 are batch processing code examples that build on the previous examples and increase progressively in complexity. Example 2 is the first example in the series.

Example 2 creates a simple process flow that uses a pre-existing SAS Enterprise Miner data source that was created in an existing GUI project. For the purposes of the example, you must create the "existing" project called CodeTutor and its data source using the SAS Enterprise Miner GUI. Then the Example 2 batch processing code creates its own workspace and modifies the existing CodeTutor project. Data sources that were created in the SAS Enterprise Miner GUI or in an earlier batch processing session are interchangeable and can be reused in future batch processing code.

Create the Example 2 GUI Project CodeTutor

To create the "existing" CodeTutor project that the Example 2 code will modify, open the SAS Enterprise Miner GUI. Select **File** ⇨ **New** ⇨ **Project** from the SAS Enterprise Miner main menu to open the Create New Project wizard.

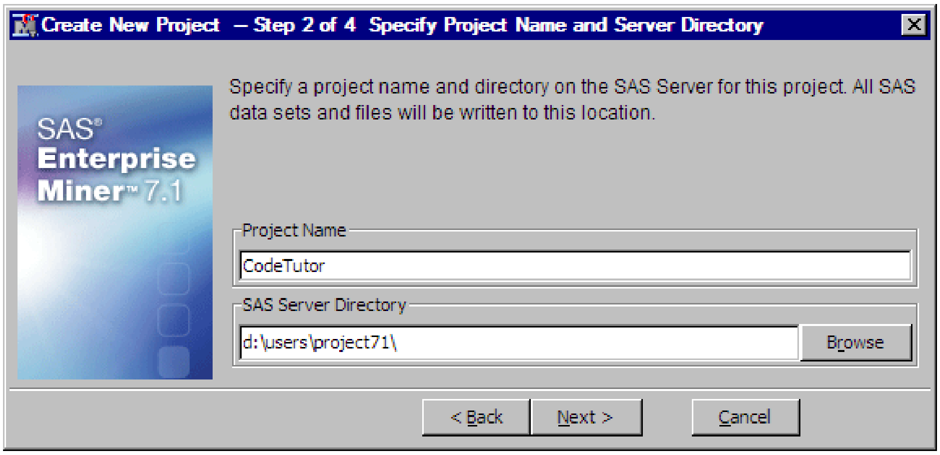
First, select a SAS server, or accept the default SAS Server that your administrator has configured.


Create New Project – Step 1 of 4 Select SAS Server

Select a SAS Server for this project. All processing will take place on this server.

SAS Server: SASApp - Logical Workspace Server

Next, specify CodeTutor as your **Project Name**, and then specify the server location that you will use to store your CodeTutor project files.



Create New Project – Step 2 of 4 Specify Project Name and Server Directory

Specify a project name and directory on the SAS Server for this project. All SAS data sets and files will be written to this location.

Project Name: CodeTutor

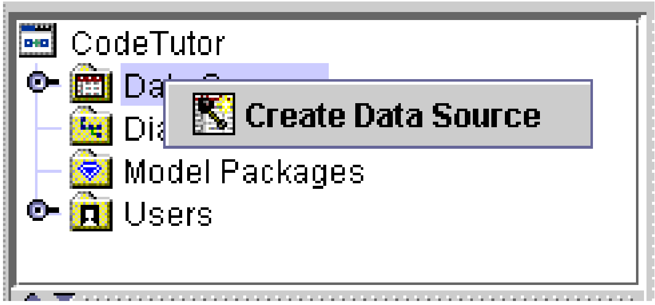
SAS Server Directory: d:\users\project71\

The project path in the illustration specifies a location on the SAS server that you selected in the last step. The project path does not point to the D:\ drive of the client computer that is connected to the SAS server. When your batch processing code needs to specify the project path, you cannot use the path that is local to the SAS server. You will need to change the server's local project path to a global network path. The global network path points to the same address on the SAS server, but it is understood by all members of the network. In this example, the local SAS server project path resolves into the global path: **`\\emddev\EMDDEV-D\users\projects71`**.

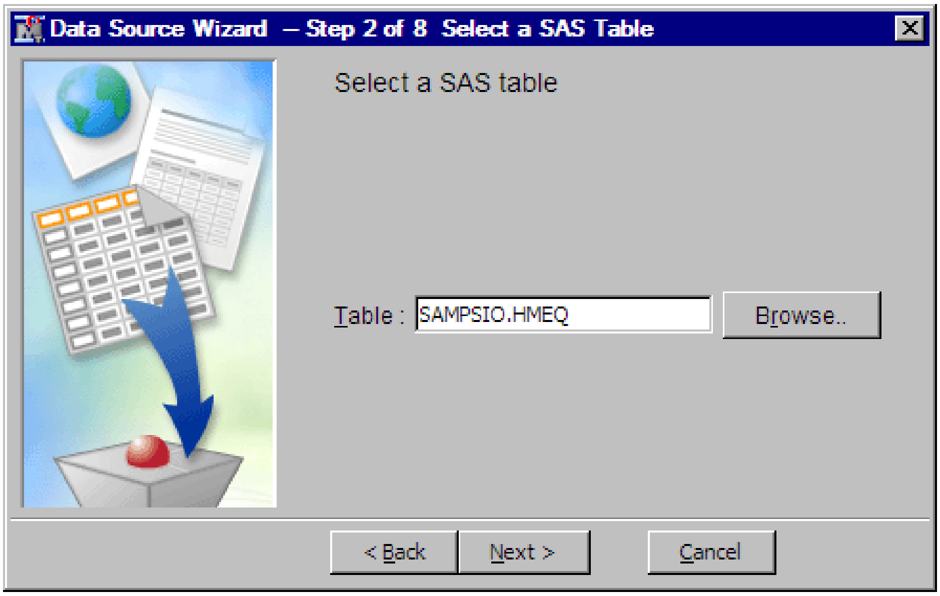
Your SAS server project path and global network project path will be different from the examples above. You should substitute your own global project path everywhere that the example path above appears.

Create and Configure the Home Equity Data Source for the CodeTutor Project

After you create the CodeTutor project using the SAS Enterprise Miner GUI, use the Project Panel to add the Home Equity data source to the project. Right-click the Data Sources folder and select Create Data Source.



Use the Data Source wizard to select the default SAS Table metadata source, select **Next**, then type SAMPSIO.HMEQ to specify your table, and click **Next**.



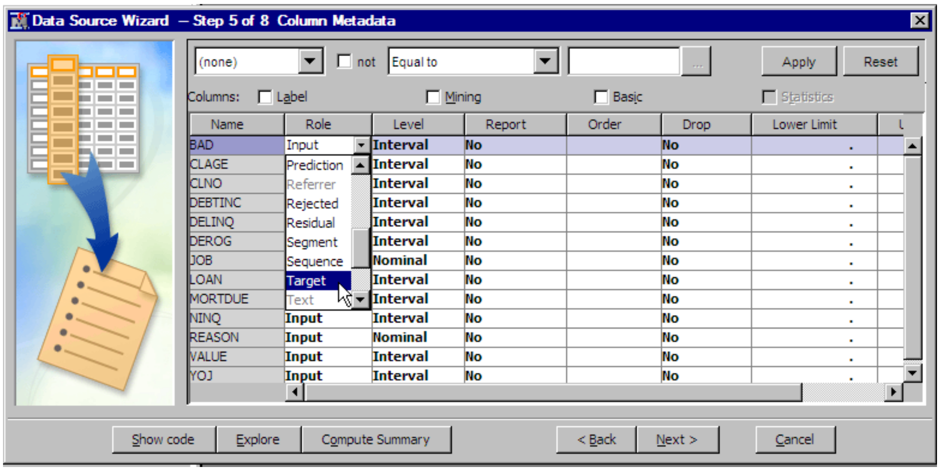
Data Source Wizard – Step 2 of 8 Select a SAS Table

Select a SAS table

Table : SAMPSIO.HMEQ

Step 3 of the Data Source Wizard displays your table properties. Select **Next**, and then select **Next** again to accept the default (Basic) Metadata Advisor Option setting in step 4.

Use the Column Metadata window of step 5 to set the BAD variable to the role of Target, then select **Next**.



Data Source Wizard – Step 5 of 8 Column Metadata

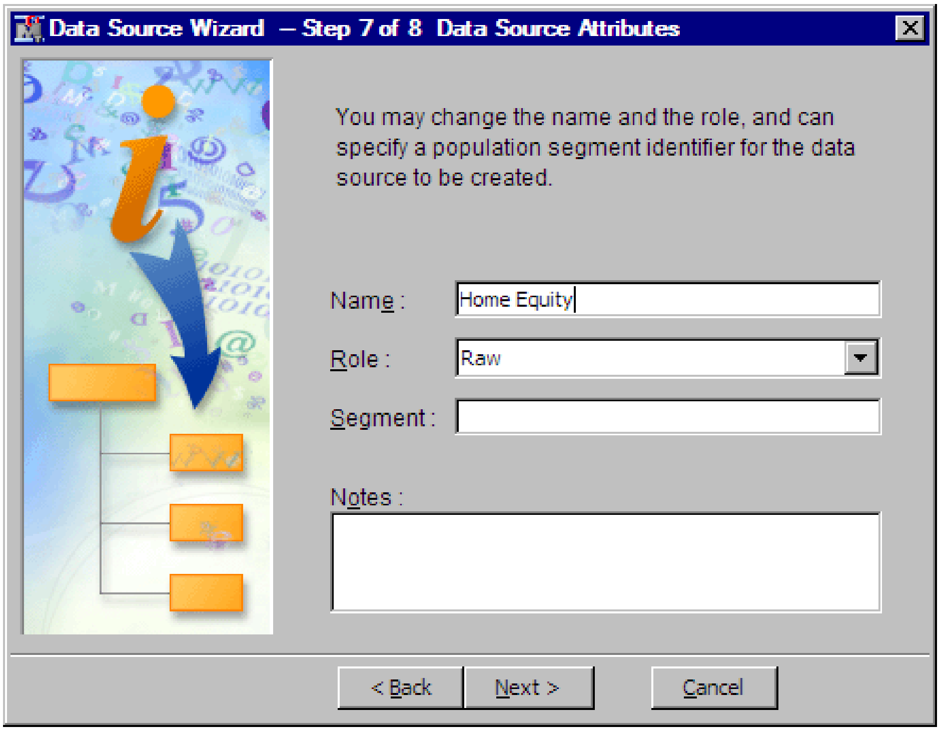
(none) □ not Equal to

Columns: □ Label □ Mining □ Basic □ Statistics

Name	Role	Level	Report	Order	Drop	Lower Limit	L
BAD	Input	Interval	No		No	.	
CLAGE	Prediction	Interval	No		No	.	
CLNO	Referrer	Interval	No		No	.	
DEBTINC	Rejected	Interval	No		No	.	
DELINQ	Residual	Interval	No		No	.	
DEROG	Segment	Interval	No		No	.	
JOB	Sequence	Nominal	No		No	.	
LOAN	Target	Interval	No		No	.	
MORTDUE	Text	Interval	No		No	.	
NINQ	Input	Interval	No		No	.	
REASON	Input	Nominal	No		No	.	
VALUE	Input	Interval	No		No	.	
YOJ	Input	Interval	No		No	.	

Select **Next** again to accept the default values creating a data sample in step 6. This exercise does not require a sample data set.

Use the Data Source Attributes in step 7 to change your data source label to Home Equity instead of the default table name HMEQ. This helps make the contents of your projects’s Data Sources folder a little easier to view. Then select **Next**.



Data Source Wizard — Step 7 of 8 Data Source Attributes

You may change the name and the role, and can specify a population segment identifier for the data source to be created.

Name : Home Equity

Role : Raw

Segment :

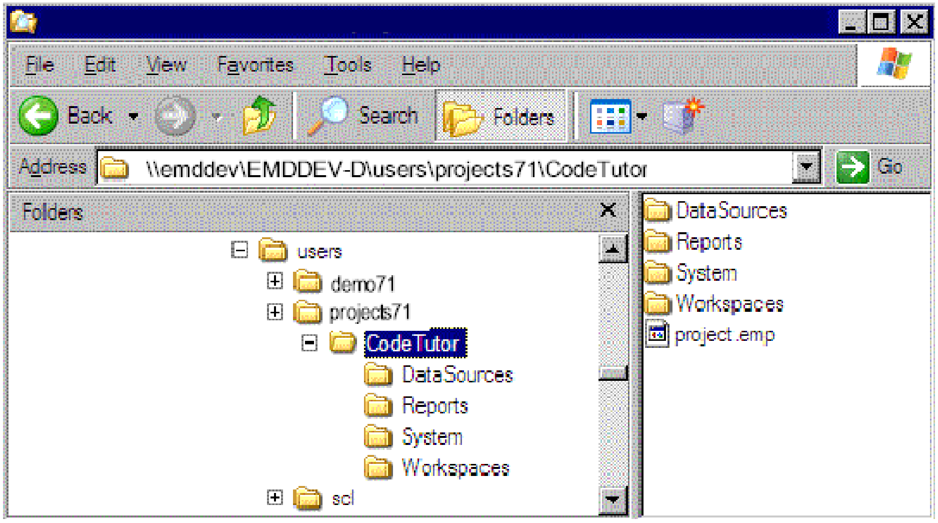
Notes :

View the Data Source Wizard summary in step 8, then select Next to close the Data Source Wizard.

After you create the CodeTutor project and add your HomeEquity data source, close SAS Enterprise Miner 12.3. Do not construct any process flow diagrams in the GUI. You have created an empty project at this point. The project file structure is created on the server, but there is no workspace and no process flow diagram. The Example 2 batch processing code will create the workspace and the process flow for the empty project.

View the Example 2 CodeTutor Project Files

Use a file utility to examine the CodeTutor project that you created on the SAS server. If you follow the project path on the server, your project structure should resemble the following:



Address \\emddev\EMDDEV-D\users\projects71\CodeTutor

```
users
  demo71
  projects71
    CodeTutor
      DataSources
      Reports
      System
      Workspaces
  scl
```

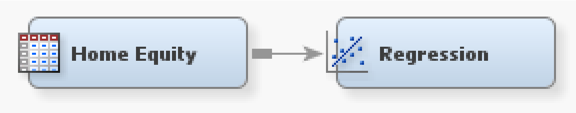
```
DataSources
Reports
System
Workspaces
project.emp
```

The DataSources folder should contain files for the HomeEquity data source. The Reports and Workspaces folders are empty now.

As in Example 1, Example 2 creates a process flow using only batch processing code. However, it is helpful to visualize the equivalent of the batch processing code flow as it would appear if it were created in the GUI. Therefore, an overview of the process flow diagram to be created and an overview of the node properties follows. It is not necessary to construct the process flow diagram below in the GUI for this example.

Example 2 Diagram Overview:

Example 2 uses batch processing code to create a regression process flow inside an existing SAS Enterprise Miner project. If the Example 2 process flow were created in the SAS Enterprise Miner GUI, it would look like the process flow diagram that is shown below:



Example 2 IDS Node Properties Overview:

The property settings for the Home Equity data source as created above do not require further configuration.

Example 2 Regression Node Properties Overview:

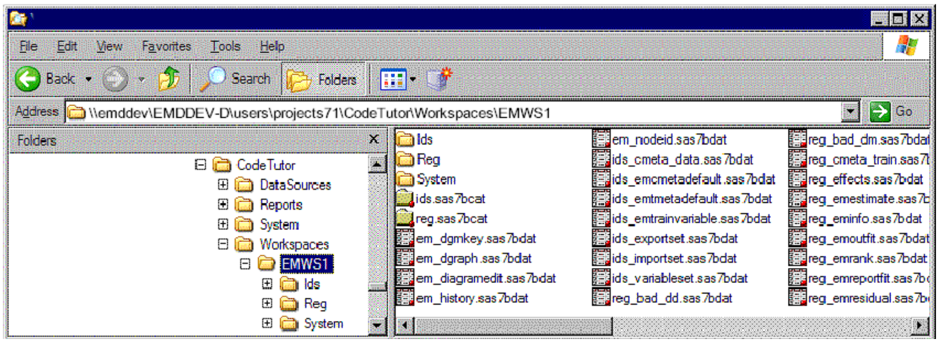
The Regression node in this example uses default property settings.

Example 2 Output Location:

You can find the output from Example 2 by reviewing the SAS log. Near the end of the log, look for lines that resemble these:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS1
```

These are the network mappings for the project path, the workspace path, and the workspace name. Your mapped paths will be different. After the batch processing code runs, you should see the new workspace folder, EMWS1, and the accompanying files and subfolders for the batch processing code example.



Example 2 Batch Processing Code:

After creating the CodeTutor project with the HMEQ data source on your SAS server, close SAS Enterprise Miner 12.3 and open a batch SAS session. You submit the code either as a batch job or through the SAS Program Editor window. Remember to change your EM_PROJECT path in the example code to your own network path before copying and pasting your batch processing code into the Program Editor window.

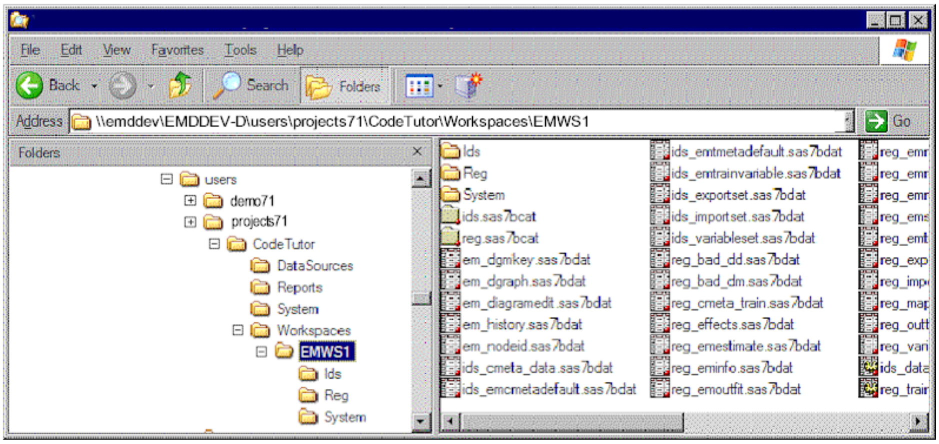
```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 2***     ;
* Use %let statements to hold values for project location     ;
* and project name, the name and location of the summary data ;
* set, specify single thread execution, and use the Replace   ;
* edit mode.                                                  ;
*------------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTutor;
%let EM_WSNAME = ;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = SINGLE;
%let EM_EDITMODE = R;
%let EM_DEBUG = ;
%let EM_ACTION = Run;

*------------------------------------------------------------*;
* Create Workspace data set and populate the property values  ;
* using the variables created by the %let statements above    ;
*------------------------------------------------------------*;
data workspace;
length property $64 value $100;
property= 'PROJECTLOCATION'; value= "<&EM_PROJECT>";     output;
property= 'PROJECTNAME';     value= "<&EM_PROJECTNAME>"; output;
property= 'WORKSPACENAME';   value= "<&EM_WSNAME>";      output;
property= 'SUMMARYDATASET';  value= "<&EM_SUMMARY>";     output;
property= 'NUMTASKS';        value= "<&EM_NUMTASKS>";    output;
property= 'EDITMODE';        value= "<&EM_EDITMODE>";    output;
property= 'DEBUG';           value= " ";               output;
run;

*------------------------------------------------------------*;
* Create Nodes data set for Data Source and Regression nodes. ;
* Create text labels for the GUI which read "Regression" for  ;
* the Regression node, and "HomeEquity" for the Data Source   ;
* node.                                                       ;
*------------------------------------------------------------*;

data nodes;
length id $12 component $32 description $64;

id= "Ids"; component="DataSource"; description= "HomeEquity"; output;
id= "Reg"; component="Regression"; description= "Regression"; output;

run;

*------------------------------------------------------------*;
* Create Node Properties data set for the Data Source node    ;
* Specify the HMEQ data set from the SAMPSIO library as the   ;
* data source. The Regression node uses all default property  ;
* settings, so no declarations are required for the Regression;
* node.                                                       ;
```

```
*-------------------------------------------------------------*;

data nodeprops;
length id $12 property $32 value $64;

id= "Ids"; property="DataSource"; value= "HOMEEQUITY"; output;

run;

*-------------------------------------------------------------*;
* Create Connections data set                                 ;
*-------------------------------------------------------------*;

data connect;
length from to $12;
input from to;
cards;

Ids Reg
;

run;



*-------------------------------------------------------------*;
* Create Actions that run the process flow or create reports, ;
* depending on the contents of the EM_ACTION variable set in  ;
* the %let statements at the top of this example. The flow is ;
* run from the Regression node.                               ;
*-------------------------------------------------------------*;

%macro emaction;
%let actionstring = %upcase(<&EM_ACTION>);
%if %index(<&actionstring>, RUN) or
    %index(<&actionstring>, REPORT) %then %do;

data actions;
length id $12 action $40;
id = "Reg";

%if %index(<&actionstring>, RUN) %then %do;
    action='run'; output;
    %end;
%if %index(<&actionstring>, REPORT) %then %do;
    action='report'; output;
    %end;
run;

%end;
%mend;
%emaction;

*-------------------------------------------------------------*;
* Execute the Actions                                         ;
* Use the %EM5BATCH macro and call the component data sets    ;
* that were created from the DATA step code in this example   ;
*-------------------------------------------------------------*;

  %EM5BATCH(execute,
      workspace=workspace,
      nodes=nodes,
      connect=connect,
      nodeprops=nodeprops,
      action=actions);
run;
```

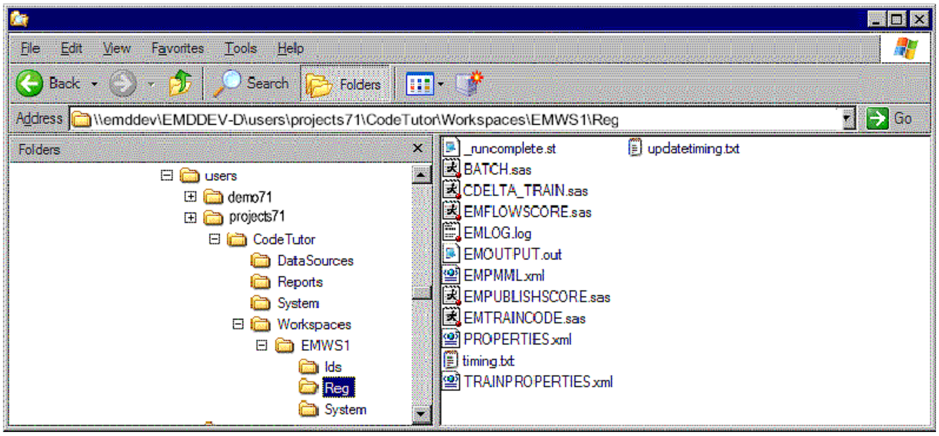
After the batch processing code runs successfully, examine the SAS log for errors. You can also use the SAS log to locate the project path, where you can view your workspace and output files.

View Example 2 Results Using a File Utility

When you examine the CodeTutor project folder after running the Example 2 batch processing code, notice that the DataSources, Reports, and System folders are not changed. However, the Workspaces folder, which was empty before the batch processing code ran, has changed. The Workspaces folder now contains a subfolder titled EMWS1. The EMWS1 folder contains subfolders for the data source (Ids), the Regression node (Reg), and a System folder.



The Ids and Reg folders contain node log and output files that are created each time a process flow diagram runs. The log and output files contain information such as property settings, training and scoring code, and PMML information.



View Example 2 Results Using the SAS Enterprise Miner GUI

Example 2 successfully demonstrates how batch processing code can use existing GUI projects and existing data sources, while creating new workspaces and process flows. If you start the SAS Enterprise Miner GUI and open the CodeTutor project, you can see the diagram workspace EMWS1 and the corresponding process flow diagram that you created using the SAS Enterprise Miner batch processing interface.

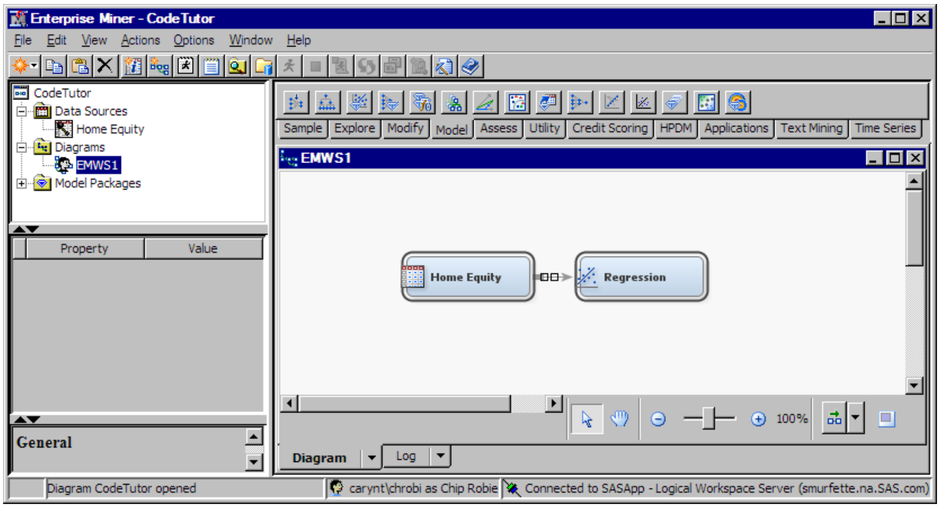


Diagram CodeTutor opened

carynt\chrobi as Chip Robie

Connected to SASApp - Logical Workspace Server (smurfette.na.SAS.com)

Example 3: Modify an Existing Project and Workspace, Compare Models, and Run Reports

Examples 2 through 6 are batch processing code examples that build on the previous example and increase progressively in complexity. Example 3 is the second example in the series. Example 3 modifies the empty workspace that you created in Example 2 and adds two nodes to the process flow diagram: a Decision Tree node and a Model Comparison node.

The workspace that you created in Example 2 is called EMWS1. In order to modify the EMWS1 workspace using batch processing code, you must set the diagram EDITMODE to M, for Merge. The Merge setting enables batch processing code to add nodes and connections to existing process flow diagrams. The batch processing code in Example 3 uses a %LET statement at the top of the batch processing code to initialize the EDITMODE variable to Merge. The Workspace data set in the batch processing code calls the EDITMODE variable. The Create Nodes data set adds the Decision Tree node parallel to the Regression node and adds a Model Comparison node to assess the output from the two different modeling nodes.

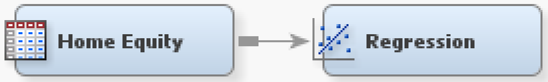
Because the existing process flow diagram has changed, it must be rerun. The FORCERUN property in the workspace data set is now set to Y, which forces the entire process flow diagram (including the new nodes) to be rerun.

The project workspace specifies that the execution will occur in a single thread.

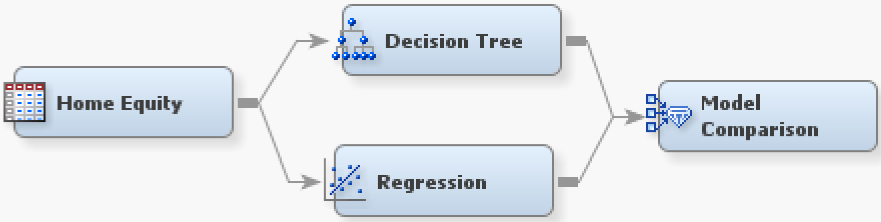
Note that in the Create Nodes data set of the Example 3 batch processing code, the DecisionTree and Model Comparison ID components use placeholders '$1' and '$2' instead of assigning text. When you modify an existing workspace, it is a good idea to use placeholder IDs such as $1 and $2 in order to prevent accidental modification of pre-existing properties.

Example 3 Diagram Overview:

The diagrams below show the SAS Enterprise Miner process flow diagrams that you created in the batch processing environment. If you used the SAS Enterprise Miner GUI to open the diagrams that you created in batch processing code Examples 2 and 3 you would see:



Example 2 process flow diagram



Example 3 process flow diagram

It is not necessary to create either of these process flow diagrams using the GUI in this example. The batch processing code will create them. The diagrams above are shown to help batch processing code users visualize the example.

Example 3 Node Properties Overview:

The Input Data node and Regression nodes retain their configurations from Example 2. The added Decision Tree and Model Comparison nodes use the default configurations and do not require any properties to be set for node execution.

Example 3 Output Location:

You can find the output from Example 3 by reviewing the SAS log. Near the end of the log, look for lines that resemble these:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS1
```

Example 3 Batch Processing Code:

Note: Because Example 3 was created by modifying Example 2, additions or changes to the code are shown in a lighter shade. You should submit and run Example 2 (with your own project mapping and data source) before you run the batch processing code for Example 3.

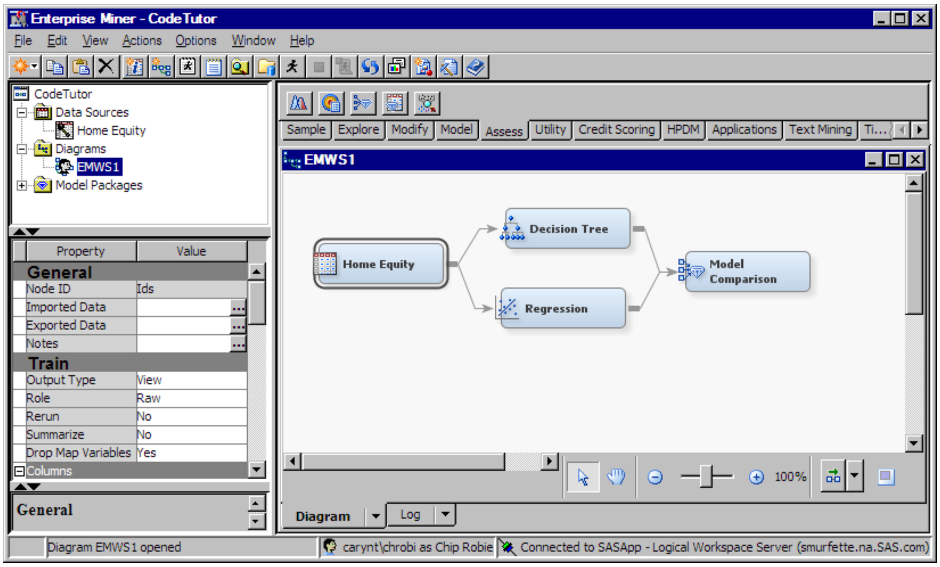
```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 3***     ;
* Use %let statements to hold values for project location     ;
* and project name, the workspace name, the name and location ;
* of the summary data set, specify the Merge batch edit mode, ;
* and execute the process flow diagram in a single thread     ;
*------------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTutor;
%let EM_WSNAME = EMWS1;
```

```
%let EM_EDITMODE = M;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = SINGLE;
%let EM_DEBUG = ;
%let EM_ACTION = RUN;

*------------------------------------------------------------*;
* Create Workspace data set and populate the property values  ;
* using the variables created by the %let statements above.   ;
* The NUMTASKS data step picks up the value of SINGLE         ;
* and the FORCERUN data step ensures that all nodes will be   ;
* run when the batch process flow diagram is run.             ;
*------------------------------------------------------------*;

data workspace;
length property $64 value $100;

property= 'PROJECTLOCATION'; value= "<&EM_PROJECT>";     output;
property= 'PROJECTNAME';     value= "<&EM_PROJECTNAME>"; output;
property= 'WORKSPACENAME';   value= "<&EM_WSNAME>";      output;
property= 'EDITMODE';        value= "<&EM_EDITMODE>";    output;
property= 'SUMMARYDATASET';  value= "<&EM_SUMMARY>";     output;
property= 'NUMTASKS';        value= "<&EM_NUMTASKS>";    output;
property= 'DEBUG';           value= " ";               output;
property='FORCERUN';         value="Y";                output;

run;
*------------------------------------------------------------*;
* Create Nodes Data Set                                       ;
* The Decision Tree and Model Compare nodes are added to the  ;
* existing flow by adding to the nodes data. Note that the    ;
* existing Data Source and Regression nodes from Example 2    ;
* are already in the nodes data and are not repeated in this  ;
* iteration of the Create Nodes data set.                     ;
*------------------------------------------------------------*;

data nodes;
length id $12 component $32 description $64;
id= "$1"; component="DecisionTree"; description= "Decision Tree"; output;
id= "$2"; component="ModelCompare"; description= "Model Compare"; output;
run;
*------------------------------------------------------------*;
* Create Connections Data Set                                 ;
* The Decision Tree and Model Compare nodes are added to the  ;
* existing flow by adding to the connect data. Note that the  ;
* existing connections from Example 2 are not restated. Only  ;
* the new connections in the flow diagram are specified. Also ;
* note the use of wildcard IDs ($1 and $2) instead of text.   ;
*------------------------------------------------------------*;

data connect;
length from to $12;
input from to;
cards;

Ids $1
```

```
Reg $2
$1 $2
;
run;
*-------------------------------------------------------------*;
* Create Actions that run the process flow or create reports, ;
* depending on the contents of the EM_ACTION variable set in  ;
* the %let statements at the top of this example. The flow is ;
* run and the reports are generated from the Model Comparison ;
* node ($2) instead of the Regression node.                   ;
*-------------------------------------------------------------*;

%macro emaction;
%let actionstring = %upcase(<&EM_ACTION>);
%if %index(<&actionstring>, RUN)
or %index(<&actionstring>, REPORT) %then %do;
data actions;
length id $12 action $40;
id="$2";

%if %index(<&actionstring>, RUN) %then %do;
action='run';
output;
%end;

%if %index(<&actionstring>, REPORT) %then %do;
action='report';
output;
%end;
run;

%end;
%mend;
%emaction;
*-------------------------------------------------------------*;
* Execute the Actions                                         ;
* Use the %EM5BATCH macro and call the component data sets    ;
* that were created from the DATA step code in the accumulated;
* examples.                                                   ;
*-------------------------------------------------------------*;

%EM5BATCH(execute,
      workspace=workspace,
      nodes=nodes,
      connect=connect,
      datasources=datasources,
      action=actions);

run;
```

View Example 3 Results Using the SAS Enterprise Miner GUI

You can view the changes Example 3 made to the CodeTutor project in the SAS Enterprise Miner GUI after your batch processing code runs. Open the CodeTutor project and you can see the nodes that were added in the Example 3 batch processing code.


Enterprise Miner - CodeTutor
File Edit View Actions Options Window Help
CodeTutor
Data Sources
Home Equity
Diagrams
EMWS1
Model Packages
Sample Explore Modify Model Assess Utility Credit Scoring HPDM Applications Text Mining Ti...
EMWS1
Home Equity
Decision Tree
Regression
Model Comparison
100%
Diagram
Log
General

Property	Value
General	
Node ID	Ids
Imported Data	...
Exported Data	...
Notes	...
Train	
Output Type	View
Role	Raw
Rerun	No
Summarize	No
Drop Map Variables	Yes
Columns	

Diagram EMWS1 opened | carynt\chrobi as Chip Robie | Connected to SASApp - Logical Workspace Server (smurfette.na.SAS.com)

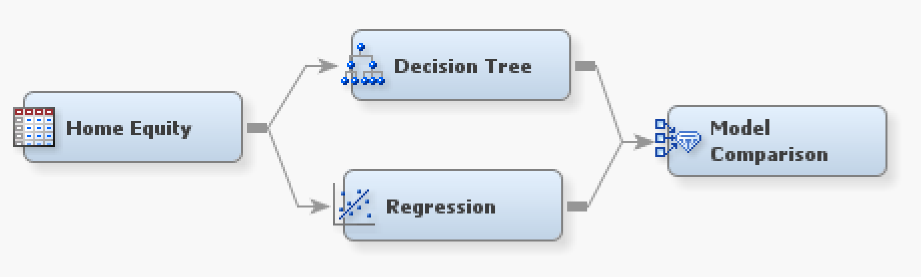
Example 4: Modify the Example 3 Workspace by Adding a New Data Source and New Metadata

In Example 4, you further modify the project workspace that you created in Example 2 and modified in Example 3. Example 4 uses the %EMDS macro to add a new data source named DMEXA1TABLE to the project workspace. The EMDS filename and %INCLUDE statements ensure that the EMDS catalog and formats are found. The EMDS libref points to the DataSources folder of the project that you are working in. Then the batch processing code uses a DATA step to modify the metadata for DMEXA1TABLE.

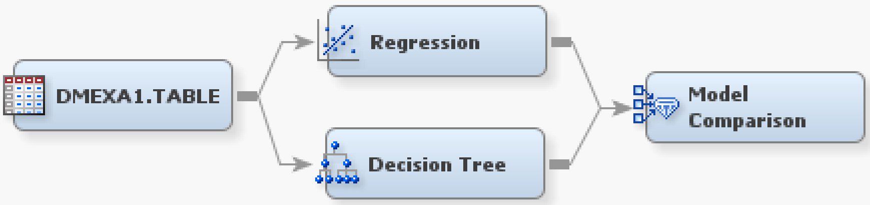
The Datasource property of the Input Data node is modified to change the data source from HOMEEQUITY (the data source that was used in Examples 2 and 3) to DMEXA1TABLE, the newly created data source. The entire batch process flow is run and a report is generated.

Example 4 Diagram Overview:

The diagrams below show the GUI equivalents of the SAS Enterprise Miner 1 example process flow diagrams that the batch processing code creates in Examples 3 and 4. It is not necessary to create the process flow diagrams in the GUI. The example batch processing code creates them. The diagrams are shown to help batch processing code users visualize the changes from one example to the next.



Example 3 process flow diagram



Example 4 process flow diagram

Example 4 Node Properties Overview:

The nodes in Example 4 all use default property settings.

Example 4 Output Location:

You can find the output from Example 4 by reviewing the SAS log from your batch processing code run. Near the end of the log, look for lines that resemble these:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS1
```

Your output path will be different.

Example 4 Batch Processing Code:

Because Example 4 was created by modifying Example 3, additions or changes to the code are shown in a lighter shade. You should submit and run Example 3 (with your own project mapping) before you update and submit the batch processing code for Example 4.

As in previous examples, you will need to substitute your own values for the library path to EMDS, the userID in the %EMDS macro, and the EM_PROJECT path mapping.

Make your changes to the batch processing code below and submit the batch processing code to SAS.

```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 4***     ;
* Locate the formatting catalogs for the SAMPSIO library.     ;
* The 'emdsmcro' fileref permits the emutil.emds catalog to   ;
* be referenced as an external file and enables the %EMDS     ;
* macro, which is not included in base SAS. The EMDS libref   ;
* points to the project's DataSources subdirectory.           ;
*------------------------------------------------------------*;

options fmtsearch=(work sampsio.emfmt);
filename emdsmcro
         catalog 'sashelp.emutil.emds.source' ;
%include emdsmcro ;
libname EMDS '\\emddev\EMDDEV-D\users\projects71\CodeTutor\DataSources' ;


*------------------------------------------------------------*;
* The %EMDS macro uses the DMEXA1 data to create metadata for ;
* the DMEXA1TABLE data source. The macro names 'PURCHASE' as  ;
* the target variable. The DATA step modifies columns metadata;
* manually. 'ACCOUNT' becomes an ID variable and 'STATECOD'   ;
* becomes a rejected variable.                                ;
```

```
*------------------------------------------------------------*;

  %EMDS(data=SAMPSIO.DMEXA1,
      rootLibrary=EMDS,
      target=PURCHASE,
      name=DMEXA1TABLE,
      userid=chrobi,
      tablerole=TRAIN,
      adviseMode=ADVANCED
      );
data <&emds_cm>
   set <&emds_cm>
      if NAME="ACCOUNT" then ROLE="ID";
   else
      if NAME="STATECOD" then ROLE="REJECTED";

run;

*------------------------------------------------------------*;
* Use %let statements to hold values for project location     ;
* and project name, the workspace name, the name and location ;
* of the summary data set, specify Merge batch edit mode, and ;
* single threaded batch execution. Add 'REPORT' to the        ;
* EM_ACTION actions place holder.                             ;
*------------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTutor;
%let EM_WSNAME = EMWS1;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = SINGLE;
%let EM_EDITMODE = M;
%let EM_DEBUG = ;
%let EM_ACTION = RUN REPORT;


*------------------------------------------------------------*;
* Create Workspace data set and populate the property values  ;
* using the variables created by the %let statements above.   ;
*------------------------------------------------------------*;
data workspace;
length property $64 value $100;

property= 'PROJECTLOCATION'; value= "<&EM_PROJECT>";     output;
property= 'PROJECTNAME';     value= "<&EM_PROJECTNAME>"; output;
property= 'WORKSPACENAME';   value= "<&EM_WSNAME>";      output;
property= 'EDITMODE';        value= "<&EM_EDITMODE>";    output;
property= 'SUMMARYDATASET';  value= "<&EM_SUMMARY>";     output;
property= 'NUMTASKS';        value= "<&EM_NUMTASKS>";    output;
property= 'DEBUG';           value= " ";               output;

run;

*------------------------------------------------------------*;
* Create node properties data set                             ;
*------------------------------------------------------------*;
```

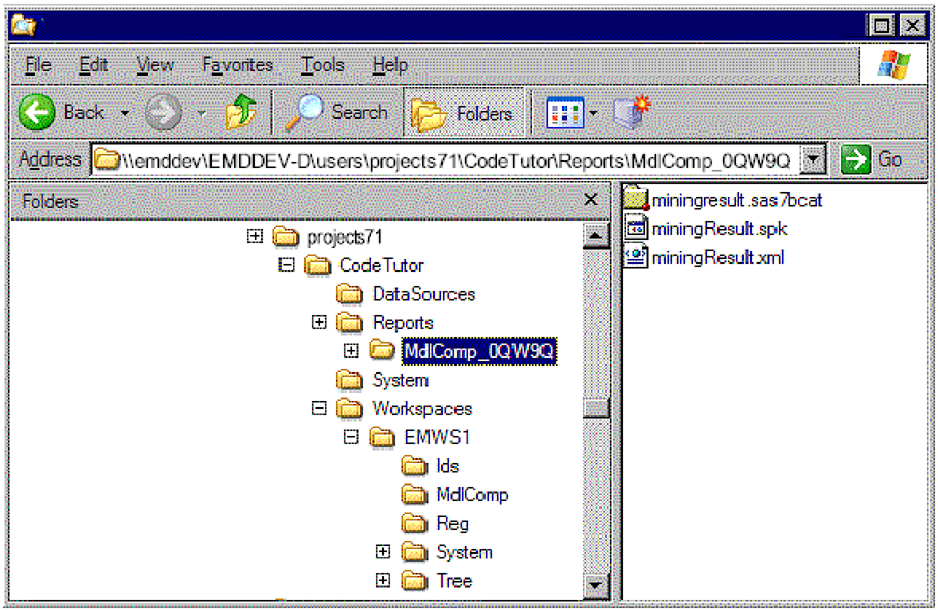
```
data nodeprops;
length id $12 property $32 value $64;

id= "Ids"; property="DataSource"; value= "dmexa1table"; output;
run;

*------------------------------------------------------------*;
* Create Actions to Run data set with the new nodes.          ;
*------------------------------------------------------------*;

%macro emaction;
%let actionstring = %upcase(<&EM_ACTION>);

%if %index(<&actionstring>, RUN)
or %index(<&actionstring>, REPORT) %then %do;

data actions;
length id $12 action $40;
id="MdlComp";

%if %index(<&actionstring>, RUN) %then %do;
   action='run';
   output;
%end;

%if %index(<&actionstring>, REPORT) %then %do;
   action='report';
   output;
%end;

run;

%end;
%mend;
%emaction;

*------------------------------------------------------------*;
* Execute the Actions using the %EM5BATCH macro               ;
*------------------------------------------------------------*;
%EM5BATCH(execute,
      workspace=workspace,
      nodeprops=nodeprops,
      action=actions
nodes=,
      connect=,
      datasources=);
run;
```

Note that the arguments for the nodes, connect, and datasources definitions are blank. This prevents the %EM5BATCH macro from automatically using the nodes data set, connections data set, and datasources data set that was most recently defined in Example 3. More information is available about the Batch Processing Code Data Set Caching section of this document.

After the batch processing code runs successfully, examine the SAS log for errors. You can also use the SAS log to locate the project path, where you can view the workspace and output files.

View Example 4 Results Using a File Utility

You can view the Example 4 output files using a file utility and the path that is found in the SAS log. Below you can see the Reports subfolder of the CodeTutor project with the newly created Model Comparison reports. Inside the Reports subfolder are the mining results in SAS file format, XML file format, and SPK file format. Opening the DataSources subfolder will also reveal the new DMEXA1TABLE data source as well.

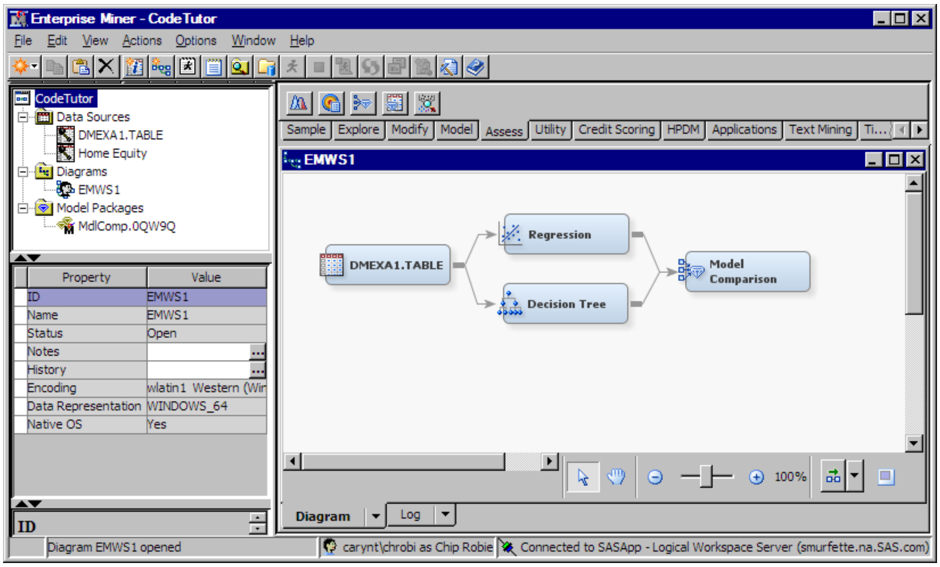


Address: \\emddev\EMDDEV-D\users\projects71\CodeTutor\Reports\MdlComp_0QW9Q

View Example 4 Results Using the SAS Enterprise Miner GUI

You can also view the changes that Example 4 made to the CodeTutor project in the SAS Enterprise Minerz GUI after your batch processing code runs. Open the CodeTutor project and you can see the various changes that were made by the Example 4 batch processing code. The SAS Enterprise Miner GUI below shows the following information:

- The Project Navigator shows the newly created DMEXA1TABLE data source in the project's Data Sources folder.
- The Project Navigator shows the newly created Model Compare report in the project's Model Packages folder.
- The Diagram Workspace shows the changed process flow diagram.
- If you select the DMEXA1TABLE Data Source in your process flow diagram or in the DMEXA1TABLE in the Data Sources folder, you see that the data set and its metadata were properly created.


Enterprise Miner - CodeTutor
File Edit View Actions Options Window Help
CodeTutor
Data Sources
DMEXA1.TABLE
Home Equity
Diagrams
EMWS1
Model Packages
MdlComp.0QW9Q
Sample Explore Modify Model Assess Utility Credit Scoring HPDM Applications Text Mining Ti...
EMWS1
DMEXA1.TABLE
Regression
Decision Tree
Model Comparison
100%
Diagram
Log
ID

Property	Value
ID	EMWS1
Name	EMWS1
Status	Open
Notes	
History	
Encoding	wlatin1 Western (Wir
Data Representation	WINDOWS_64
Native OS	Yes

Diagram EMWS1 opened | carynt\chrobi as Chip Robie | Connected to SASApp - Logical Workspace Server (smurfette.na.SAS.com)

View Example 4 Results in the SAS Enterprise Miner Results Package

You can also open and browse the Model Compare report that is found in the Model Packages folder of the CodeTutor Project Navigator by double-clicking the report under the Model Packages folder in the Project Navigator.

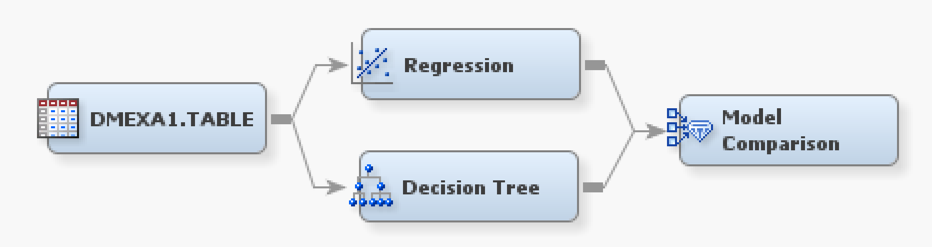
You can use the SAS Enterprise Miner GUI to modify the process flow diagram that you first created in batch processing code. You can open up the process flow diagram in the GUI, use the GUI to make changes in the diagram, then save the GUI changes in more detailed batch processing code.

Example 5: Modify the Example 4 Workspace with a Target Profile Profit Matrix and Metadata

Example 5 further modifies the workspace that was changed in Example 4. In Example 5, you create a target profile decision matrix for the Input Data. The target profile decision matrix is a profit matrix that has two or more decisions. Example 5 uses the %EMTP macro to create the profit matrix. The profit matrix weights are specified in the DECDATA data set. Example 5 uses a DATA step to modify the DECDATA data set. The EM_DECMETA_PURCHASE and the EM_DECDATA_PURCHASE properties are used to specify the decision matrix and metadata information. The batch processing code then runs the entire process flow and creates a report.

Example 5 Diagram Overview:

The diagram below shows the GUI equivalents of the SAS Enterprise Miner process flow diagrams that the batch processing code creates in Example 5. It is not necessary to create the process flow diagram in the GUI. The diagrams are shown to help batch processing code users visualize the changes from one example to the next.



Example 5 process flow diagram

Example 5 Node Property Overview:

The nodes in Example 5 all use default node property settings.

Example 5 Output Location:

As in previous examples, you can find the output files from Example 5 by reviewing the SAS log. Near the end of the log, look for lines that resemble these:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS1
```

Your output path will be different.

Example 5 Batch Processing Code:

Because Example 5 was created by modifying Example 4, additions or changes to the code are shown in a lighter shade of color. You should submit and run Example 4 (with your own project mapping) before you update and submit the batch processing code for Example 5.

As in previous examples, you will need to substitute your own values for the library path to EMDS, the userID in the %EMDS macro, and the EM_PROJECT path mapping.

Make your changes to the batch processing code below and submit the batch processing code to SAS.

```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 5***     ;
* Locate the formatting catalogs for the SAMPSIO library.     ;
* The 'emtpmcro' fileref permits the emutil.emtp catalog to   ;
* be referenced as an external file and enables the %EMTP     ;
* macro, which is not included in base SAS. The EMDS libref   ;
* points to the project's Datasources subdirectory.           ;
*------------------------------------------------------------*;
options fmtsearch=(work sampsio.emfmt);
filename emtpmcro
         catalog 'sashelp.emutil.emtp.source' ;
%include emtpmcro ;
libname EMDS  '\\emddev\EMDDEV-D\users\projects71\CodeTutor\DataSources' ;


*------------------------------------------------------------*;
* Use the %EMTP macro to create a target profile decision     ;
* matrix which is a profit matrix with two decisions. The     ;
* target variable is 'PURCHASE'. The profit matrix weights are;
* specified using 'DECDATA' in the DATA step.                 ;
*------------------------------------------------------------*;
%EMTP(data = SAMPSIO.DMEXA1,
      target = PURCHASE,
      columnsmeta = EMDS.DMEXA1TABLE_CM,
      decdata = WORK.DECDATA,
      decmeta = WORK.DECMETA,
      dectype=PROFIT,
      numdec=2);
data WORK.DECDATA;
   set WORK.DECDATA;

   if PURCHASE='YES' then do;
      DECISION1 = 1;
      DECISION2 = -1;
   end;
   else do;
      DECISION1 = 0;
      DECISION2 = 1;
   end;
run;

*------------------------------------------------------------*;
* Use %let statements to hold values for project location     ;
* and project name, the workspace name, the name and location ;
* of the summary data set, use SINGLE to specify a single     ;
* thread for process flow diagram work processing, and specify;
* the Merge batch edit mode.                                  ;
*------------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTutor;
%let EM_WSNAME = EMWS1;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = SINGLE;
%let EM_EDITMODE = M;
%let EM_DEBUG = ;
%let EM_ACTION =RUN REPORT;

*------------------------------------------------------------*;
* Create Workspace data set and populate the property values  ;
* using the variables created by the %let statements above.   ;
*------------------------------------------------------------*;
data workspace;
length property $64 value $100;

property= 'PROJECTLOCATION'; value= "<&EM_PROJECT>";     output;
property= 'PROJECTNAME';     value= "<&EM_PROJECTNAME>"; output;
property= 'WORKSPACENAME';   value= "<&EM_WSNAME>";      output;
property= 'EDITMODE';        value= "<&EM_EDITMODE>";    output;
property= 'SUMMARYDATASET';  value= "<&EM_SUMMARY>";     output;
property= 'NUMTASKS';        value= "<&EM_NUMTASKS>";    output;
property= 'DEBUG';           value= "";                            output;

run;

*------------------------------------------------------------*;
* Create Node Properties data set and assign the Target       ;
* Profile data sets to the Ids node. The EM_DECMETA_PURCHASE  ;
* property specifies the metadata information and the         ;
* EM_DECDATA_PURCHASE property specifies the decision matrix  ;
* information.                                                ;
*-------------------------------------------------------------*;

data nodeprops;
length id $12 property $32 value $64;

id= "Ids"; property="EM_DECMETA_PURCHASE"; value= "WORK.DECMETA"; output;
id= "Ids"; property="EM_DECDATA_PURCHASE"; value= "WORK.DECDATA"; output;

run;

*-------------------------------------------------------------*;
* Create the Actions to Run data set                          ;
*-------------------------------------------------------------*;
%macro emaction;
%let actionstring = %upcase(<&EM_ACTION>);
%if %index(<&actionstring>, RUN) or %index(<&actionstring>, REPORT) %then %do;
data actions;
length id $12 action $40;
id="MdlComp";

%if %index(<&actionstring>, RUN) %then %do;
   action='run';
   output;
%end;
%if %index(<&actionstring>, REPORT) %then %do;
   action='report';
   output;
%end;
run;
%end;
%mend;
%emaction;

*-------------------------------------------------------------*;
* Execute the Actions using the %EM5BATCH macro               ;
*-------------------------------------------------------------*;

  %em5batch(execute,
    workspace=workspace,
    nodeprops=nodeprops,
    action=actions);
run;
```

After the batch processing code runs successfully, examine the SAS log for errors. You can also use the SAS log to locate the path to the project directory, where you can view the workspace and output files.

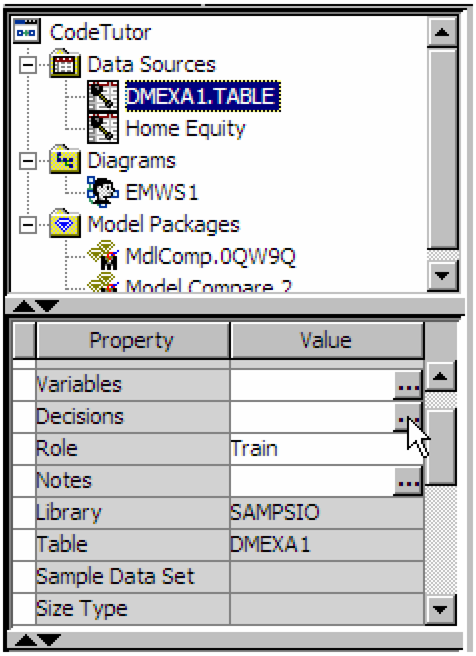
View Example 5 Results Using the SAS Enterprise Miner GUI

After your batch processing code runs, you can open the SAS Enterprise Miner GUI to view the target profile matrix and the additional report that Example 5 created in the CodeTutor project. Open the CodeTutor project and you can see the various changes that were made by the Example 5 batch processing code.

- The Project Navigator shows the additional Model Compare report that was created during Example 5 in the project's Model Packages folder. You can open and browse the new report.
- Select the DMEXA1TABLE data source in your process flow diagram or in the Data Sources folder of the Project Navigator to populate the Properties Panel with the DMEXA1TABLE information. You can use the Properties Panel to open the Decision Processing table in the GUI and see the decision matrix that was created.

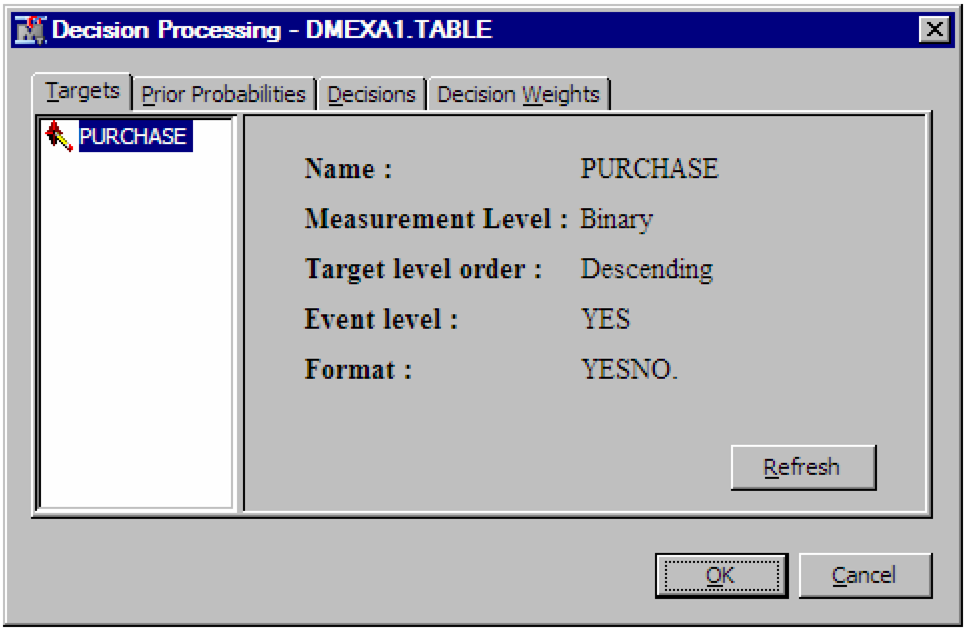
View the Example 5 Decision Matrix for DMEXA1TABLE:

To open the Decision Processing window, select the DMEXA1TABLE data source, select the Decisions property of the Settings group, and right-click the icon in the right column as shown below:



Property	Value
Variables	...
Decisions	...
Role	Train
Notes	...
Library	SAMPSIO
Table	DMEXA1
Sample Data Set	
Size Type	

This opens the Decision Processing -- DMEXA1TABLE window. The Targets tab shows you information about the target variable PURCHASE:



Name : PURCHASE

Measurement Level : Binary

Target level order : Descending

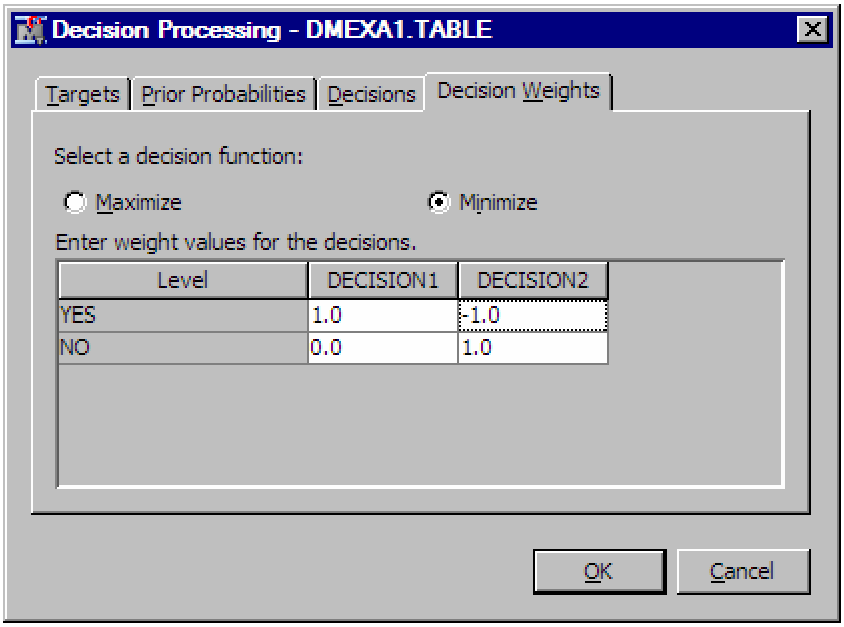
Event level : YES

Format : YESNO.

You can click the Prior Probabilities tab if you want to see the event probabilities that were summarized from target variable levels and counts.

The Decisions tab indicates whether decisions are used, and also provides an interface to specify labels for decision variables, or to specify cost variable or constant metadata.

The Decision Weights tab displays the decision matrix and weights that were created in Example 5.



Select a decision function:

○ Maximize ● Minimize

Enter weight values for the decisions.

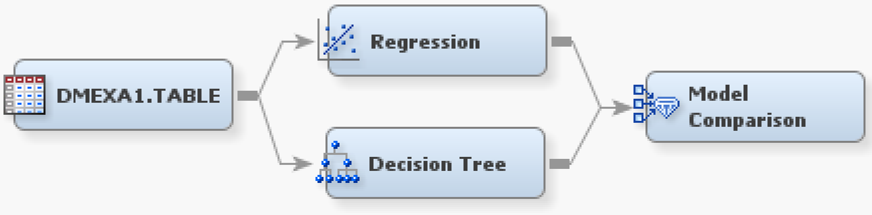
Level	DECISION1	DECISION2
YES	1.0	-1.0
NO	0.0	1.0

Example 6: Create a Target Profile for a Previously Defined Data Source

Example 6 is a continued modification of Example 5. This example shows how to create a target profile and assign it to a previously defined data source. The DMEXA1TABLE_TP data set is a registration data set for the profiles of different targets that are associated with that data source. In Example 6, the EMTP data set creates the profit matrix weight data set for profile 0 (EMDS.DMEXA1TABLE_D0) and the metadata data set for profile 0 (EMDS.DMEXA1TABLE_M0.

Example 6 Diagram Overview:

The diagram below shows the GUI equivalents of the SAS Enterprise Miner process flow diagrams that the batch processing code creates in Examples 6. It is not necessary to create the process flow diagram in the GUI. The diagrams are shown to help batch processing code users visualize the changes from one example to the next.



Example 6 process flow diagram

The representative GUI Process Flow Diagram for Example 6 is the same as Example 5.

Example 6 Node Property Overview:

The nodes in Example 6 all use default node properties.

Example 6 Output Location:

The output for Example 6 is in the same project location as the output files for Examples 2 through 5.

Example 6 Batch Processing Code:

Because Example 6 was created by modifying Example 5, additions or changes to the code are shown in a lighter shade of color. You should submit and run Example 5 (with your own project mapping) before you update and submit the batch processing code for Example 6.

As in previous examples, you will need to substitute your own values for the library path to EMDS and for your EM_PROJECT path mapping.

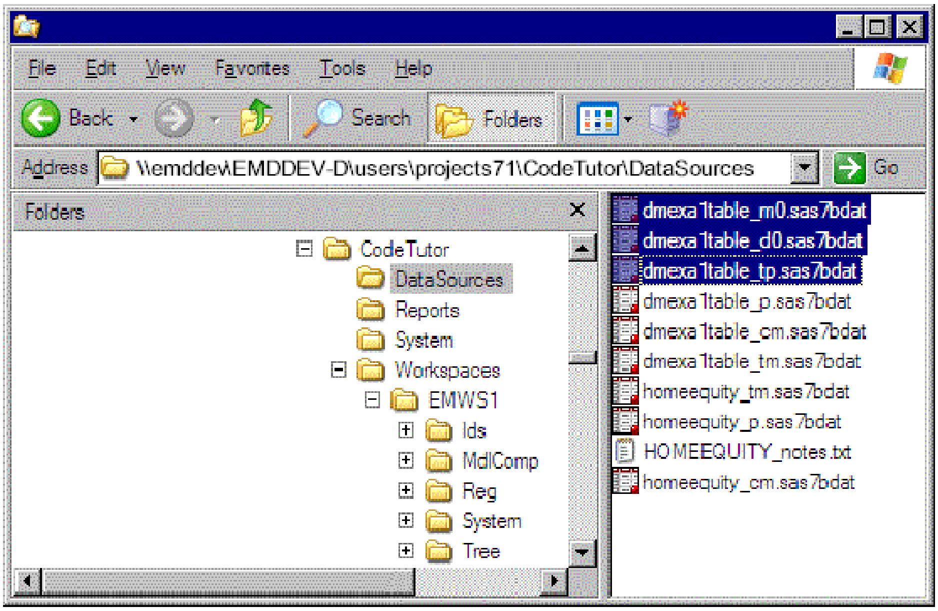
Make your changes to the batch processing code below and submit the batch processing code to SAS:

```
*-------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 6***     ;
* Locate the formatting catalogs for the SAMPSIO library.     ;
* The 'emtpmcro' fileref permits the emutil.emtp catalog to   ;
* be referenced as an external file and enables the %EMTP     ;
* macro, which is not included in base SAS. The EMDS libref   ;
* points to the project's Datasources subdirectory.           ;
*-------------------------------------------------------------*;
options fmtsearch=(work sampsio.emfmt);
filename emtpmcro catalog 'sashelp.emutil.emtp.source' ;
%include emtpmcro ;
libname EMDS  '\\emddev\EMDDEV-D\users\projects71\CodeTutor\DataSources' ;


*-------------------------------------------------------------*;
* Use the DATA step to build the target profile registration  ;
* data set called DMEXAITABLE_TP.                             ;
*------------------------------------------------------------*;

data EMDS.DMEXA1TABLE_TP;
length ProfileID 8;
length ProfileName $64;
length Target $64;
length Level $16;
length Use $8;
ProfileID = 0;
ProfileName='Profile 0';
Target='Purchase';
Use='Y';
output;

*------------------------------------------------------------*;
* Create Target Profile for the table SAMPSIO.DMEXA1 with a   ;
* target variable called 'PURCHASE'. Columnsmeta and decision ;
* matrix decision data and meta data are generated for the    ;
* profit matrix with two decisions. The DATA step adds values ;
* for the profit matrix.                                      ;
*------------------------------------------------------------*;

  %EMTP(data = SAMPSIO.DMEXA1,
      target = PURCHASE,
      columnsmeta = EMDS.DMEXA1TABLE_CM,
      decdata = EMDS.DMEXA1TABLE_D0,
      decmeta = EMDS.DMEXA1TABLE_M0,
      dectype = PROFIT,
      numdec = 2);

data EMDS.DMEXA1TABLE_D0;
   set EMDS.DMEXA1TABLE_D0;
   if PURCHASE='YES' then do;
      DECISION1 =1;
      DECISION2=-1;
   end;
   else do;
      DECISION1=0;
      DECISION2=1;
   end;

run;

*------------------------------------------------------------*;
* Use a DATA step to assign cost values for each variable in  ;
* the profit matrix                                           ;
*------------------------------------------------------------*;

  data EMDS.DMEXA1TABLE_M0;
   set EMDS.DMEXA1TABLE_M0;
   if VARIABLE = 'DECISION1' then COST='1.5';
   else
   if VARIABLE = 'DECISION2' then COST='0.5';

run;
```

After the batch processing code runs successfully, examine the SAS log for errors. You can also use the SAS log to locate the path to the project directory, where you can view the workspace and output files.

View Example 6 Results with a File Utility:

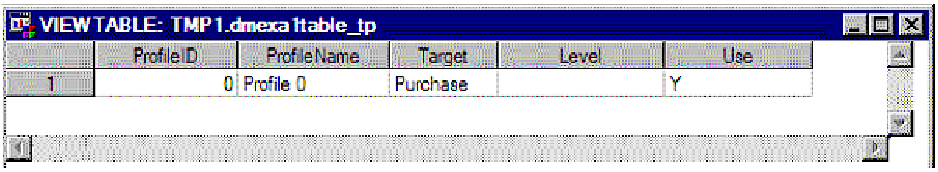
You can view the files created in Example 6 using a file utility and the project output path that is found in the SAS log. Below, you can see the newly created SAS binary data files for the DMEXA1TABLE_TP registration data set, the DMEXA1TABLE_D0 profit matrix weight data set, and the DMEXA1TABLE_M0 metadata data set for Profile 0:



Address: \\emddev\EMDDEV-D\users\projects71\CodeTutor\DataSources

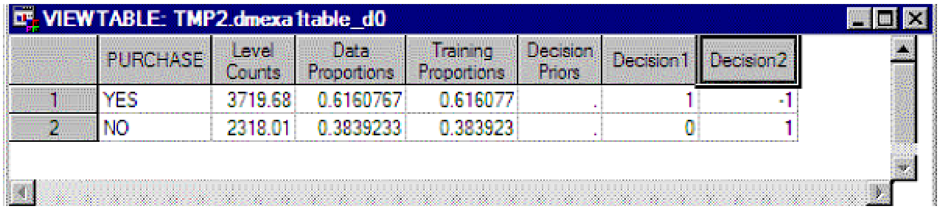
dmexa1table_m0.sas7bdat
dmexa1table_d0.sas7bdat
dmexa1table_tp.sas7bdat
dmexa1table_p.sas7bdat
dmexa1table_cm.sas7bdat
dmexa1table_tm.sas7bdat
homeequity_tm.sas7bdat
homeequity_p.sas7bdat
HOMEEQUITY_notes.txt
homeequity_cm.sas7bdat

Clicking the DMEXA1TABLE_TP.sas7bdat file opens the target profile registration data set, where Target Profile 0 is defined in a SAS table:


VIEWTABLE: TMP1.dmexa1table_tp

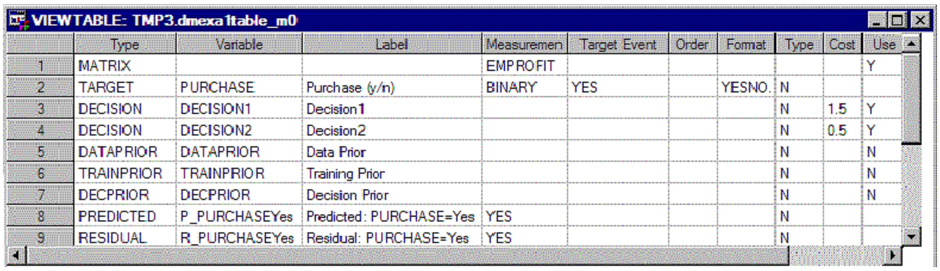
	ProfileID	ProfileName	Target	Level	Use
1	0	Profile 0	Purchase		Y

Clicking the DMEXA1TABLE_D0.sas7bdat file opens the profit matrix weight data set for Profile 0 in a SAS table:


VIEWTABLE: TMP2.dmexa1table_d0

	PURCHASE	Level Counts	Data Proportions	Training Proportions	Decision Priors	Decision1	Decision2
1	YES	3719.68	0.6160767	0.616077	.	1	-1
2	NO	2318.01	0.3839233	0.383923	.	0	1

Clicking the DMEXA1TABLE_M0 opens the metadata data set for Profile 0 in a SAS table, where you can see the costs that Example 6 associated with the profit matrix decisions.

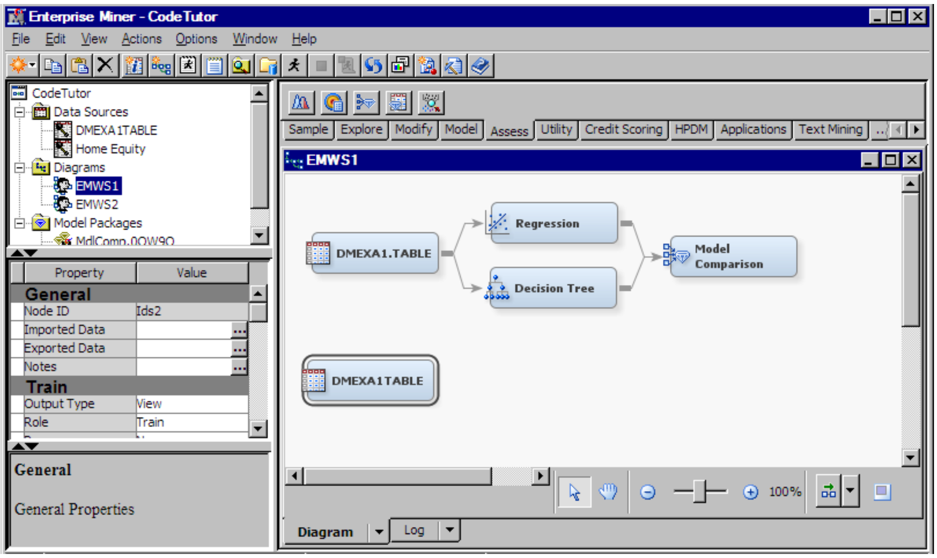


VIEWTABLE: TMP3.dmexa1table_m0

	Type	Variable	Label	Measuremen	Target Event	Order	Format	Type	Cost	Use
1	MATRIX			EMPROFIT						Y
2	TARGET	PURCHASE	Purchase (y/n)	BINARY	YES		YESNO.	N		
3	DECISION	DECISION1	Decision1					N	1.5	Y
4	DECISION	DECISION2	Decision2					N	0.5	Y
5	DATAPRIOR	DATAPRIOR	Data Prior					N		N
6	TRAINPRIOR	TRAINPRIOR	Training Prior					N		N
7	DECPRIOR	DECPRIOR	Decision Prior					N		N
8	PREDICTED	P_PURCHASEYes	Predicted: PURCHASE=Yes	YES				N		
9	RESIDUAL	R_PURCHASEYes	Residual: PURCHASE=Yes	YES				N		

View Example 6 Results in the SAS Enterprise Miner GUI:

You can use the SAS Enterprise Miner GUI to view the profit matrix that Example 6 created in the CodeTutor project. Open the EMWS1 Diagram in your CodeTutor project and drag a new copy of the DMEXA1TABLE data source from the DataSource folder of the Project Navigator to an empty section of the Diagram Workspace:

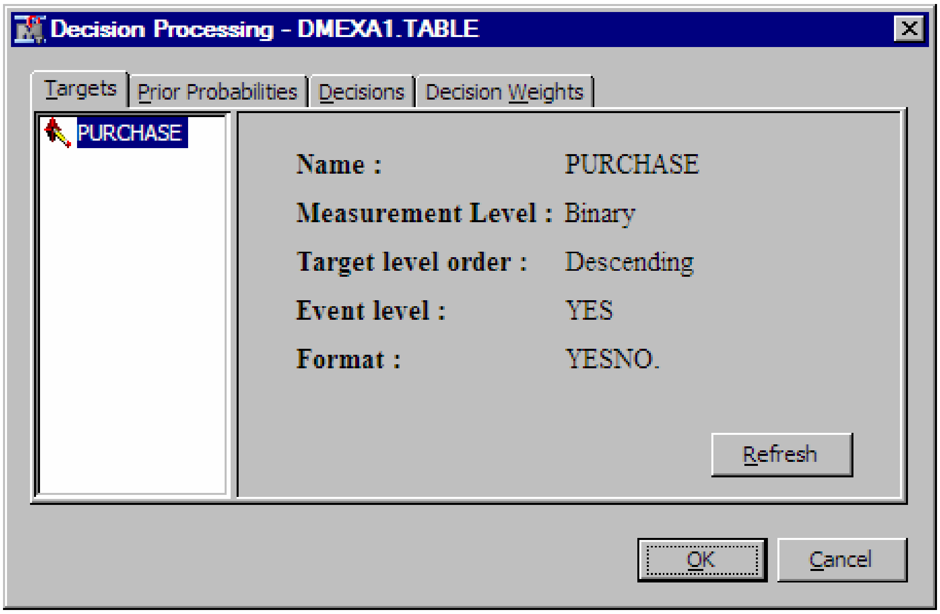


Property	Value
General	
Node ID	Ids2
Imported Data	...
Exported Data	...
Notes	...
Train	
Output Type	View
Role	Train

General

General Properties

When you select the DMEXA1TABLE data source, you can go to the Properties Panel and click the icon in the right column of the Decisions property to open the Decision Processing window.



Name : PURCHASE

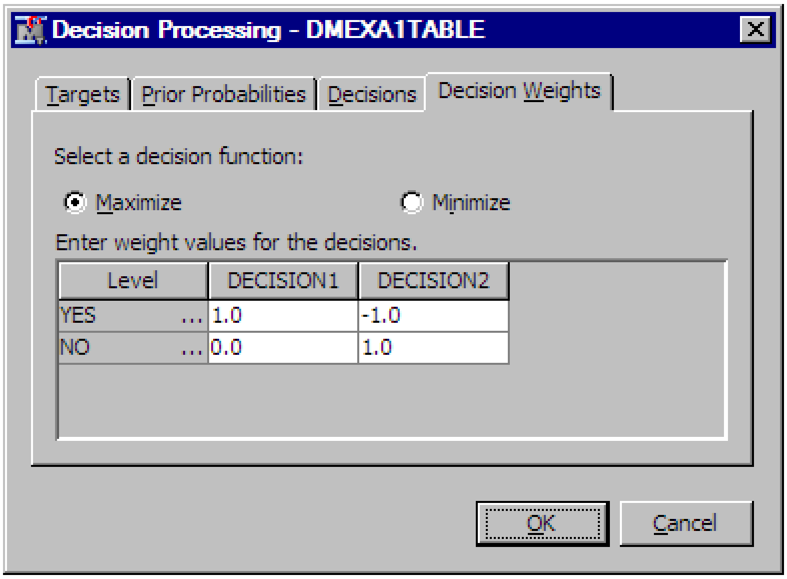
Measurement Level : Binary

Target level order : Descending

Event level : YES

Format : YESNO.

Verify that the Decisions tab shows Decisions are set to Yes, then select the Decision Weights tab.



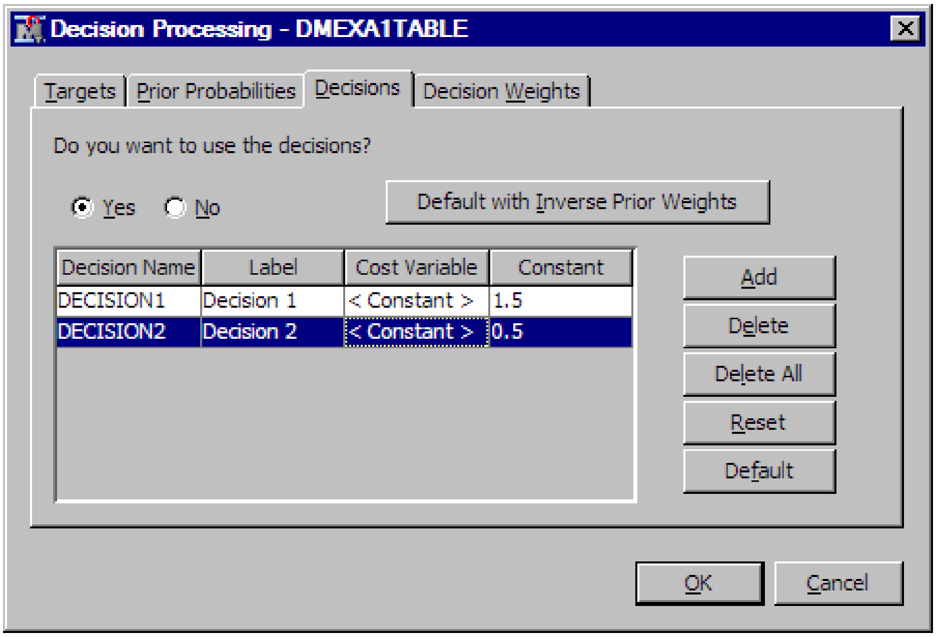
Select a decision function:

● Maximize ○ Minimize

Enter weight values for the decisions.

Level	DECISION1	DECISION2
YES ...	1.0	-1.0
NO ...	0.0	1.0

Because you are building a profit matrix, select Maximize. To insert cost information, select the Decisions tab.



Do you want to use the decisions?

● Yes ○ No

Decision Name	Label	Cost Variable	Constant
DECISION1	Decision 1	< Constant >	1.5
DECISION2	Decision 2	< Constant >	0.5

The Decisions tab of the Decision Processing - DMEXA1TABLE window shows the cost matrix that was created in Example 6.

Example 7: Create a GUI Process Flow Diagram, Export the GUI Diagram as Batch Processing Code, Specify a New Project Location, Run Batch Processing Code and Report, Modify Batch Processing Code and Report, Open the Batch-Created Project in the GUI

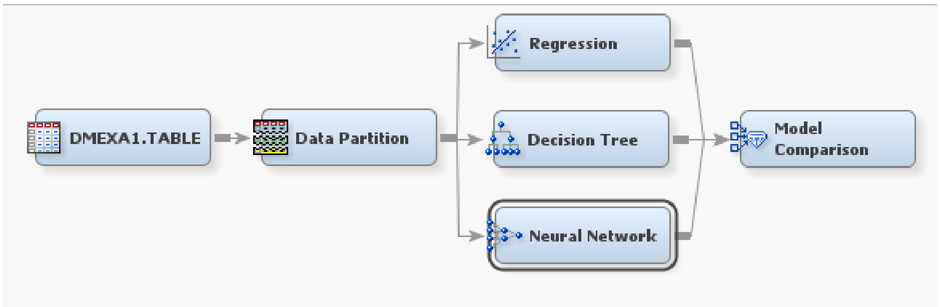
Example 7 is a comprehensive example that shows how to build, modify, and run projects from both the GUI and batch processing interfaces. The example uses components from earlier examples to show how SAS Enterprise Miner projects can be created, modified, and run in both the GUI and in the batch processing interfaces. Example 7 has parts A and B.

Example 7A adds a new workspace to the project and builds a new process flow diagram using the SAS Enterprise Miner GUI. The new process flow diagram uses the DMEXA1TABLE data source that was created in Example 4 and also used in Examples 5 and 6. First, the GUI process flow diagram is exported as batch processing code with a report. Then the exported batch processing code is modified so that it creates a new project workspace, runs, and then creates a report. The workspace in the new project is purged, which results in an empty project that contains reports only.

Some modeling node properties are changed in the batch processing code and it runs again, creating a second report in the project folder before purging the workspace again. Then, you use the GUI to open the CodeTutor project and see the new workspace that you created using the batch processing interface. The new workspace in the CodeTutor project should contain the new reports, but there will be no diagrams or workspaces.

Example 7A Diagram Overview:

Begin by opening the CodeTutor project in the SAS Enterprise Miner GUI. Right-click the Diagrams folder in the Project Navigator and create a new workspace called EMWS2. Open the EMWS2 workspace and construct the following process flow diagram. Use the DMEXA1TABLE data source that was created in Example 4.



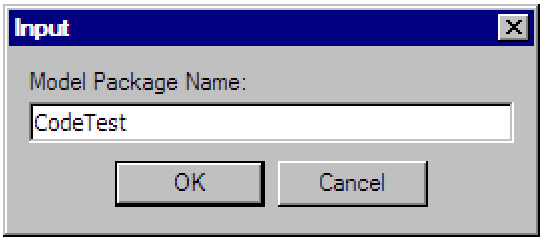
Example 7 process flow diagram

Example 7A Node Properties Overview

Use the current metadata settings in the DMEXA1TABLE node, and use the default settings for the rest of the nodes in the initial process flow diagram.

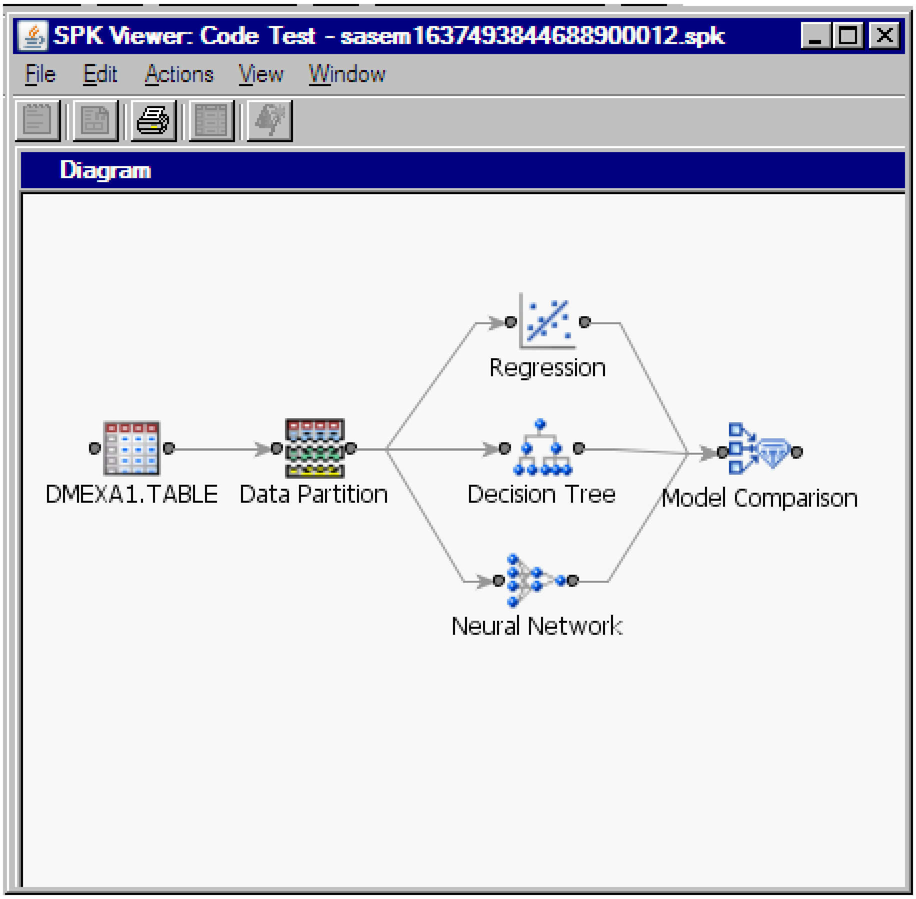
Export the Example 7A Model:

Right-click the Model Comparison node, and select Run to run the process flow diagram for Example 7. When the diagram run completes, close the dialog boxes, then right-click the Model Comparison node again, and from the pop-up menu, select Create Model Package.



Model Package Name: CodeTest

Enter a name for your model package and select **OK**. After a moment, the model package that you created will appear in the Project Navigator, in the Model Packages folder. You can double click on the model package name and open the SPK viewer for your package:



To see the batch code that is associated with the package file you just created, from the SPK Viewer window menu, select **View** ⇨ **Batch Code** ⇨ **SAS Code**.

The SAS batch code that you generate in the SAS Enterprise Miner GUI, and then save as a package file should resemble the code below. The code represents the process flow diagram that was created and exported as a package, in the CodeTutor project in the EMWS2 workspace.

Modify and Submit the Example 7A Exported Code:

Now modify the batch processing code so that it creates a new project called CodeTest and creates a process flow diagram in a new workspace called EMWS. When the process flow runs, a Model Compare report is generated.

The exported batch processing code is listed below.

The modifications will change the following items:

- the SAS Enterprise Miner project path
- the project name
- the workspace name

Changes to the generated batch processing code are shown in a lighter shade of color. After you cut and paste the code below, make your own path specifications and submit the code.

Note: All the changes are in the first three lines of the batch processing code.

```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 7A***    ;
```

```
*------------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71

%let EM_PROJECTNAME = CodeTest;

%let EM_WSNAME = EMWS;

%let EM_SUMMARY =WORK.SUMMARY;
%let EM_NUMTASKS =SINGLE;
%let EM_EDITMODE =R;
%let EM_DEBUG =;
%let EM_ACTION =run report;


*------------------------------------------------------------*;
* Create workspace data set;
*------------------------------------------------------------*;

data workspace;
length property $64 value $100;

property= 'PROJECTLOCATION';  value= "&EM_PROJECT";      output;
property= 'PROJECTNAME';      value= "&EM_PROJECTNAME";  output;
property= 'WORKSPACENAME';    value= "&EM_WSNAME";       output;
property= 'SUMMARYDATASET';   value= "&EM_SUMMARY";      output;
property= 'NUMTASKS';         value= "&EM_NUMTASKS";     output;
property= 'EDITMODE';         value= "&EM_EDITMODE";     output;
property= 'DEBUG';            value= "&&EM_DEBUG";       output;

run;
*------------------------------------------------------------*;
* Create nodes data set;
*------------------------------------------------------------*;

data nodes;
length id $12 component $32 description $64;

id= "MdlComp"; component="ModelCompare";  description= "Model Comparison"; output;
id= "Neural";  component="NeuralNetwork"; description= "Neural Network";   output;
id= "Tree";    component="DecisionTree";  description= "Decision Tree";    output;
id= "Reg";     component="Regression";    description= "Regression";       output;
id= "Part";    component="Partition";     description= "Data Partition";   output;
id= "Ids";     component="DataSource";    description= "dmexa1table";      output;

run;
*------------------------------------------------------------*;
* Data Source Information for Ids;
* Id: DMEXA1TABLE;
* Libname: SAMPSIO;
* Path: ('C:\Program Files\SAS\SAS System\9.2\core\sample'
*        'C:\Program Files\SAS\SAS System\9.2\risk\sample'
*        'C:\Program Files\SAS\SAS System\9.2\irp\sample'
*        'C:\Program Files\SAS\SAS System\9.2\eis\sample';
*------------------------------------------------------------*;
```

```
*------------------------------------------------------------*;
* DataSource Properties;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_P;
length Property $ 32 Value $ 200;

infile cards dsd;
input Property $ Value $;
informat Property $CHAR32. Value;

cards;
    Name,dmexa1table
    CreateDate,1366358663
    ModifyDate,1366799630.6
    CreatedBy,emtest
    ModifiedBy,emtest
;

run;
*------------------------------------------------------------*;
* Columns Metadata;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_CM;
length NAME $ 64
ROLE $ 32
LEVEL $ 10
ORDER $ 8
CREATOR $ 32
FORMATTYPE $ 10
FAMILY $ 10
LOWERLIMIT 8
UPPERLIMIT 8
REPORT $ 1
DISTRIBUTION $ 20
COMMENT $ 64
PRICE 8
TYPE $ 1
LABEL $ 200
FORMAT $ 20
INFORMAT $ 20
INDEX $ 1
INDEXTYPE $ 9
LENGTH 8
DROP $ 1
;

infile cards dsd;
input NAME $
ROLE $
LEVEL $
ORDER $
CREATOR $
FORMATTYPE $
FAMILY $
```

```
LOWERLIMIT
UPPERLIMIT
REPORT $
DISTRIBUTION $
COMMENT $
PRICE
TYPE $
LABEL $
FORMAT $
INFORMAT $
INDEX $
INDEXTYPE $
LENGTH
DROP $
;

label LABEL="Variable Label"
FORMAT="Variable Format"
INFORMAT="Variable Informat"
LENGTH="Variable Length"
;

informat NAME $CHAR64.
ROLE $CHAR32.
LEVEL $CHAR10.
ORDER $CHAR8.
CREATOR $CHAR32.
FORMATTYPE $CHAR10.
FAMILY $CHAR10.
REPORT $CHAR1.
DISTRIBUTION $CHAR20.
COMMENT $CHAR64.
TYPE $CHAR1.
LABEL
FORMAT $CHAR20.
INFORMAT $CHAR20.
INDEX $CHAR1.
INDEXTYPE $CHAR9.
DROP $CHAR1.
;

cards;
ACCTNUM,REJECTED,NOMINAL,,,,,.,.,N,,,.,C,Account Number,,,N,NONE,10,N
AGE,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Age,,,N,NONE,8,N
AMOUNT,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Dollars Spent,,,N,NONE,8,N
APPAREL,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Apparel Purch.,,,N,NONE,8,N
APRTMNT,INPUT,BINARY,,,CATEGORY,,.,.,N,,,.,C,Rents Apartment,$YESNO.,,N,NONE,3,N
BLANKETS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Blankets Purch.,,,N,NONE,8,N
COATS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Coats Purch.,,,N,NONE,8,N
COUNTY,INPUT,INTERVAL,,,,,.,.,N,,,.,N,County Code,,,N,NONE,8,N
CUSTDATE,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Date 1st Order,,,N,NONE,8,N
DISHES,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Dishes Purch.,,,N,NONE,8,N
DOMESTIC,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Domestic Prod.,,,N,NONE,8,N
DPM12,INPUT,INTERVAL,,,,,.,.,N,,,.,N,$ Value/Mailing,,,N,NONE,8,N
EDLEVEL,INPUT,NOMINAL,,,USER,,.,.,N,,,.,N,,EDFMT.,,N,NONE,8,N
FLATWARE,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Flatware Purch.,,,N,NONE,8,N
```

```
FREQUENT,FREQ,INTERVAL,,,,,.,.,N,,,.,N,Order Frequency,,N,NONE,8,N
GENDER,INPUT,BINARY,,,,,.,.,N,,,.,C,Gender,,N,NONE,6,N
HEAT,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,HEATFMT.,,N,NONE,8,N
HHAPPAR,INPUT,NOMINAL,,,,,.,.,N,,,.,N,His/Her Apparel,,N,NONE,8,N
HOMEACC,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Home Furniture,,N,NONE,8,N
HOMEVAL,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Home Value,,N,NONE,8,N
INCOME,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Yearly Income,,N,NONE,8,N
JEWELRY,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Jewelry Purch.,,,N,NONE,8,N
JOB,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,JOBFMT.,,N,NONE,8,N
KITCHEN,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Kitchen Prod.,,,N,NONE,8,N
LAMPS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Lamps Purch.,,,N,NONE,8,N
LEISURE,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Leisure Prod.,,,N,NONE,8,N
LINENS,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Linens Purch.,,,N,NONE,8,N
LUXURY,INPUT,BINARY,,,,,.,.,N,,,.,N,Luxury Items,,N,NONE,8,N
MARITAL,INPUT,BINARY,,USER,,.,.,N,,,.,N,Married (y/n),YESNO.,,N,NONE,8,N
MENSWARE,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Mens Apparel,,N,NONE,8,N
MOBILE,INPUT,BINARY,,USER,,.,.,N,,,.,N,Occupied <1 yr,YESNO.,,N,NONE,8,N
NTITLE,INPUT,NOMINAL,,,,,.,.,N,,,.,C,Name Prefix,,N,NONE,4,N
NUMCARS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,,,,N,NONE,8,N
NUMKIDS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,,,,N,NONE,8,N
ORIGIN,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,ORIGFMT.,,N,NONE,8,N
OUTDOOR,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Outdoor Prod.,,,N,NONE,8,N
PROMO13,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Promo: 8-13 mon,,N,NONE,8,N
PROMO7,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Promo: 1-7 mon.,,,N,NONE,8,N
PURCHASE,TARGET,BINARY,,USER,,.,.,N,,,.,N,Purchase (y/n),YESNO.,,N,NONE,8,N
RACE,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,RACEFMT.,,N,NONE,8,N
RECENCY,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Recency,,N,NONE,8,N
RETURN,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Total Returns,,N,NONE,8,N
SNGLMOM,INPUT,BINARY,,USER,,.,.,N,,,.,N,Single Mom,YESNO.,,N,NONE,8,N
STATECOD,REJECTED,NOMINAL,,,,,.,.,N,,,.,C,State Code,,N,NONE,2,N
TELIND,INPUT,BINARY,,CATEGORY,,.,.,N,,,.,C,Telemarket Ind.,$YESNO.,,N,NONE,3,N
TMKTORD,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Telemarket Ord.,,,N,NONE,8,N
TOWELS,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Towels Purch.,,,N,NONE,8,N
TRAVTIME,INPUT,INTERVAL,,,,,.,.,N,,,.,N,,,,N,NONE,8,N
WAPPAR,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Ladies Apparel,,N,NONE,8,N
WCOAT,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Ladies Coats,,N,NONE,8,N
;

run;

*------------------------------------------------------------*;
* Table Metadata;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_TM;
length MEMNAME $ 32
MEMTYPE $ 8
MEMLABEL $ 40
TYPEMEM $ 8
ENGINE $ 8
CRDATE 8
MODATE 8
NOBS 8
NCOLS 8
ROLE $ 20
USEEXTERNALDATA $ 1
```

```
SAMPLINGRATE 8
SEGMENT $ 20
libname $ 8
;

infile cards dsd;
input MEMNAME $
MEMTYPE $
MEMLABEL $
TYPEMEM $
ENGINE $
CRDATE
MODATE
NOBS
NCOLS
ROLE $
USEEXTERNALDATA $
SAMPLINGRATE
SEGMENT $
libname $
;

format CRDATE DATETIME16.
MODATE DATETIME16.
;

informat MEMNAME $CHAR32.
MEMTYPE $CHAR8.
MEMLABEL $CHAR40.
TYPEMEM $CHAR8.
ENGINE $CHAR8.
ROLE $CHAR20.
USEEXTERNALDATA $CHAR1.
SEGMENT $CHAR20.
libname $CHAR8.
;

cards;
DMEXA1,DATA,,DATA,V9,1267003746.567,
  1267003746.567,1966,50,TRAIN,,.,,SAMPSIO
;

run;
*------------------------------------------------------------*;
* Target Profile;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_TP;
length ProfileID 8
ProfileName $ 64
Target $ 64
Level $ 16
Use $ 8
;

infile cards dsd;
```

```
input ProfileID
ProfileName $
Target $
Level $
Use $
;

informat ProfileName $CHAR64.
Target $CHAR64.
Level $CHAR16.
Use $CHAR8.
;

cards;
0,Profile 0,PURCHASE,,Y
;
run;

data WORK.DMEXA1TABLE_M0;
length _TYPE_ $ 32
VARIABLE $ 32
LABEL $ 40
LEVEL $ 32
EVENT $ 32
ORDER $ 10
FORMAT $ 32
TYPE $ 1
COST $ 32
USE $ 1
;

infile cards dsd;
input _TYPE_ $
VARIABLE $
LABEL $
LEVEL $
EVENT $
ORDER $
FORMAT $
TYPE $
COST $
USE $
;

label _TYPE_="Type"
VARIABLE="Variable"
LABEL="Label"
LEVEL="Measurement Level"
EVENT="Target Event"
ORDER="Order"
FORMAT="Format"
TYPE="Type"
COST="Cost"
USE="Use"
;
```

```
informat _TYPE_ $CHAR32.
VARIABLE $CHAR32.
LABEL $CHAR40.
LEVEL $CHAR32.
EVENT $CHAR32.
ORDER $CHAR10.
FORMAT $CHAR32.
TYPE $CHAR1.
COST $CHAR32.
USE $CHAR1.
;

cards;
MATRIX,,,PROFIT,,,,,,Y
TARGET,PURCHASE,Purchase (y/n),BINARY,YES,,YESNO.,N,,
DECISION,DECISION1,YES,,,,,N,1.5,Y
DECISION,DECISION2,NO,,,,,N,2.0,Y
DATAPRIOR,DATAPRIOR,Data Prior,,,,,N,,Y
TRAINPRIOR,TRAINPRIOR,Training Prior,,,,,N,,N
DECPRIOR,DECPRIOR,Decision Prior,,,,,N,,N
PREDICTED,P_PURCHASEYes,Predicted: PURCHASE=Yes,YES,,,,N,,
RESIDUAL,R_PURCHASEYes,Residual: PURCHASE=Yes,YES,,,,N,,
PREDICTED,P_PURCHASENo,Predicted: PURCHASE=No,NO,,,,N,,
RESIDUAL,R_PURCHASENo,Residual: PURCHASE=No,NO,,,,N,,
FROM,F_PURCHASE,From: PURCHASE,,,,,C,,
INTO,I_PURCHASE,Into: PURCHASE,,,,,C,,
FREQ,FREQUENT,Order Frequency,INTERVAL,,,,N,,
;
run;

data WORK.DMEXA1TABLE_D0;
length PURCHASE $ 32
COUNT 8
DATAPRIOR 8
TRAINPRIOR 8
DECPRIOR 8
DECISION1 8
DECISION2 8
;

infile cards dsd;
input PURCHASE $
COUNT
DATAPRIOR
TRAINPRIOR
DECPRIOR
DECISION1
DECISION2
;

label COUNT="Level Counts"
DATAPRIOR="Data Proportions"
TRAINPRIOR="Training Proportions"
DECPRIOR="Decision Priors"
DECISION1="YES"
DECISION2="NO"
```

```
;

informat PURCHASE $CHAR32.
;

cards;
YES,3719.68,0.61607667833227,0.61607667833227,0,1,-1
NO,2318.00999999999,0.38392332166772,0.38392332166772,0,0,1
;

run;
*------------------------------------------------------------*;
* Create Datasource data set;
*------------------------------------------------------------*;

data datasources;
length id $30 key $40 value $64;

id="DMEXA1TABLE"; key="Properties";    value="WORK.DMEXA1TABLE_P";              output;
id="DMEXA1TABLE"; key="ColumnMeta";    value="WORK.DMEXA1TABLE_CM";             output;
id="DMEXA1TABLE"; key="TableMeta";     value="WORK.DMEXA1TABLE_TM";             output;
id="DMEXA1TABLE"; key="TargetProfile"; value="WORK.DMEXA1TABLE_TP";             output;
id="DMEXA1TABLE"; key="TargetProfile_DM_PURCHASE"; value="WORK.DMEXA1TABLE_M0"; output;
id="DMEXA1TABLE"; key="TargetProfile_DD_PURCHASE"; value="WORK.DMEXA1TABLE_D0"; output;
*------------------------------------------------------------*;
* Variable Attributes for Ids;
*------------------------------------------------------------*;

data WORK.Ids_VariableAttribute;
length Variable $64 AttributeName $32 AttributeValue $64;

Variable="FREQUENT"; AttributeName="ROLE"; AttributeValue="REJECTED"; Output;

run;
*------------------------------------------------------------*;
* Decmeta Data Set for Ids;
*------------------------------------------------------------*;

data WORK.Ids_PURCHASE_DM;
length _TYPE_ $ 32
VARIABLE $ 32
LABEL $ 40
LEVEL $ 32
EVENT $ 32
ORDER $ 10
FORMAT $ 32
TYPE $ 1
COST $ 32
USE $ 1
;

infile cards dsd;
input _TYPE_ $
VARIABLE $
LABEL $
LEVEL $
```

```
EVENT $
ORDER $
FORMAT $
TYPE $
COST $
USE $
;

label _TYPE_="Type"
VARIABLE="Variable"
LABEL="Label"
LEVEL="Measurement Level"
EVENT="Target Event"
ORDER="Order"
FORMAT="Format"
TYPE="Type"
COST="Cost"
USE="Use"
;

informat _TYPE_ $CHAR32.
VARIABLE $CHAR32.
LABEL $CHAR40.
LEVEL $CHAR32.
EVENT $CHAR32.
ORDER $CHAR10.
FORMAT $CHAR32.
TYPE $CHAR1.
COST $CHAR32.
USE $CHAR1.
;

cards;
MATRIX,,,PROFIT,,,,,,Y
TARGET,PURCHASE,Purchase (y/n),BINARY,YES,,YESNO.,N,,
DECISION,DECISION1,YES,,,,,N,1.5,Y
DECISION,DECISION2,NO,,,,,N,2.0,Y
DATAPRIOR,DATAPRIOR,Data Prior,,,,,N,,Y
TRAINPRIOR,TRAINPRIOR,Training Prior,,,,,N,,N
DECPRIOR,DECPRIOR,Decision Prior,,,,,N,,N
PREDICTED,P_PURCHASEYes,Predicted: PURCHASE=Yes,YES,,,,N,,
RESIDUAL,R_PURCHASEYes,Residual: PURCHASE=Yes,YES,,,,N,,
PREDICTED,P_PURCHASENo,Predicted: PURCHASE=No,NO,,,,N,,
RESIDUAL,R_PURCHASENo,Residual: PURCHASE=No,NO,,,,N,,
FROM,F_PURCHASE,From: PURCHASE,,,,,C,,
INTO,I_PURCHASE,Into: PURCHASE,,,,,C,,
FREQ,FREQUENT,Order Frequency,INTERVAL,,,,N,,
MODELDECISION,D_PURCHASE,Decision: PURCHASE,INTERVAL,,,,N,,
EXPECTEDPROFIT,EP_PURCHASE,Expected Profit: PURCHASE,INTERVAL,,,,N,,
COMPUTEDPROFIT,CP_PURCHASE,Computed Profit: PURCHASE,INTERVAL,,,,N,,
BESTPROFIT,BP_PURCHASE,Best Profit: PURCHASE,INTERVAL,,,,N,,
INVESTMENTCOST,IC_PURCHASE,Investment Cost: PURCHASE,INTERVAL,,,,N,,
ROI,ROI_PURCHASE,Return on Investment: PURCHASE,INTERVAL,,,,N,,
;
run;
*------------------------------------------------------------*;
```

```
* Decdata Data Set for Ids;
*------------------------------------------------------------*;

data WORK.Ids_PURCHASE_DD;
length PURCHASE $ 32
COUNT 8
DATAPRIOR 8
TRAINPRIOR 8
DECPRIOR 8
DECISION1 8
DECISION2 8
;

infile cards dsd;
input PURCHASE $
COUNT
DATAPRIOR
TRAINPRIOR
DECPRIOR
DECISION1
DECISION2
;

label COUNT="Level Counts"
DATAPRIOR="Data Proportions"
TRAINPRIOR="Training Proportions"
DECPRIOR="Decision Priors"
DECISION1="YES"
DECISION2="NO"
;

informat PURCHASE $CHAR32.
;

cards;
YES,3719.68,0.61607667833227,0.61607667833227,0,1,-1
NO,2318.00999999999,0.38392332166772,0.38392332166772,0,0,1
;

run;
*------------------------------------------------------------*;
* Create node properties data set;
*------------------------------------------------------------*;

data nodeprops;
length id $12 property $32 value $64;

id= "MdlComp"; property="NumberOfReportedLevels";      value= "1E-6";         output;
id= "MdlComp"; property="NormalizeReportingVariables"; value= "Y";            output;
id= "MdlComp"; property="DecileBin";                   value= "20";           output;
id= "MdlComp"; property="LiftEpsilon";                 value= "1E-6";         output;
id= "MdlComp"; property="ProfitEpsilon";               value= "1E-6";         output;
id= "MdlComp"; property="RoiEpsilon";                  value= "1E-6";         output;
id= "MdlComp"; property="ScoreDistBin";                value= "20";           output;
id= "MdlComp"; property="RocChart";                    value= "Y";            output;
id= "MdlComp"; property="RocEpsilon";                  value= "0.01";         output;
```

```
id= "MdlComp"; property="AssessAllTargetLevels";       value= "N";            output;
id= "MdlComp"; property="SelectionStatistic";          value= "_TMISC_";      output;
id= "MdlComp"; property="Component";                   value= "ModelCompare"; output;
id= "MdlComp"; property="StatisticUsed";               value= "_TMISC_";      output;
id= "Neural";  property="NetworkArchitecture";         value= "MLP";          output;
id= "Neural";  property="DirectConnection";            value= "N";            output;
id= "Neural";  property="Hidden";                      value= "3";            output;
id= "Neural";  property="Prelim";                      value= "N";            output;
id= "Neural";  property="PreliminaryRuns";             value= "5";            output;
id= "Neural";  property="PrelimMaxiter";               value= "10";           output;
id= "Neural";  property="PrelimMaxTime";               value= "1 HOUR";       output;
id= "Neural";  property="Maxiter";                     value= "20";           output;
id= "Neural";  property="Maxtime";                     value= "4 HOURS";      output;
id= "Neural";  property="TrainingTechnique";           value= "DEFAULT";      output;
id= "Neural";  property="ConvDefaults";                value= "Y";            output;
id= "Neural";  property="AbsConvValue";                value= "-1.34078E154"; output;
id= "Neural";  property="AbsFValue";                   value= "0";            output;
id= "Neural";  property="AbsFTime";                    value= "1";            output;
id= "Neural";  property="AbsGValue";                   value= "0.00001";      output;
id= "Neural";  property="AbsGTime";                    value= "1";            output;
id= "Neural";  property="AbsXValue";                   value= "1E-8";         output;
id= "Neural";  property="AbsXTime";                    value= "1";            output;
id= "Neural";  property="FConvValue";                  value= "0";            output;
id= "Neural";  property="FConvTime";                   value= "1";            output;
id= "Neural";  property="GConvValue";                  value= "1E-6";         output;
id= "Neural";  property="GConvTime";                   value= "1";            output;
id= "Neural";  property="ModelSelectionCriterion";     value= "PROFIT/LOSS";  output;
id= "Neural";  property="SuppressOutput";              value= "N";            output;
id= "Neural";  property="Residuals";                   value= "Y";            output;
id= "Neural";  property="Standardizations";            value= "N";            output;
id= "Neural";  property="HiddenUnits";                 value= "N";            output;
id= "Neural";  property="TrainCode";                   value= "";             output;
id= "Neural";  property="PrelimOutest";                value= "";             output;
id= "Neural";  property="Outest";                      value= "";             output;
id= "Neural";  property="Outfit";                      value= "";             output;
id= "Neural";  property="InitialDs";                   value= "";             output;
id= "Neural";  property="CodefileRes";                 value= "";             output;
id= "Neural";  property="CodefileNoRes";               value= "";             output;
id= "Neural";  property="Component";                   value= "NeuralNetwork"; output;
id= "Tree";    property="TrainMode";                   value= "BATCH";        output;
id= "Tree";    property="Criterion";                   value= "DEFAULT";      output;
id= "Tree";    property="SigLevel";                    value= "0.2";          output;
id= "Tree";    property="Splitsize";                   value= ".";            output;
id= "Tree";    property="LeafSize";                    value= "5";            output;
id= "Tree";    property="MinCatSize";                  value= "5";            output;
id= "Tree";    property="Maxbranch";                   value= "2";            output;
id= "Tree";    property="Maxdepth";                    value= "6";            output;
id= "Tree";    property="Nrules";                      value= "5";            output;
id= "Tree";    property="Nsurrs";                      value= "0";            output;
id= "Tree";    property="MissingValue";                value= "USEINSEARCH";  output;
id= "Tree";    property="UseVarOnce";                  value= "N";            output;
id= "Tree";    property="Subtree";                     value= "ASSESSMENT";   output;
id= "Tree";    property="NSubtree";                    value= "1";            output;
id= "Tree";    property="AssessMeasure";               value= "PROFIT/LOSS";  output;
id= "Tree";    property="AssessPercentage";            value= "0.25";         output;
id= "Tree";    property="NodeSample";                  value= "5000";         output;
```

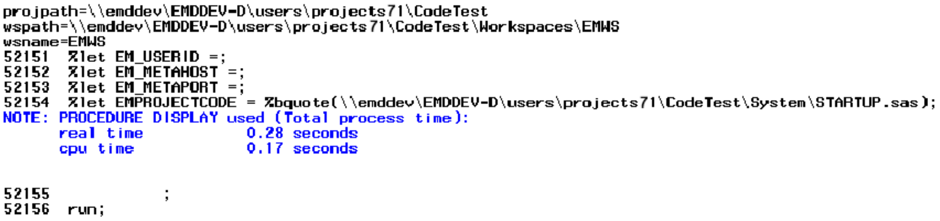
```
id= "Tree";    property="Exhaustive";               value= "5000";         output;
id= "Tree";    property="UseDecision";              value= "N";            output;
id= "Tree";    property="UsePriors";                value= "N";            output;
id= "Tree";    property="Kass";                     value= "Y";            output;
id= "Tree";    property="KassApply";                value= "BEFORE";       output;
id= "Tree";    property="Depth";                    value= "Y";            output;
id= "Tree";    property="Inputs";                   value= "N";            output;
id= "Tree";    property="NumInputs";                value= "1";            output;
id= "Tree";    property="VarSelection";             value= "Y";            output;
id= "Tree";    property="Dummy";                    value= "N";            output;
id= "Tree";    property="Leafid";                   value= "Y";            output;
id= "Tree";    property="Predict";                  value= "Y";            output;
id= "Tree";    property="NodeRole";                 value= "SEGMENT";      output;
id= "Tree";    property="Component";                value= "DecisionTree"; output;
id= "Reg";     property="MainEffect";               value= "Y";            output;
id= "Reg";     property="TwoFactor";                value= "N";            output;
id= "Reg";     property="Polynomial";               value= "N";            output;
id= "Reg";     property="PolynomialDegree";         value= "2";            output;
id= "Reg";     property="Terms";                    value= "N";            output;
id= "Reg";     property="Error";                    value= "LOGISTIC";     output;
id= "Reg";     property="LinkFunction";             value= "LOGIT";        output;
id= "Reg";     property="SuppressIntercept";        value= "N";            output;
id= "Reg";     property="InputCoding";              value= "DEVIATION";    output;
id= "Reg";     property="MinResourceUse";           value= "D";            output;
id= "Reg";     property="ModelSelection";           value= "NONE";         output;
id= "Reg";     property="SelectionCriterion";       value= "DEFAULT";      output;
id= "Reg";     property="SelectionDefault";         value= "Y";            output;
id= "Reg";     property="Sequential";               value= "N";            output;
id= "Reg";     property="SlEntry";                  value= "0.05";         output;
id= "Reg";     property="SlStay";                   value= "0.05";         output;
id= "Reg";     property="Start";                    value= "0";            output;
id= "Reg";     property="Stop";                     value= "0";            output;
id= "Reg";     property="Force";                    value= "0";            output;
id= "Reg";     property="Hierarchy";                value= "CLASS";        output;
id= "Reg";     property="Rule";                     value= "NONE";         output;
id= "Reg";     property="MaxStep";                  value= ".";            output;
id= "Reg";     property="StepOutput";               value= "N";            output;
id= "Reg";     property="OptimizationTechnique";    value= "DEFAULT";      output;
id= "Reg";     property="ModelDefaults";            value= "Y";            output;
id= "Reg";     property="MaxIterations";            value= ".";            output;
id= "Reg";     property="MaxFunctionCalls";         value= ".";            output;
id= "Reg";     property="MaxTime";                  value= "604800";       output;
id= "Reg";     property="ConvDefaults";             value= "Y";            output;
id= "Reg";     property="AbsConvValue";             value= "-1.34078E154"; output;
id= "Reg";     property="AbsFValue";                value= "0";            output;
id= "Reg";     property="AbsFTime";                 value= "1";            output;
id= "Reg";     property="AbsGValue";                value= "0.00001";      output;
id= "Reg";     property="AbsGTime";                 value= "1";            output;
id= "Reg";     property="AbsXValue";                value= "1E-8";         output;
id= "Reg";     property="AbsXTime";                 value= "1";            output;
id= "Reg";     property="FConvValue";               value= "0";            output;
id= "Reg";     property="FConvTimes";               value= "1";            output;
id= "Reg";     property="GConvValue";               value= "1E-6";         output;
id= "Reg";     property="GConvTimes";               value= "1";            output;
id= "Reg";     property="UseInEst";                 value= "N";            output;
id= "Reg";     property="InEstDataset";             value= "";             output;
```

```
id= "Reg";     property="ClParm";                      value= "N";            output;
id= "Reg";     property="CovB";                        value= "N";            output;
id= "Reg";     property="CorB";                        value= "N";            output;
id= "Reg";     property="Simple";                      value= "N";            output;
id= "Reg";     property="SuppressOutput";              value= "N";            output;
id= "Reg";     property="Details";                     value= "N";            output;
id= "Reg";     property="PrintDesignMatrix";           value= "N";            output;
id= "Reg";     property="SASSPDS";                     value= "N";            output;
id= "Reg";     property="Performance";                 value= "N";            output;
id= "Reg";     property="Component";                   value= "Regression";   output;
id= "Part";    property="Method";                      value= "DEFAULT";      output;
id= "Part";    property="TrainPct";                    value= "40";           output;
id= "Part";    property="ValidatePct";                 value= "30";           output;
id= "Part";    property="TestPct";                     value= "30";           output;
id= "Part";    property="RandomSeed";                  value= "12345";        output;
id= "Part";    property="Component";                   value= "Partition";    output;
id= "Ids";     property="DataSource";                  value= "DMEXA1TABLE";  output;
id= "Ids";     property="Scope";                       value= "LOCAL";        output;
id= "Ids";     property="Role";                        value= "TRAIN";        output;
id= "Ids";     property="Library";                     value= "SAMPSIO";      output;
id= "Ids";     property="Table";                       value= "DMEXA1";       output;
id= "Ids";     property="NCols";                       value= "50";           output;
id= "Ids";     property="NObs";                        value= "1966";         output;
id= "Ids";     property="SamplingRate";                value= ".";            output;
id= "Ids";     property="Segment";                     value= "";             output;
id= "Ids";     property="UseExternal";                 value= "";             output;
id= "Ids";     property="OutputType";                  value= "VIEW";         output;
id= "Ids";     property="ForceRun";                    value= "";             output;
id= "Ids";     property="Component";                   value= "DataSource";   output;
id= "Ids";     property="Description";                 value= "";             output;
id= "Ids";     property="UseExternalData";             value= "";             output;
id= "Ids"; property="EM_VARIABLEATTRIBUTES"; value= "WORK.Ids_VariableAttribute"; output
id= "Ids";  property="EM_DECMETA_PURCHASE";    value= "WORK.Ids_PURCHASE_DM"; output;
id= "Ids";  property="EM_DECDATA_PURCHASE";    value= "WORK.Ids_PURCHASE_DD"; output;

run;
*------------------------------------------------------------*;
* Create connections data set;
*------------------------------------------------------------*;

data connect;
length from to $12;

input from to;

cards;
   Neural MdlComp
   Tree MdlComp
   Reg MdlComp
   Part Reg
   Part Tree
   Part Neural
   Ids Part
;

run;
```

```
*------------------------------------------------------------*;
* Create actions to run data set;
*------------------------------------------------------------*;

%macro emaction;
%let actionstring = %upcase(&EM_ACTION);
%if %index(&actionstring, RUN) or %index(&actionstring, REPORT) %then
%do;

data actions;
length id $12 action $40; id="MdlComp";

   %if %index(&actionstring, RUN) %then %do;
   action='run'; output;
   %end;

   %if %index(&actionstring, REPORT) %then %do;
   action='report'; output;
   %end;

run;
%end;
%mend;
%emaction;
*------------------------------------------------------------*;
* Execute the actions;
*------------------------------------------------------------*;

%EM5BATCH(execute,
          workspace=workspace,
          nodes=nodes,
          connect=connect,
          datasources=datasources,
          nodeprops=nodeprops,
          action=actions)
          ;
run;
```

Verify the Example 7A New Project Files and Output

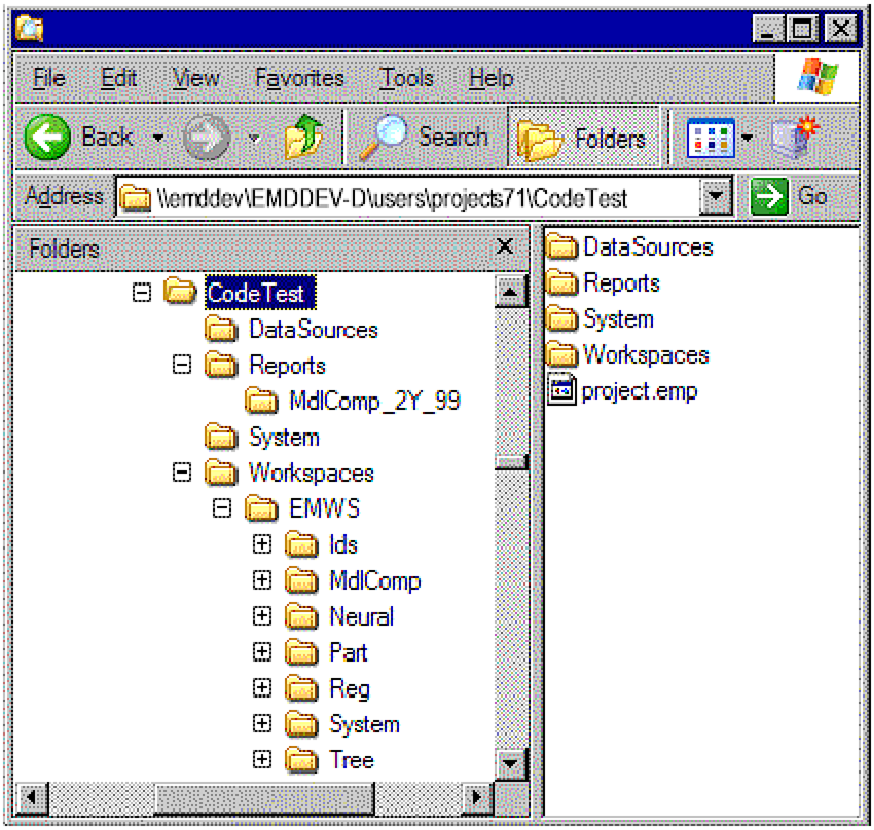
After you submit and successfully run your modified Example 7 batch code, save it to a location of your choice. The SAS log confirms the new project and workspace paths that were created by the batch processing code:



```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTest
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTest\Workspaces\EMWS
wsname=EMWS
52151  %let EM_USERID =;
52152  %let EM_METAHOST =;
52153  %let EM_METAPORT =;
52154  %let EMPROJECTCODE = %bquote(\\emddev\EMDDEV-D\users\projects71\CodeTest\System\STARTUP.sas);
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           0.28 seconds
      cpu time            0.17 seconds


52155           ;
52156  run;
```

Note: The line numbering, paths, and process times in your log will vary from the example.

You can verify the new files in a file utility:



The CodeTest project file structure has been created under the appropriate path on the server. The DataSources folder is empty, and the Reports folder holds the report that the process flow diagram created in the newly created workspace EMWS.

Purge the Example 7A Project Workspace

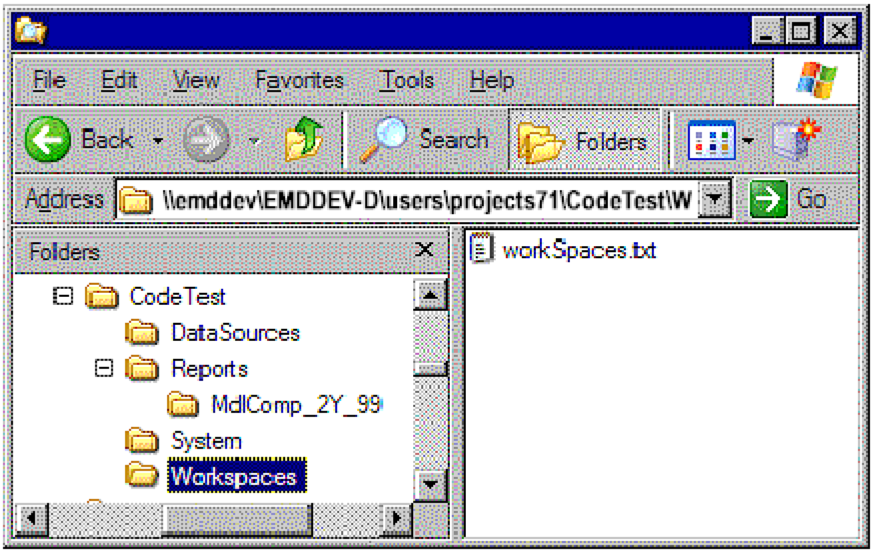
Now purge the project workspace EMWS from the CodeTest project. Submit the following code in the same SAS session that you used to run the batch processing code:

```
%em5batch(purgeworkspace);
run;
```

Your SAS log should resemble the following:

```
52157  %em5batch(purgeworkspace);

52158  %let EM_USERID =;
52159  %let EM_METAHOST =;
52160  %let EM_METAPORT =;
52161  %let EMPROJECTCODE = %bquote(\\emddev\EMDDEV-D\users\projects71\CodeTest\System\STARTUP.sas);
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           2.56 seconds
      cpu time            0.98 seconds


52162  run;
```

Use a file utility to confirm that the workspace was purged.



Modify the Example 7A Batch Processing Code a Second Time:

Now modify the existing Example7.sas batch processing code to do the following in Example 7B:

- Create a new workspace called EMWS2 for this run.
- Modify the Network Architecture node for the Neural node from MLP to GLIM.
- Modify the Criterion property for the Decision Tree node from DEFAULT to GINI.
- Modify the Model Selection property for the Regression node from NONE to STEPWISE.
- Append the %EM5BATCH purge workspace command to the end of the batch processing code.

The Example 7B batch processing code below shows the text of the most recent changes in a lighter shade of color. The batch processing code that SAS Enterprise Miner 12.3 generates for process flow diagrams is verbose. That means that the settings for all properties in all nodes are listed in the code. By contrast, the examples that are manually created are not verbose. They list only property settings which vary from the default setting.

```
*------------------------------------------------------------*;
*   ***SAS Enterprise Miner 12.3 Batch Processing Example 7B***    ;
*------------------------------------------------------------*;
%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTest;
%let EM_WSNAME = EMWS2;
%let EM_SUMMARY =WORK.SUMMARY;
%let EM_NUMTASKS =SINGLE;
%let EM_EDITMODE =R;
%let EM_DEBUG =;
%let EM_ACTION =run report;

*------------------------------------------------------------*;
* Create workspace data set;
*------------------------------------------------------------*;

data workspace;
length property $64 value $100;
```

```
property= 'PROJECTLOCATION';  value= "&EM_PROJECT";      output;
property= 'PROJECTNAME';      value= "&EM_PROJECTNAME";  output;
property= 'WORKSPACENAME';    value= "&EM_WSNAME";       output;
property= 'SUMMARYDATASET';   value= "&EM_SUMMARY";      output;
property= 'NUMTASKS';         value= "&EM_NUMTASKS";     output;
property= 'EDITMODE';         value= "&EM_EDITMODE";     output;
property= 'DEBUG';            value= "&&EM_DEBUG";       output;

run;

*------------------------------------------------------------*;
* Create nodes data set;
*------------------------------------------------------------*;

data nodes;
length id $12 component $32 description $64;

id= "MdlComp"; component="ModelCompare";  description= "Model Comparison"; output;
id= "Neural";  component="NeuralNetwork"; description= "Neural Network";   output;
id= "Tree";    component="DecisionTree";  description= "Decision Tree";    output;
id= "Reg";     component="Regression";    description= "Regression";       output;
id= "Part";    component="Partition";     description= "Data Partition";   output;
id= "Ids";     component="DataSource";    description= "dmexa1table";      output;

run;

*------------------------------------------------------------*;
* Data Source Information for Ids;
* Id: DMEXA1TABLE;
* Libname: SAMPSIO;
* Path: ('C:\Program Files\SAS\SAS System\9.2\core\sample'
*        'C:\Program Files\SAS\SAS System\9.2\risk\sample'
*        'C:\Program Files\SAS\SAS System\9.2\irp\sample'
*        'C:\Program Files\SAS\SAS System\9.2\eis\sample';
*------------------------------------------------------------*;

*------------------------------------------------------------*;
* DataSource Properties;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_P;
length Property $ 32 Value $ 200;

infile cards dsd;
input Property $ Value $;
informat Property $CHAR32. Value;

cards;
    Name,dmexa1table
    CreateDate,1366358663
    ModifyDate,1366799630.6
    CreatedBy,emtest
    ModifiedBy,emtest
;
```

```
run;

*------------------------------------------------------------*;
* Columns Metadata;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_CM;
length NAME $ 64
ROLE $ 32
LEVEL $ 10
ORDER $ 8
CREATOR $ 32
FORMATTYPE $ 10
FAMILY $ 10
LOWERLIMIT 8
UPPERLIMIT 8
REPORT $ 1
DISTRIBUTION $ 20
COMMENT $ 64
PRICE 8
TYPE $ 1
LABEL $ 200
FORMAT $ 20
INFORMAT $ 20
INDEX $ 1
INDEXTYPE $ 9
LENGTH 8
DROP $ 1
;

infile cards dsd;
input NAME $
ROLE $
LEVEL $
ORDER $
CREATOR $
FORMATTYPE $
FAMILY $
LOWERLIMIT
UPPERLIMIT
REPORT $
DISTRIBUTION $
COMMENT $
PRICE
TYPE $
LABEL $
FORMAT $
INFORMAT $
INDEX $
INDEXTYPE $
LENGTH
DROP $
;

label LABEL="Variable Label"
FORMAT="Variable Format"
```

```
INFORMAT="Variable Informat"
LENGTH="Variable Length"
;

informat NAME $CHAR64.
ROLE $CHAR32.
LEVEL $CHAR10.
ORDER $CHAR8.
CREATOR $CHAR32.
FORMATTYPE $CHAR10.
FAMILY $CHAR10.
REPORT $CHAR1.
DISTRIBUTION $CHAR20.
COMMENT $CHAR64.
TYPE $CHAR1.
LABEL
FORMAT $CHAR20.
INFORMAT $CHAR20.
INDEX $CHAR1.
INDEXTYPE $CHAR9.
DROP $CHAR1.
;

cards;
ACCTNUM,REJECTED,NOMINAL,,,,.,.,N,,,.,C,Account Number,,N,NONE,10,N
AGE,INPUT,INTERVAL,,,,.,.,N,,,.,N,Age,,N,NONE,8,N
AMOUNT,INPUT,INTERVAL,,,,.,.,N,,,.,N,Dollars Spent,,N,NONE,8,N
APPAREL,INPUT,INTERVAL,,,,.,.,N,,,.,N,Apparel Purch.,,N,NONE,8,N
APRTMNT,INPUT,BINARY,,CATEGORY,,.,.,N,,,.,C,Rents Apartment,$YESNO.,,N,NONE,3,N
BLANKETS,INPUT,NOMINAL,,,,.,.,N,,,.,N,Blankets Purch.,,N,NONE,8,N
COATS,INPUT,NOMINAL,,,,.,.,N,,,.,N,Coats Purch.,,N,NONE,8,N
COUNTY,INPUT,INTERVAL,,,,.,.,N,,,.,N,County Code,,N,NONE,8,N
CUSTDATE,INPUT,INTERVAL,,,,.,.,N,,,.,N,Date 1st Order,,N,NONE,8,N
DISHES,INPUT,NOMINAL,,,,.,.,N,,,.,N,Dishes Purch.,,N,NONE,8,N
DOMESTIC,INPUT,INTERVAL,,,,.,.,N,,,.,N,Domestic Prod.,,N,NONE,8,N
DPM12,INPUT,INTERVAL,,,,.,.,N,,,.,N,$ Value/Mailing,,N,NONE,8,N
EDLEVEL,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,EDFMT.,,N,NONE,8,N
FLATWARE,INPUT,NOMINAL,,,,.,.,N,,,.,N,Flatware Purch.,,N,NONE,8,N
FREQUENT,FREQ,INTERVAL,,,,.,.,N,,,.,N,Order Frequency,,N,NONE,8,N
GENDER,INPUT,BINARY,,,,.,.,N,,,.,C,Gender,,N,NONE,6,N
HEAT,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,HEATFMT.,,N,NONE,8,N
HHAPPAR,INPUT,NOMINAL,,,,.,.,N,,,.,N,His/Her Apparel,,N,NONE,8,N
HOMEACC,INPUT,INTERVAL,,,,.,.,N,,,.,N,Home Furniture,,N,NONE,8,N
HOMEVAL,INPUT,INTERVAL,,,,.,.,N,,,.,N,Home Value,,N,NONE,8,N
INCOME,INPUT,INTERVAL,,,,.,.,N,,,.,N,Yearly Income,,N,NONE,8,N
JEWELRY,INPUT,NOMINAL,,,,.,.,N,,,.,N,Jewelry Purch.,,N,NONE,8,N
JOB,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,JOBFMT.,,N,NONE,8,N
KITCHEN,INPUT,NOMINAL,,,,.,.,N,,,.,N,Kitchen Prod.,,N,NONE,8,N
LAMPS,INPUT,NOMINAL,,,,.,.,N,,,.,N,Lamps Purch.,,N,NONE,8,N
LEISURE,INPUT,NOMINAL,,,,.,.,N,,,.,N,Leisure Prod.,,N,NONE,8,N
LINENS,INPUT,INTERVAL,,,,.,.,N,,,.,N,Linens Purch.,,N,NONE,8,N
LUXURY,INPUT,BINARY,,,,.,.,N,,,.,N,Luxury Items,,N,NONE,8,N
MARITAL,INPUT,BINARY,,USER,,.,.,N,,,.,N,Married (y/n),YESNO.,,N,NONE,8,N
MENSWARE,INPUT,NOMINAL,,,,.,.,N,,,.,N,Mens Apparel,,N,NONE,8,N
MOBILE,INPUT,BINARY,,USER,,.,.,N,,,.,N,Occupied <1 yr,YESNO.,,N,NONE,8,N
NTITLE,INPUT,NOMINAL,,,,.,.,N,,,.,C,Name Prefix,,N,NONE,4,N
```

```
NUMCARS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,,,N,NONE,8,N
NUMKIDS,INPUT,NOMINAL,,,,,.,.,N,,,.,N,,,N,NONE,8,N
ORIGIN,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,ORIGFMT.,,N,NONE,8,N
OUTDOOR,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Outdoor Prod.,,,N,NONE,8,N
PROMO13,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Promo: 8-13 mon,,,N,NONE,8,N
PROMO7,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Promo: 1-7 mon.,,,N,NONE,8,N
PURCHASE,TARGET,BINARY,,USER,,.,.,N,,,.,N,Purchase (y/n),YESNO.,,N,NONE,8,N
RACE,INPUT,NOMINAL,,USER,,.,.,N,,,.,N,,RACEFMT.,,N,NONE,8,N
RECENCY,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Recency,,,N,NONE,8,N
RETURN,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Total Returns,,,N,NONE,8,N
SNGLMOM,INPUT,BINARY,,USER,,.,.,N,,,.,N,Single Mom,YESNO.,,N,NONE,8,N
STATECOD,REJECTED,NOMINAL,,,,,.,.,N,,,.,C,State Code,,,N,NONE,2,N
TELIND,INPUT,BINARY,,CATEGORY,,.,.,N,,,.,C,Telemarket Ind.,$YESNO.,,N,NONE,3,N
TMKTORD,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Telemarket Ord.,,,N,NONE,8,N
TOWELS,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Towels Purch.,,,N,NONE,8,N
TRAVTIME,INPUT,INTERVAL,,,,,.,.,N,,,.,N,,,N,NONE,8,N
WAPPAR,INPUT,INTERVAL,,,,,.,.,N,,,.,N,Ladies Apparel,,,N,NONE,8,N
WCOAT,INPUT,NOMINAL,,,,,.,.,N,,,.,N,Ladies Coats,,,N,NONE,8,N
;

run;

*------------------------------------------------------------*;
* Table Metadata;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_TM;
length MEMNAME $ 32
MEMTYPE $ 8
MEMLABEL $ 40
TYPEMEM $ 8
ENGINE $ 8
CRDATE 8
MODATE 8
NOBS 8
NCOLS 8
ROLE $ 20
USEEXTERNALDATA $ 1
SAMPLINGRATE 8
SEGMENT $ 20
libname $ 8
;

infile cards dsd;
input MEMNAME $
MEMTYPE $
MEMLABEL $
TYPEMEM $
ENGINE $
CRDATE
MODATE
NOBS
NCOLS
ROLE $
USEEXTERNALDATA $
SAMPLINGRATE
```

```
SEGMENT $
libname $
;

format CRDATE DATETIME16.
MODATE DATETIME16.
;

informat MEMNAME $CHAR32.
MEMTYPE $CHAR8.
MEMLABEL $CHAR40.
TYPEMEM $CHAR8.
ENGINE $CHAR8.
ROLE $CHAR20.
USEEXTERNALDATA $CHAR1.
SEGMENT $CHAR20.
libname $CHAR8.
;

cards;
DMEXA1,DATA,,DATA,V9,1267003746.567,1267003746.567,1966,50,TRAIN,,.,,SAMPSIO
;

run;

*------------------------------------------------------------*;
* Target Profile;
*------------------------------------------------------------*;

data WORK.DMEXA1TABLE_TP;
length ProfileID 8
ProfileName $ 64
Target $ 64
Level $ 16
Use $ 8
;

infile cards dsd;
input ProfileID
ProfileName $
Target $
Level $
Use $
;

informat ProfileName $CHAR64.
Target $CHAR64.
Level $CHAR16.
Use $CHAR8.
;

cards;
0,Profile 0,PURCHASE,,Y
;

run;
```

```
data WORK.DMEXA1TABLE_M0;
length _TYPE_ $ 32
VARIABLE $ 32
LABEL $ 40
LEVEL $ 32
EVENT $ 32
ORDER $ 10
FORMAT $ 32
TYPE $ 1
COST $ 32
USE $ 1
;

infile cards dsd;
input _TYPE_ $
VARIABLE $
LABEL $
LEVEL $
EVENT $
ORDER $
FORMAT $
TYPE $
COST $
USE $
;

label _TYPE_="Type"
VARIABLE="Variable"
LABEL="Label"
LEVEL="Measurement Level"
EVENT="Target Event"
ORDER="Order"
FORMAT="Format"
TYPE="Type"
COST="Cost"
USE="Use"
;

informat _TYPE_ $CHAR32.
VARIABLE $CHAR32.
LABEL $CHAR40.
LEVEL $CHAR32.
EVENT $CHAR32.
ORDER $CHAR10.
FORMAT $CHAR32.
TYPE $CHAR1.
COST $CHAR32.
USE $CHAR1.
;

cards;
MATRIX,,,PROFIT,,,,,,Y
TARGET,PURCHASE,Purchase (y/n),BINARY,YES,,YESNO.,N,,
DECISION,DECISION1,YES,,,,,N,1.5,Y
DECISION,DECISION2,NO,,,,,N,2.0,Y
```

```
DATAPRIOR,DATAPRIOR,Data Prior,,,,,N,,Y
TRAINPRIOR,TRAINPRIOR,Training Prior,,,,,N,,N
DECPRIOR,DECPRIOR,Decision Prior,,,,,N,,N
PREDICTED,P_PURCHASEYes,Predicted: PURCHASE=Yes,YES,,,,N,,
RESIDUAL,R_PURCHASEYes,Residual: PURCHASE=Yes,YES,,,,N,,
PREDICTED,P_PURCHASENo,Predicted: PURCHASE=No,NO,,,,N,,
RESIDUAL,R_PURCHASENo,Residual: PURCHASE=No,NO,,,,N,,
FROM,F_PURCHASE,From: PURCHASE,,,,,C,,
INTO,I_PURCHASE,Into: PURCHASE,,,,,C,,
FREQ,FREQUENT,Order Frequency,INTERVAL,,,,N,,
;

run;

data WORK.DMEXA1TABLE_D0;
length PURCHASE $ 32
COUNT 8
DATAPRIOR 8
TRAINPRIOR 8
DECPRIOR 8
DECISION1 8
DECISION2 8
;

infile cards dsd;
input PURCHASE $
COUNT
DATAPRIOR
TRAINPRIOR
DECPRIOR
DECISION1
DECISION2
;

label COUNT="Level Counts"
DATAPRIOR="Data Proportions"
TRAINPRIOR="Training Proportions"
DECPRIOR="Decision Priors"
DECISION1="YES"
DECISION2="NO"
;

informat PURCHASE $CHAR32.
;

cards;
YES,3719.68,0.61607667833227,0.61607667833227,0,1,-1
NO,2318.00999999999,0.38392332166772,0.38392332166772,0,0,1
;

run;

*------------------------------------------------------------*;
* Create Datasource data set;
*------------------------------------------------------------*;
```

```
data datasources;
length id $30 key $40 value $64;

id="DMEXA1TABLE"; key="Properties";    value="WORK.DMEXA1TABLE_P";             output;
id="DMEXA1TABLE"; key="ColumnMeta";    value="WORK.DMEXA1TABLE_CM";            output;
id="DMEXA1TABLE"; key="TableMeta";     value="WORK.DMEXA1TABLE_TM";            output;
id="DMEXA1TABLE"; key="TargetProfile"; value="WORK.DMEXA1TABLE_TP";            output;
id="DMEXA1TABLE"; key="TargetProfile_DM_PURCHASE"; value="WORK.DMEXA1TABLE_M0"; output;
id="DMEXA1TABLE"; key="TargetProfile_DD_PURCHASE"; value="WORK.DMEXA1TABLE_D0"; output;

*------------------------------------------------------------*;
* Variable Attributes for Ids;
*------------------------------------------------------------*;

data WORK.Ids_VariableAttribute;
length Variable $64 AttributeName $32 AttributeValue $64;

Variable="FREQUENT"; AttributeName="ROLE"; AttributeValue="REJECTED"; Output;

run;

*------------------------------------------------------------*;
* Decmeta Data Set for Ids;
*------------------------------------------------------------*;

data WORK.Ids_PURCHASE_DM;
length _TYPE_ $ 32
VARIABLE $ 32
LABEL $ 40
LEVEL $ 32
EVENT $ 32
ORDER $ 10
FORMAT $ 32
TYPE $ 1
COST $ 32
USE $ 1
;

infile cards dsd;
input _TYPE_ $
VARIABLE $
LABEL $
LEVEL $
EVENT $
ORDER $
FORMAT $
TYPE $
COST $
USE $
;

label _TYPE_="Type"
VARIABLE="Variable"
LABEL="Label"
LEVEL="Measurement Level"
EVENT="Target Event"
```

```
ORDER="Order"
FORMAT="Format"
TYPE="Type"
COST="Cost"
USE="Use"
;

informat _TYPE_ $CHAR32.
VARIABLE $CHAR32.
LABEL $CHAR40.
LEVEL $CHAR32.
EVENT $CHAR32.
ORDER $CHAR10.
FORMAT $CHAR32.
TYPE $CHAR1.
COST $CHAR32.
USE $CHAR1.
;

cards;
MATRIX,,,PROFIT,,,,,,Y
TARGET,PURCHASE,Purchase (y/n),BINARY,YES,,YESNO.,N,,
DECISION,DECISION1,YES,,,,,N,1.5,Y
DECISION,DECISION2,NO,,,,,N,2.0,Y
DATAPRIOR,DATAPRIOR,Data Prior,,,,,N,,Y
TRAINPRIOR,TRAINPRIOR,Training Prior,,,,,N,,N
DECPRIOR,DECPRIOR,Decision Prior,,,,,N,,N
PREDICTED,P_PURCHASEYes,Predicted: PURCHASE=Yes,YES,,,,N,,
RESIDUAL,R_PURCHASEYes,Residual: PURCHASE=Yes,YES,,,,N,,
PREDICTED,P_PURCHASENo,Predicted: PURCHASE=No,NO,,,,N,,
RESIDUAL,R_PURCHASENo,Residual: PURCHASE=No,NO,,,,N,,
FROM,F_PURCHASE,From: PURCHASE,,,,,C,,
INTO,I_PURCHASE,Into: PURCHASE,,,,,C,,
FREQ,FREQUENT,Order Frequency,INTERVAL,,,,N,,
MODELDECISION,D_PURCHASE,Decision: PURCHASE,INTERVAL,,,,N,,
EXPECTEDPROFIT,EP_PURCHASE,Expected Profit: PURCHASE,INTERVAL,,,,N,,
COMPUTEDPROFIT,CP_PURCHASE,Computed Profit: PURCHASE,INTERVAL,,,,N,,
BESTPROFIT,BP_PURCHASE,Best Profit: PURCHASE,INTERVAL,,,,N,,
INVESTMENTCOST,IC_PURCHASE,Investment Cost: PURCHASE,INTERVAL,,,,N,,
ROI,ROI_PURCHASE,Return on Investment: PURCHASE,INTERVAL,,,,N,,
;

run;

*------------------------------------------------------------*;
* Decdata Data Set for Ids;
*------------------------------------------------------------*;

data WORK.Ids_PURCHASE_DD;
length PURCHASE $ 32
COUNT 8
DATAPRIOR 8
TRAINPRIOR 8
DECPRIOR 8
DECISION1 8
DECISION2 8
```

```
;

infile cards dsd;
input PURCHASE $
COUNT
DATAPRIOR
TRAINPRIOR
DECPRIOR
DECISION1
DECISION2
;

label COUNT="Level Counts"
DATAPRIOR="Data Proportions"
TRAINPRIOR="Training Proportions"
DECPRIOR="Decision Priors"
DECISION1="YES"
DECISION2="NO"
;

informat PURCHASE $CHAR32.
;

cards;
YES,3719.68,0.61607667833227,0.61607667833227,0,1,-1
NO,2318.00999999999,0.38392332166772,0.38392332166772,0,0,1
;

run;

*------------------------------------------------------------*;
* Create node properties data set;
*------------------------------------------------------------*;

data nodeprops;
length id $12 property $32 value $64;

id= "MdlComp"; property="NumberOfReportedLevels";      value= "1E-6";          output;
id= "MdlComp"; property="NormalizeReportingVariables"; value= "Y";            output;
id= "MdlComp"; property="DecileBin";                   value= "20";           output;
id= "MdlComp"; property="LiftEpsilon";                 value= "1E-6";         output;
id= "MdlComp"; property="ProfitEpsilon";               value= "1E-6";         output;
id= "MdlComp"; property="RoiEpsilon";                  value= "1E-6";         output;
id= "MdlComp"; property="ScoreDistBin";                value= "20";           output;
id= "MdlComp"; property="RocChart";                    value= "Y";            output;
id= "MdlComp"; property="RocEpsilon";                  value= "0.01";         output;
id= "MdlComp"; property="AssessAllTargetLevels";       value= "N";            output;
id= "MdlComp"; property="SelectionStatistic";          value= "_TMISC_";      output;
id= "MdlComp"; property="Component";                   value= "ModelCompare"; output;
id= "MdlComp"; property="StatisticUsed";               value= "_TMISC_";      output;
id= "Neural";  property="NetworkArchitecture";         value= "GLIM";         output;
id= "Neural";  property="DirectConnection";            value= "N";            output;
id= "Neural";  property="Hidden";                      value= "3";            output;
id= "Neural";  property="Prelim";                      value= "N";            output;
id= "Neural";  property="PreliminaryRuns";             value= "5";            output;
id= "Neural";  property="PrelimMaxiter";               value= "10";           output;
```

```
id= "Neural";  property="PrelimMaxTime";               value= "1 HOUR";       output;
id= "Neural";  property="Maxiter";                     value= "20";           output;
id= "Neural";  property="Maxtime";                     value= "4 HOURS";      output;
id= "Neural";  property="TrainingTechnique";           value= "DEFAULT";      output;
id= "Neural";  property="ConvDefaults";                value= "Y";            output;
id= "Neural";  property="AbsConvValue";                value= "-1.34078E154"; output;
id= "Neural";  property="AbsFValue";                   value= "0";            output;
id= "Neural";  property="AbsFTime";                    value= "1";            output;
id= "Neural";  property="AbsGValue";                   value= "0.00001";      output;
id= "Neural";  property="AbsGTime";                    value= "1";            output;
id= "Neural";  property="AbsXValue";                   value= "1E-8";         output;
id= "Neural";  property="AbsXTime";                    value= "1";            output;
id= "Neural";  property="FConvValue";                  value= "0";            output;
id= "Neural";  property="FConvTime";                   value= "1";            output;
id= "Neural";  property="GConvValue";                  value= "1E-6";         output;
id= "Neural";  property="GConvTime";                   value= "1";            output;
id= "Neural";  property="ModelSelectionCriterion";     value= "PROFIT/LOSS";  output;
id= "Neural";  property="SuppressOutput";              value= "N";            output;
id= "Neural";  property="Residuals";                   value= "Y";            output;
id= "Neural";  property="Standardizations";            value= "N";            output;
id= "Neural";  property="HiddenUnits";                 value= "N";            output;
id= "Neural";  property="TrainCode";                   value= "";             output;
id= "Neural";  property="PrelimOutest";                value= "";             output;
id= "Neural";  property="Outest";                      value= "";             output;
id= "Neural";  property="Outfit";                      value= "";             output;
id= "Neural";  property="InitialDs";                   value= "";             output;
id= "Neural";  property="CodefileRes";                 value= "";             output;
id= "Neural";  property="CodefileNoRes";               value= "";             output;
id= "Neural";  property="Component";                   value= "NeuralNetwork"; output;
id= "Tree";    property="TrainMode";                   value= "BATCH";        output;
id= "Tree";    property="Criterion";                   value= "GINI";         output;
id= "Tree";    property="SigLevel";                    value= "0.2";          output;
id= "Tree";    property="Splitsize";                   value= ".";            output;
id= "Tree";    property="LeafSize";                    value= "5";            output;
id= "Tree";    property="MinCatSize";                  value= "5";            output;
id= "Tree";    property="Maxbranch";                   value= "2";            output;
id= "Tree";    property="Maxdepth";                    value= "6";            output;
id= "Tree";    property="Nrules";                      value= "5";            output;
id= "Tree";    property="Nsurrs";                      value= "0";            output;
id= "Tree";    property="MissingValue";                value= "USEINSEARCH";  output;
id= "Tree";    property="UseVarOnce";                  value= "N";            output;
id= "Tree";    property="Subtree";                     value= "ASSESSMENT";   output;
id= "Tree";    property="NSubtree";                    value= "1";            output;
id= "Tree";    property="AssessMeasure";               value= "PROFIT/LOSS";  output;
id= "Tree";    property="AssessPercentage";            value= "0.25";         output;
id= "Tree";    property="NodeSample";                  value= "5000";         output;
id= "Tree";    property="Exhaustive";                  value= "5000";         output;
id= "Tree";    property="UseDecision";                 value= "N";            output;
id= "Tree";    property="UsePriors";                   value= "N";            output;
id= "Tree";    property="Kass";                        value= "Y";            output;
id= "Tree";    property="KassApply";                   value= "BEFORE";       output;
id= "Tree";    property="Depth";                       value= "Y";            output;
id= "Tree";    property="Inputs";                      value= "N";            output;
id= "Tree";    property="NumInputs";                   value= "1";            output;
id= "Tree";    property="VarSelection";                value= "Y";            output;
id= "Tree";    property="Dummy";                       value= "N";            output;
```

```
id= "Tree";    property="Leafid";                      value= "Y";            output;
id= "Tree";    property="Predict";                     value= "Y";            output;
id= "Tree";    property="NodeRole";                    value= "SEGMENT";      output;
id= "Tree";    property="Component";                   value= "DecisionTree"; output;
id= "Reg";     property="MainEffect";                  value= "Y";            output;
id= "Reg";     property="TwoFactor";                   value= "N";            output;
id= "Reg";     property="Polynomial";                  value= "N";            output;
id= "Reg";     property="PolynomialDegree";            value= "2";            output;
id= "Reg";     property="Terms";                       value= "N";            output;
id= "Reg";     property="Error";                       value= "LOGISTIC";     output;
id= "Reg";     property="LinkFunction";                value= "LOGIT";        output;
id= "Reg";     property="SuppressIntercept";           value= "N";            output;
id= "Reg";     property="InputCoding";                 value= "DEVIATION";    output;
id= "Reg";     property="MinResourceUse";              value= "D";            output;
id= "Reg";     property="ModelSelection";              value= "STEPWISE";     output;
id= "Reg";     property="SelectionCriterion";          value= "DEFAULT";      output;
id= "Reg";     property="SelectionDefault";            value= "Y";            output;
id= "Reg";     property="Sequential";                  value= "N";            output;
id= "Reg";     property="SlEntry";                     value= "0.05";         output;
id= "Reg";     property="SlStay";                      value= "0.05";         output;
id= "Reg";     property="Start";                       value= "0";            output;
id= "Reg";     property="Stop";                        value= "0";            output;
id= "Reg";     property="Force";                       value= "0";            output;
id= "Reg";     property="Hierarchy";                   value= "CLASS";        output;
id= "Reg";     property="Rule";                        value= "NONE";         output;
id= "Reg";     property="MaxStep";                     value= ".";            output;
id= "Reg";     property="StepOutput";                  value= "N";            output;
id= "Reg";     property="OptimizationTechnique";       value= "DEFAULT";      output;
id= "Reg";     property="ModelDefaults";               value= "Y";            output;
id= "Reg";     property="MaxIterations";               value= ".";            output;
id= "Reg";     property="MaxFunctionCalls";            value= ".";            output;
id= "Reg";     property="MaxTime";                     value= "604800";       output;
id= "Reg";     property="ConvDefaults";                value= "Y";            output;
id= "Reg";     property="AbsConvValue";                value= "-1.34078E154"; output;
id= "Reg";     property="AbsFValue";                   value= "0";            output;
id= "Reg";     property="AbsFTime";                    value= "1";            output;
id= "Reg";     property="AbsGValue";                   value= "0.00001";      output;
id= "Reg";     property="AbsGTime";                    value= "1";            output;
id= "Reg";     property="AbsXValue";                   value= "1E-8";         output;
id= "Reg";     property="AbsXTime";                    value= "1";            output;
id= "Reg";     property="FConvValue";                  value= "0";            output;
id= "Reg";     property="FConvTimes";                  value= "1";            output;
id= "Reg";     property="GConvValue";                  value= "1E-6";         output;
id= "Reg";     property="GConvTimes";                  value= "1";            output;
id= "Reg";     property="UseInEst";                    value= "N";            output;
id= "Reg";     property="InEstDataset";                value= "";             output;
id= "Reg";     property="ClParm";                      value= "N";            output;
id= "Reg";     property="CovB";                        value= "N";            output;
id= "Reg";     property="CorB";                        value= "N";            output;
id= "Reg";     property="Simple";                      value= "N";            output;
id= "Reg";     property="SuppressOutput";              value= "N";            output;
id= "Reg";     property="Details";                     value= "N";            output;
id= "Reg";     property="PrintDesignMatrix";           value= "N";            output;
id= "Reg";     property="SASSPDS";                     value= "N";            output;
id= "Reg";     property="Performance";                 value= "N";            output;
id= "Reg";     property="Component";                   value= "Regression";   output;
```

```
id= "Part";    property="Method";                      value= "DEFAULT";      output;
id= "Part";    property="TrainPct";                    value= "40";           output;
id= "Part";    property="ValidatePct";                 value= "30";           output;
id= "Part";    property="TestPct";                     value= "30";           output;
id= "Part";    property="RandomSeed";                  value= "12345";        output;
id= "Part";    property="Component";                   value= "Partition";    output;
id= "Ids";     property="DataSource";                  value= "DMEXA1TABLE";  output;
id= "Ids";     property="Scope";                       value= "LOCAL";        output;
id= "Ids";     property="Role";                        value= "TRAIN";        output;
id= "Ids";     property="Library";                     value= "SAMPSIO";      output;
id= "Ids";     property="Table";                       value= "DMEXA1";       output;
id= "Ids";     property="NCols";                       value= "50";           output;
id= "Ids";     property="NObs";                        value= "1966";         output;
id= "Ids";     property="SamplingRate";                value= ".";            output;
id= "Ids";     property="Segment";                     value= "";             output;
id= "Ids";     property="UseExternal";                 value= "";             output;
id= "Ids";     property="OutputType";                  value= "VIEW";         output;
id= "Ids";     property="ForceRun";                    value= "";             output;
id= "Ids";     property="Component";                   value= "DataSource";   output;
id= "Ids";     property="Description";                 value= "";             output;
id= "Ids";     property="UseExternalData";             value= "";             output;
id= "Ids"; property="EM_VARIABLEATTRIBUTES"; value= "WORK.Ids_VariableAttribute"; output
id= "Ids";   property="EM_DECMETA_PURCHASE";   value= "WORK.Ids_PURCHASE_DM"; output;
id= "Ids";  p roperty="EM_DECDATA_PURCHASE";   value= "WORK.Ids_PURCHASE_DD"; output;

run;

*------------------------------------------------------------*;
* Create connections data set;
*------------------------------------------------------------*;

data connect;
length from to $12;

input from to;

cards;
   Neural MdlComp
   Tree MdlComp
   Reg MdlComp
   Part Reg
   Part Tree
   Part Neural
   Ids Part
;

run;
*------------------------------------------------------------*;
* Create actions to run data set;
*------------------------------------------------------------*;

%macro emaction;
%let actionstring = %upcase(&EM_ACTION);
%if %index(&actionstring, RUN) or %index(&actionstring, REPORT) %then
%do;
```

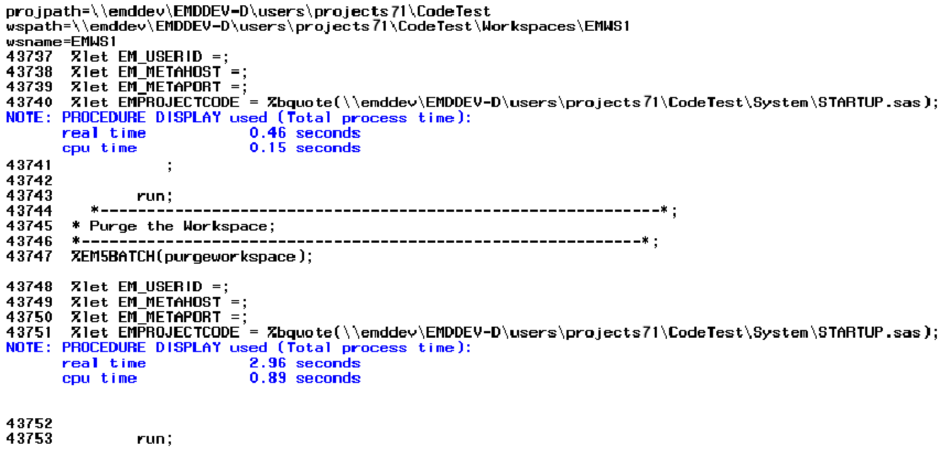
```
data actions;
length id $12 action $40; id="MdlComp";

   %if %index(&actionstring, RUN) %then %do;
   action='run'; output;
   %end;

   %if %index(&actionstring, REPORT) %then %do;
   action='report'; output;
   %end;

run;
%end;
%mend;
%emaction;
*------------------------------------------------------------*;
* Execute the actions;
*------------------------------------------------------------*;

%EM5BATCH(execute,
          workspace=workspace,
          nodes=nodes,
          connect=connect,
          datasources=datasources,
          nodeprops=nodeprops,
          action=actions)
          ;

       run;


*------------------------------------------------------------*;
* Purge the Workspace;
*------------------------------------------------------------*;

%EM5BATCH(purgeworkspace);
run;
```

Submit the Example 7B Batch Processing Code to SAS:

Modify your saved Example7.sas batch processing code or cut and paste the code above into your SAS Program Editor window. If you did not edit your saved batch file, remember to change your project path in the code above before you submit it. The modified batch processing code should create a new workspace EMWS2 in the CodeTest project, rerun the flow with the changed modeling properties, generate a second report, and purge the EMWS2 workspace, leaving the CodeTest project empty except for the two entries in the Reports folder.

Verify Example 7B Changes in SAS Log

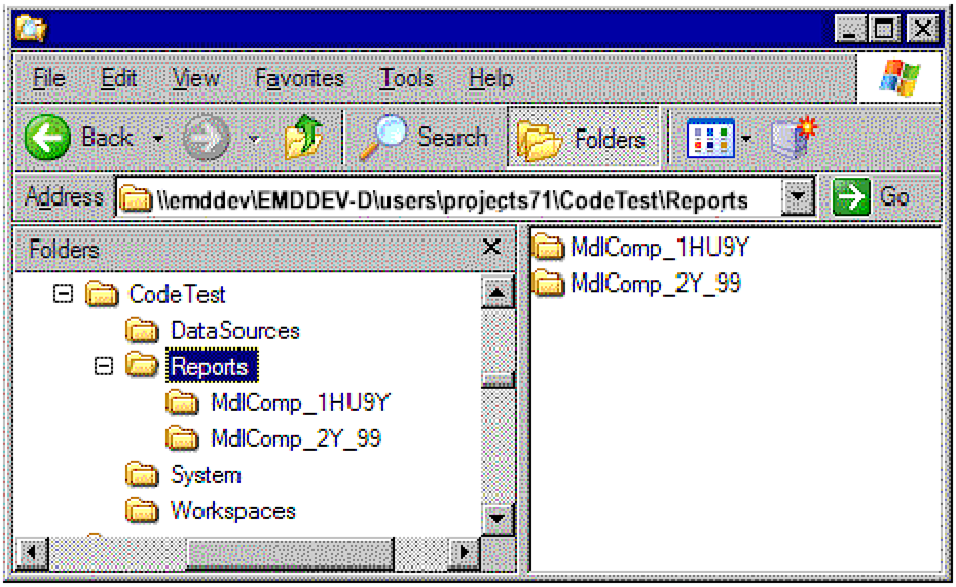
Examine the SAS log after you run the batch processing code to verify that the new workspace was created and purged.



```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTest
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTest\Workspaces\EMWS1
wsname=EMWS1
43737   %let EM_USERID =;
43738   %let EM_METAHOST =;
43739   %let EM_METAPORT =;
43740   %let EMPROJECTCODE = %bquote(\\emddev\EMDDEV-D\users\projects71\CodeTest\System\STARTUP.sas);
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           0.46 seconds
      cpu time            0.15 seconds

43741             ;
43742
43743          run;
43744     *-----------------------------------------------------------*;
43745   * Purge the Workspace;
43746   *-----------------------------------------------------------*;
43747   %EM5BATCH(purgeworkspace);

43748   %let EM_USERID =;
43749   %let EM_METAHOST =;
43750   %let EM_METAPORT =;
43751   %let EMPROJECTCODE = %bquote(\\emddev\EMDDEV-D\users\projects71\CodeTest\System\STARTUP.sas);
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           2.96 seconds
      cpu time            0.89 seconds


43752
43753          run;
```

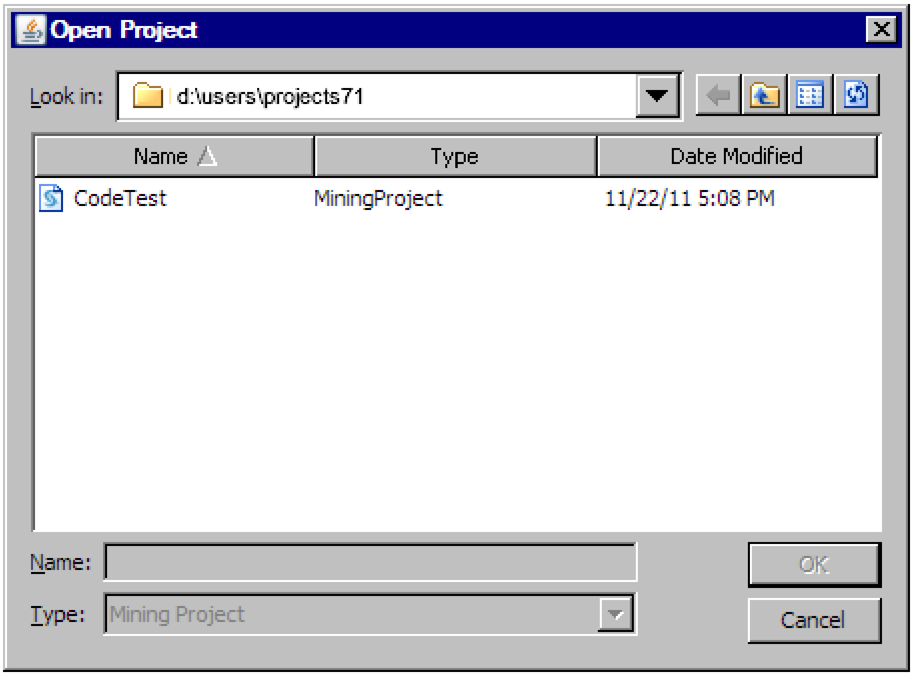
Verify Example 7B Changes Using a File Utility:

The file utility shows that the CodeTest project is still empty, with no remaining workspaces and with two Model Comparison reports.



Verify Example 7B Changes Using the SAS Enterprise Miner GUI:

Start the SAS Enterprise Miner GUI to view the CodeTest project and its contents. When the Welcome to SAS Enterprise Miner window opens, select **Open Project** and you should be able to navigate to your project location and find your saved project file. Select it and click **OK**.



Look in: d:\users\projects71

Name	Type	Date Modified
CodeTest	MiningProject	11/22/11 5:08 PM

Name:

Type: Mining Project

When the CodeTest project that you created using batch processing code opens in the GUI, you can verify that there are already two Model Comparison reports in the Model Packages folder and that DMEXA1TABLE is found in the Data Sources folder. The rest of the project is empty.

You can use the reports to re-create precise process flow diagram equivalents of the code that created the reports. If you open the report packages, you can view or print the process flow diagrams, or you can view XML files that contain model package summaries, workspace summaries, and batch processing code summaries. You can examine the wide variety of scoring reports as well.

By using the CodeTest project that was created in Example 7B, you can now reconstruct and create batch processing code for the two process flow diagrams in the reports, modify either of the reconstructed process flow diagrams in a new workspace, add a new process flow diagram to the project, or simply continue to accumulate reports from batch processing code process flows that are part of chron jobs.

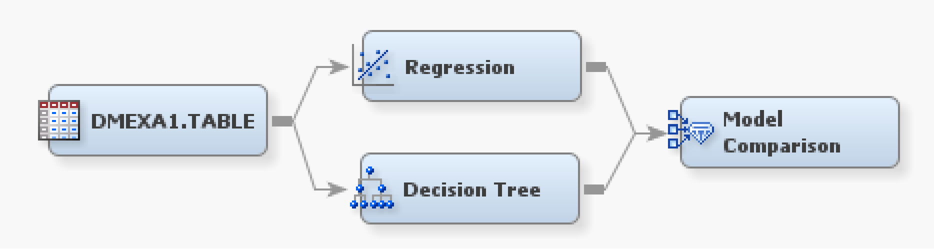
Example 8: Decrease Run Time by Executing Nodes Simultaneously

SAS Enterprise Miner batch processing can use a SAS feature known as MP CONNECT from the SAS/CONNECT product to run two or more nodes in the same work flow simultaneously. Example 8 will explain when this type of multitasking is appropriate and how to modify the batch code to allow nodes to run concurrently.

Multitasking within a SAS Enterprise Miner workflow can be used when two or more nodes originate from the same preceding node and their tasks have no interdependencies. Because the Decision Tree node and the Regression node in the previous example do not require information from one another to function properly, the workflow can be optimized with MP CONNECT, meaning that both nodes can be executed simultaneously.

Example 8 Process Flow Diagram Overview:

The diagram in this example is identical to the diagram created in the Example 6 on page 1386 . However, in Example 8, the batch processing code is changed and the Decision Tree node and the Regression node execute simultaneously.



Example 8 Node Properties Overview:

The nodes in the process flow diagram all use the default settings.

Example 8 Output Location:

You can find the output from Example 8 by reviewing the SAS log. Near the end of the log, look for lines that resemble the following:

```
projpath=\\emddev\EMDDEV-D\users\projects71\CodeTutor
wspath=\\emddev\EMDDEV-D\users\projects71\CodeTutor\Workspaces\EMWS1
wsname=EMWS1
```

Example 8 Batch Processing Code:

When two or more nodes in a workflow meet the requirements for MP CONNECT multiprocessing, a simple change must be made to the value of EM_NUMTASKS before the batch processing code will run the nodes concurrently. In the previous examples, the value of EM_NUMTASKS has been SINGLE. A value of SINGLE or 0 results in nodes executing sequentially. Change the value of EM_NUMTASKS to HARDWARE to base the number of separate threads launched for node execution on the number of processors on the server. Set the value to MAX to launch as many threads as there are parallel nodes on a diagram. Specify 1, 2, or 4 to enable node execution in 1, 2, or 4 additional parallel threads, respectively. In all cases, one thread is used to run control functions.

Because Example 8 modifies the batch code from the Example 6, only the macro variable definition code is provided below. All further code is identical to that from Example 6.

```
*------------------------------------------------------------*;
* ***SAS Enterprise Miner 12.3 Batch Processing Example 8***  ;
* Use %let statements to hold values for project location     ;
* and project name, the workspace name, the name and location ;
* of the summary data set, specify the Merge batch edit mode, ;
* and execute the process flow diagram in a several threads   ;
* -----------------------------------------------------------*;

%let EM_PROJECT = \\emddev\EMDDEV-D\users\projects71;
%let EM_PROJECTNAME = CodeTutor;
%let EM_WSNAME = EMWS1;
%let EM_EDITMODE = M;
%let EM_SUMMARY = WORK.SUMMARY;
%let EM_NUMTASKS = HARDWARE;
%let EM_DEBUG = ;
%let EM_ACTION = RUN
```

Chapter 86

SAS Enterprise Miner Batch Processing Appendix

SAS Enterprise Miner Batch Processing Appendix

Batch Processing Definition Data Set Format Tables

The tables below provide the required variable formatting information that is required to create batch processing definition data sets using SAS DATA step programming.

- Table 86.1 on page 1429
- Table 86.2 on page 1430
- Table 86.3 on page 1430
- Table 86.4 on page 1430
- Table 86.5 on page 1430

Table 86.1 *Workspace Data Set Formatting*

Variable	Description	Type	Length
Property	Name of the workspace configuration property	C	32
Value	Value to assign to the property	C	64

Table 86.2 *Nodes Data Set Formatting*

Variable	Description	Type	Length
ID	Unique ID assigned to the node, such as $1.	C	12
Component	Component name of the tool in the Components Table, such as DataSource	C	32
Description	Description of the node, such as SAMPSIO Asssociation Data.	C	64

Table 86.3 *Action Data Set Formatting*

Variable	Description	Type	Length
ID	Unique ID assigned to the node, such as $1.	C	12
Action	Name of the action to perform, such as run, update, or report.	C	12

Table 86.4 *Connections Data Set Formatting*

Variable	Description	Type	Length
From	Unique ID assigned to the predecessor node, such as $1.	C	12
To	Unique ID assigned to the successor node, such as $2.	C	12

Table 86.5 *Node Properties Data Set Formatting*

Variable	Description	Type	Length
ID	Unique ID assigned to the node, such as $1.	C	12

Variable	Description	Type	Length
Property Name	Name of the node execution property, such as PORT_DATA_ DATA	C	32
Property Value	Value to assign to the property, such as SAMPSIO.ASSOCS	C	64

Batch Processing Component Names

You must manually create the nodes data set that defines the nodes you use in your batch process flow diagram.

The node definitions list one node per statement, in the form

```
id component Description=
```

where

ID	A unique ID you choose for each node you use. Each ID begins with a dollar sign. For example, $1
component	The component name of the node. See the section below for node component names.
description	Your description of the tool. For example, SAMPSIO Association Data

An example of a simple node definition:

```
data work.properties;
input id $12 property $32 value $34;
cards;
  $1 DataSource Description=IDS using SAMPSIO.ASSOCS
  $2 Assocation Description=SAMPSIO Association Data
;
run;
```

Batch Processing Component Names by Node Type

The SAS Enterprise Miner user interface provides a list of batch processing component names organized by node type. To open a complete table of SAS Enterprise Miner batch processing component names by node type, from the Enterprise Miner main menu, select **Help ⇨ Component Properties**.

Note: Credit Scoring for SAS Enterprise Miner and SAS Text Miner are not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner or SAS Text Miner, the tools will not be available in your user interface.

Chapter 87
SAS Enterprise Miner Macro Reference

SAS Enterprise Miner Macro Variables Reference

SAS Enterprise Miner Macro Variables

System Macro Variables

- **EM_DEBUG** — controls information that is sent to the log. Use this feature only in conjunction with SAS Technical Support.

  Valid values are as follows:

  - **LOG** sends the node log to the SAS Log window.
  - **METHOD** indicates to print the classes and methods that are called by the application in the log.
  - **OUTPUT** sends the node output to the SAS Output window.
  - **SOURCE** indicates to print additional information (for example, scoring steps of each node in the log).
  - **TESTSINGLEMODE** disables the use of parallel processing and runs each node sequentially.
- **EM_DISPLAY_UDF** — This macro variable specifies whether any user-defined formats that exist in the format search path are downloaded when a project is opened. Subsequently, user-defined formats are used to display data for tables that reference these formats. When tables do not reference any user-defined formats, only the raw data values are displayed. Valid values are YES or NO, and NO is the default value.
- **EM_EXPLOREOBS_DEFAULT** — specifies the default number of observations that are downloadable for visualization. The default setting for EM_EXPLOREOBS_DEFAULT is based on the record length.
- **EM_EXPLOREOBS_MAX** — controls the number of rows of data downloaded to the client for interactive graphics in the explore data actions. The value should be

large enough to select a sample that accurately represents the population but not so large to overload your network and computer memory. The default value is dynamically determined by the record length of your selected data set. Set this value when you think the default value is not appropriate.

- **EM_NUMTASK** — controls the number of nodes that will be run concurrently. Use this to control the compute load processed by your system. Possible values are nonnegative integers, SINGLE, MAX or MAXIMUM, and DEFAULT.

  If you specify 0, 1, or SINGLE, then the nodes in a process flow diagram will run sequentially. That is, parallel processing is not used.

  When you specify MAX or MAXIMUM, SAS/CONNECT is used and SAS Enterprise Miner attempts to run as many nodes as is possible in parallel. For example, if you have 4 nodes that can be run in parallel, the system with attempt to run all four nodes concurrently.

  If you specify DEFAULT, parallel processing is used to run nodes concurrently. If the SAS grid is not used, the maximum number of concurrent nodes is determined by the value of the automatic system macro variable SYSNCPU. If the SAS grid is used, the maximum number of concurrent nodes is the value returned by the GRDSVC_NNODES function. This function reports the total number of job slots that are available for use on the SAS grid.

- **EM_PATH** — returns the ID of the predecessor node in a process flow diagram.

  The macro parameters are as follows:

  - **NODEID** — specifies the ID of the current node.
  - **OUTDS** — specifies the output data set.
  - **NODES** — specifies which node IDs to retrieve.

    Valid values are as follows:

    - ALL — returns all nodes in a process flow diagram.
    - PREDECESSOR — returns the immediate predecessor node.
    - SCORE — returns the nodes in the scoring path.
    - PATH — returns the nodes in the training path.

- **EM_VIEW_BUFSIZE** — This macro variable sets the buffer size when processing a view. The default value is 64M.

- **EM_WIPTRACE** — This macro variable determines whether information passed to the Java middle-tier is written to the log. In general, this information is provided in list format. Valid values for this macro are YES and NO, and the default value is NO. This variable is available only in SAS Enterprise Miner 7.1 and later.

- **RPM_DEBUG** — This macro variable is used by the EM_RPM macro, and controls information sent to the log file. Use this feature only in conjunction with SAS Technical Support

  Valid values are as follows:

  - **LOG** sends the node log to the SAS Log window.
  - **METHOD** indicates to print the classes and methods that are called by the application in the log.
  - **OUTPUT** sends the node output to the SAS Output window.
  - **SOURCE** indicates to print additional information (for example, scoring steps of each node in the log).

- **_EM_Char_Left_Paren** — specifies the character that is used as the open parenthesis.
- **_EM_Char_Right_Paren** — specifies the character that is used as the open parenthesis.
- **_EM_Char_Colon** — specifies the character that is used as the open parenthesis.

Advisor Macro Variables

- **EM_TRAIN_MAXLEVELS** — specifies the maximum number of levels that are allowed when computing the fit statistics with PROC DMDB. The default value is 512.

General Modeling Macro Variables

- **EM_ASSESS** — controls the execution of model assessment functions for modelling nodes. Valid values are Y or N, and Y is the default value. Specify N to suppress all assessment reports.
- **EM_DECMETA_MAXLEVELS** — controls the maximum number of levels for any target variable for which a target profile can be built. The default value is 32.
- **EM_GROUPASSESS** — controls the execution of model assessment functions for model nodes within a group processing segment. The possible values are Y or N. The default value is Y. Use this setting to speed processing when you only want to see the model assessment details of the final population model, rather than the individual models.
- **EM_MAXGROUPASSESS** — controls the maximum number of groups for which predictive model assessment will be executed for a modeling node within a group processing segment. The default value is blank.
- **EM_PMML** — enables the generation of PMML score code by the nodes that support PMML. Valid values are Y, Yes, N, and No, and the default value is No.
- **EM_TRAIN_MAXLEVELS** — specifies the maximum number of levels that are permitted in a target variable that can have a target profile. The default value is 512.

Neural and Autoneural Macro Variables

- **EM_NEURAL_PERFORMANCE_DATA** — provides more control over performance within both the Neural Network and AutoNeural nodes based on hardware as well as the size of the problem being solved. This determines whether a data set or view is passed to PROC NEURAL, as well as the options that are passed to the performance statement within PROC NEURAL.

  Valid values are as follows:

  - NOUTIL — A view of the training data is processed and the corresponding performance statement will contain the options `ALLDETAILS NOUTILFILE`.
  - TEMP — A data set is used and the performance statement will contain the options `ALLDETAILS NOUTFILFILE`.
  - UTIL — A view of the training data is processed and the corresponding performance statement will contain the options `ALLDETAILS PAGESIZE=262144 COMPILE THREADS=YES`.
  - _blank_ — Functions the same as specifying TEMP.
- **EM_NEURAL_PERFORMANCE_STATEMENT** — controls whether the performance statement passed to PROC NEURAL in both the Neural Network and

AutoNeural nodes. If this is not initialized, the performance statement is built based on the value of the **EM_NEURAL_PERFORMANCE_DATA** macro variable.

Decision Tree Macro Variables

- **EM_ARBOR** — gives users the ability to pass additional arguments to the PROC ARBOR call within the Decision Tree node and other nodes that use PROC ARBOR via EMFUNC.TREE.CLASS
- **EM_INTERACTIVE_TREE_MAXOBS** — controls the maximum number of observations used for the interactive decision tree. The default value is dynamically controlled by the record length of your selected data set.
- **EM_INTERACTIVE_TREE_SAMPLEMETHOD** — controls the sampling method used to create the data for the interactive decision tree. This is used in combination with **EM_INTERACTIVE_TREE_MAXOBS**. Valid values are FIRSTN, RANDOM, and STRATIFY. The default value is RANDOM.

IGN Macro Variables

- **EM_IGN** — gives users the ability to pass additional arguments to the PROC ARBOR call within the IGN node.

Multiplot Macro Variables

- **EM_FONT** — sets the font used by the Multiplot node.

Reporter Macro Variables

- **EM_REPORT_FOOTNOTE1** — used to set the text of the first footnote.
- **EM_REPORT_FOOTNOTE2** — used to set the text of the second footnote.
- **EM_REPORT_GDEVICE** — used to set the graphics device.
- **EM_REPORT_HEADER_SIZE** — sets the size, in points, for report headers and data cells for tables printed by ODS. The default value is 6.
- **EM_REPORT_NODE_FONT** — sets the font that is used for nodes in a process flow diagram. The default value is SIMPLEX.
- **EM_REPORT_ODS_FONT** — the font used in the ODS template for all ODS output.
- **EM_REPORT_PRINT_MAXVARS** — defines the maximum number of variables to print. The default value is 40.
- **EM_REPORT_SUPPRESS_DATE** — enables the date to be suppressed from generated reports. Valid values are Y and N, and the default value is N.
- **EM_REPORT_TEXT_FONT** — specifies the font that is used for report headers. The default value is SWISS.
- **EM_REPORT_TEXT_SIZE** — specifies the font size, in points, for graph labels and axes. The default value is 10.
- **EM_REPORT_TITLE** — used to set the text of the title.
- **EM_REPORT_TITLE_SIZE** — sets the size, in points, for report titles. The default value is 16.
- **EM_SUMMARY_REPORT_MAXOBS** — specifies the sample size that is used to generate information such as variable rankings, scorecard data, and classification results. The default value is determined by the record length of the selected data set.

- **EM_SUMMARY_REPORT_SAMPLEMETHOD** — controls the sampling method that is used for variable rankings, scorecards, summary-style reports, and RPM results. This macro variable is used in conjunction with **EM_SUMMARY_REPORT_MAXOBS**. Valid values are FIRSTN, RANDOM, or STRATIFY, and the default value is RANDOM.

SAS Text Miner Macro Variables

General SAS Text Miner Macro Variables

- **TM_DEBUG** — specifies whether some intermediate data sets are deleted. Valid values are 0, 1, and ALL. When set to 0, or not set, some intermediate data sets are deleted. When set to 1, extra information is printed for the Text Rule Builder. When set to ALL, all of extra information is printed. The default value is 0.

Text Mining Node Macro Variables

Note: The following macro variables were deprecated in SAS Text Miner 5.1.

- **TM_MINDESCTERMS** — is the minimum number of times that a term must appear in a cluster to be considered a descriptive term. The value specified here must be an integer and the default value is 2.
- **TM_ROLLWEIGHT** — defines how terms are rolled up.

  The following values are possible:

  - 1 — roll-up terms are those that have the highest weight.
  - 2 — roll-up terms are those that have the highest value of weight multiplied by log(numdocs+1), where numdocs is the number of documents in which the term occurs.
  - 3 — roll-up terms are those that have the highest value of weight multiplied by sqrt(numdocs), where numdocs is the number of documents in which the term occurs. This is the default value.
  - 4 — roll-up terms are those that have the highest value of weight multiplied by numdocs.
- **TM_SVDDATA** — specifies to delete the SVD input matrix when set to 0 and to not delete the SVD input matrix when set to 1. The default value is 0.

Text Filter Node Macro Variables

Note: The following variables are available only in SAS Text Miner 4.1 and later.

- **TMM_DICTPEN** — is the penalty to apply to the SPEDIS() value if a dictionary data set is specified and a potential misspelling exists in the dictionary. The default value is 2.
- **TMM_MAXCHILD** — is the maximum number of documents in which a term can occur and also be considered a misspelling. The default value is log10(numdocs)+1.
- **TMM_MAXSPEDIS** — is the maximum SPEDIS() for a misspelling and its parent to have and also evaluate as a misspelling. The default value is 15.
- **TMM_MINPARENT** — is the minimum number of documents in which a term must occur to be considered a possible parent. The default value is log10(numdocs) +4.

- **TMM_MULTIPEN** — is the penalty to apply to the SPEDIS() value if one of the terms is a multi-word term. The default value is 2.

Text Rule Builder Node Macro Variables

- **TM_DEBUG** — specifies whether extra information is printed to the Text Rule Builder node's log. Specify 0 if you do not wish to print extra information, 1 to print extra information, or ALL to print all extra information. The default value is 0.
- **TMB_MAXTEXTLEN** — specifies the number of characters that must be common between two documents for those documents to be considered the same for active learning purposes. The first &TMB_MAXTEXTLEN characters of the parse variable are stored, and the first &TMB_MAXTEXTLEN characters for two documents are identical, user changes are applied. The default value is 200.

Text Topic Node Macro Variables

- **TMM_DOCCUTOFF** — is the document cutoff value for user topics and single-term topics in the Topic table. Higher values decrease the number of documents assigned to a topic. The default value is 0.001.
- **TMM_MAX_TOPIC_ANGLE** — is the maximum cosine allowed between two topics for them to be considered equivalent; a lower number eliminates fewer topics based on closeness to other topics; a higher number eliminates more topics based on closeness to other topics. For example, to change the maximum topic angle for exclusion of a topic to be 5 degrees, put "%let tmm_max_topic_angle=5;" in your project start code. A single-term topic or multi-term topic is excluded if its topic vector is within this many degrees of any lower number topic created. The default value is 3.

  Note: This macro variable was deprecated in SAS Text Miner 5.1.
- **TMM_NORM_PIVOT** — is a value between 0 and 1 that is used for the pivot normalization of document length. If you want longer documents to contain many more topics than short documents, set this value closer to 1. If you want short documents and long documents to contain about the same number of topics, set this value closer to 0. The default value is 0.5.
- **TMM_TERMCUTOFF** — is used to determine the default term cutoff for user topics and single-term topics in the Topic table. Higher values decrease the number of terms assigned to a topic. The default value of 0.001 means that any term in a user or single-term topic with a nonzero weight is used.
- **TMM_TERM_CUTOFF_PCT** — For any multi-term topic, the initial term cutoff is set to this value multiplied by the highest weighted term. For example, if the highest weighted term is 2, the default term cutoff for that multi-term topic will be 0.2. By default, for multi-term topics, any terms where the absolute value of the weight is lower than this number times the maximum weight are not included. The default value is 0.001.

Miscellaneous Macro Variables

The SCOREXML Macro Variables

- **_EMDEBUG** — specifies if the SCOREXML macro should delete work files. Specify 0 to delete work files and 1 to keep work files. The default value is 0.

- **_VARMETADATA** — specifies if the SCOREXML macro should delete the data sets that contain the metadata for the input and output variables. Specify 0 to delete this information and 1 to keep this information. The default value is 0.
- **_XMLVERSIONNUM** — used to set an alternate version number for the style or format of the XML generated. The default is 6.2. For example: `%let _xmlVersionNum = 9.9;` would yield a line in the XML output that resembles `<Version>9.9</Version>`.

EM_MigrateProject Macro Variables

- **_EM_PATHLEN** — determines the length of the buffers that are used for file paths. Valid values are integers between 256 and 32767, and the default value is 32767.
- **_EM_TRACE** — specifies if your debugging output will contain more messages and the options `NOTES SOURCE MPRINT MLOGIC`. Valid values are ON and OFF, and OFF is the default value.

SAS Enterprise Miner %EMDS Macro

Description of the %EMDS Macro

You can use the Enterprise Miner Data Source (%EMDS) macro to create data source definitions. You can create data source definitions in any directory, but creating them in the Enterprise Miner Global Data Set (EMGDS) location is a convenient way to make them available to Enterprise Miner.

You can use the %EMDS macro for the following:

- when you run external processes such as nightly data integration jobs or query applications
- when you run the %EM5BATCH macro and need to create temporary data source definitions

Syntax

%EMDS (data=*source-libname.source-data-set-name*,
  <option-1=*value*>,
  <option-n=*value*>
);

The %EMDS macro has the following options and settings:

Use This Option...	**To Specify...**	**Default Value**
cost=	cost variable	
crossid=	cross ID variable	
data=	input libref and data set	&syslast
id=	ID variable	
freq=	frequency variable	
sequence=	sequence variable	

Use This Option...	To Specify...	Default Value
rootLibrary=	target libref	EMGDS
target=	target variable	
timeid=	time ID variable	
name=	display name of the data source	
tablerole=	modeling role of the table	
userid=	user ID of the user who generates the data source	

Advisor Options

The %EMDS macro has the following advisor options and settings:

Note: Advisor options can be used only after setting option adviseMode=ADVANCED.

Use This Option	To Specify This	Default Value
adviseMode=	BASIC or ADVANCED advisor options	BASIC
applyIntervalLevelLowerLimit	whether to apply the IntervalLowerLimit option	Y
intervalLevelLowerLimit	lower limit for interval variables	20
applyMaxClassLevels	whether to apply the MaxClassLevels option	Y
maxClassLevels	maximum number of class levels before a variable is rejected	20
identifyEmptyColumns	whether to identify empty columns	Y
maxPercentMissing	maximum percent missing values allowed before a variable is rejected	50

Text Mining Options

The %EMDS macro has the following text mining options and settings:

Use This Option	To Specify This	Default Value
URL=	the URL= variable points to the Web location used to supply text. The URL role works in conjunction with the TEXTLOC role.	
TEXTLOC=	The TEXTLOC= variable contains the complete path to the text source file, relative to the server that Text Miner is running on.	

Information Only Options

The %EMDS macro has the following information-only options and settings:

Note: Information-only options are not used for modeling.

Use This Option	To Specify This	Default Value
segment=	segment variable	
samplerate=	sampling rate	
useexternal=	whether to use external data	

Usage

The following code is one example that uses the Enterprise Miner Data Source (%EMDS) macro:

```
filename code catalog "sashelp.emutil.em_loadutilmacros.source";
%INCLUDE code;
LIBNAME EMGDS "/projects/global_datasources";
%EMDS(data=mylib.mydata,target=sometarget);
```

Note: To view the data source in the Enterprise Miner user interface, you must specify a location for the Enterprise Miner Global Data Set (EMGDS) library. You should assign the EMGDS library in either your server start-up code or in your project start-up code before you run sample code similar to the above.

You can also create target profile definitions for a data source by using the Enterprise Miner %EMTP macro. For more information about the Enterprise Miner %EMTP macro, see SAS Enterprise Miner %EMTOP Target Profiler Macro on page 1441 .

SAS Enterprise Miner %EMTP Target Profiler Macro

Overview of the %EMTP Target Profiler Macro

Use the Enterprise Miner Target Profiler (%EMTP) macro to create target profile definitions for a data source. Use the %EMTP macro when you want to create models that are weighted by the values of decisions. When you run the %EM5BATCH macro, you can use the %EMTP macro to create target profiles.

Syntax

%EMTP (data=*source-libname.source-data-set-name*,

<option-1=*value*>,

<option-n=*value*>

);

The Enterprise Miner Target Profile (%EMTP) macro has the following options:

Use This Option	To Specify This.	Default Value
columnsmeta=	columnsmeta data set	
decdata=	name of the decision data set, if decision matrix is used	WORK.DECDATA
decmeta=	name of the decision metadata data set, if decision matrix is used	WORK.DECMETA
dectype=	type of decision (LOSS or PROFIT), if decision matrix is used	
numdec=	number of decisions	
target=	target variable name	

Usage Example

The following code contains an example usage of the %EMTP macro:

```
FILENAME code catalog "sashelp.emutil.emtp.source";
%INCLUDE code;
LIBNAME EMGDS "/projects/global_datasources";
%EMTP(
  data=mylib.mydata,
  target=sometarget,
  columnsmeta=emds.mydata_cm
  );
```

Note: To view the data source in the Enterprise Miner user interface, you must specify a location for the Enterprise Miner Global Data Source (EMGDS) library. Before you try to run code similar to the sample above, you must first assign the EMGDS library location to SAS. You make the EMGDS library assignment status in either your Enterprise Miner server start-up code, or in your Enterprise Miner project start-up code.

Part 19

SAS Enterprise Miner Model Deployment

Chapter 88

SAS Enterprise Miner 12.3 Model Deployment

SAS Enterprise Miner 12.3 Model Deployment

Introduction to SAS Enterprise Miner Model Deployment

After you develop a data mining model for pattern discovery or detection, you need to deploy the model in your enterprise environment. Data mining models can be deployed in many forms, such as

- Models can be distributed to executive management in report form.
- Models can be archived in storage for future use.
- Models can be shared with other data miners.
- Models can be used to score new data records.
- Models can be published into a repository maintained by dedicated deployment professionals.

Modeling Node Results

Exporting the Results

After you run a node, there are a number of ways to export the contents of your model.

- Right-click the node and select **Create Model Package** from the pop-up menu.
- Select the node in the Diagram Workspace, and then click **Create Model Package** in the SAS Enterprise Miner Toolbar shortcut buttons.
- Click the node and select **Actions** ⇨ **Create Model Package** from the main menu.

You are prompted to enter a name for the model package when you click **Create Model Package**. Choose a name that describes properly either the function or purpose of the process flow diagram or modeling tool.

Storing Model Packages

By default, model packages are stored within the Reports subdirectory of your project directory. The folder is named by a combination of the name that you specified when you saved the model package and a string of random alphanumeric characters.

Model package folders contain the following files:

- miningresult.sas7bcat — SAS catalog that contains a single SLIST file with metadata about the model.
- miningResult.spk — the SAS Model Package File. For more information, see SAS Package Files, directly below this section.
- miningResult.xml — XML file that contains metadata about the modeling node. This file contains the same information as miningresult.sas7bcat.

SAS Package Files

About SAS Package Files

A SAS Package (SPK) file is a container file. Modeling results are exported as SAS Package files. SAS Package files are compressed. The compressed SAS Package file can contain

- SAS files
  - SAS catalogs
  - SAS data sets
  - SAS databases (such as DMDB, FDB, and MDDB)
  - SAS SQL views
- Binary files (such as Excel, GIF, JPG, PDF, PowerPoint, and Word)
- HTML files (including ODS output)
- References strings (such as a URL)
- Text files (such as a SAS program)

Reading SAS Package Files

You can use the following to view SAS Package files:

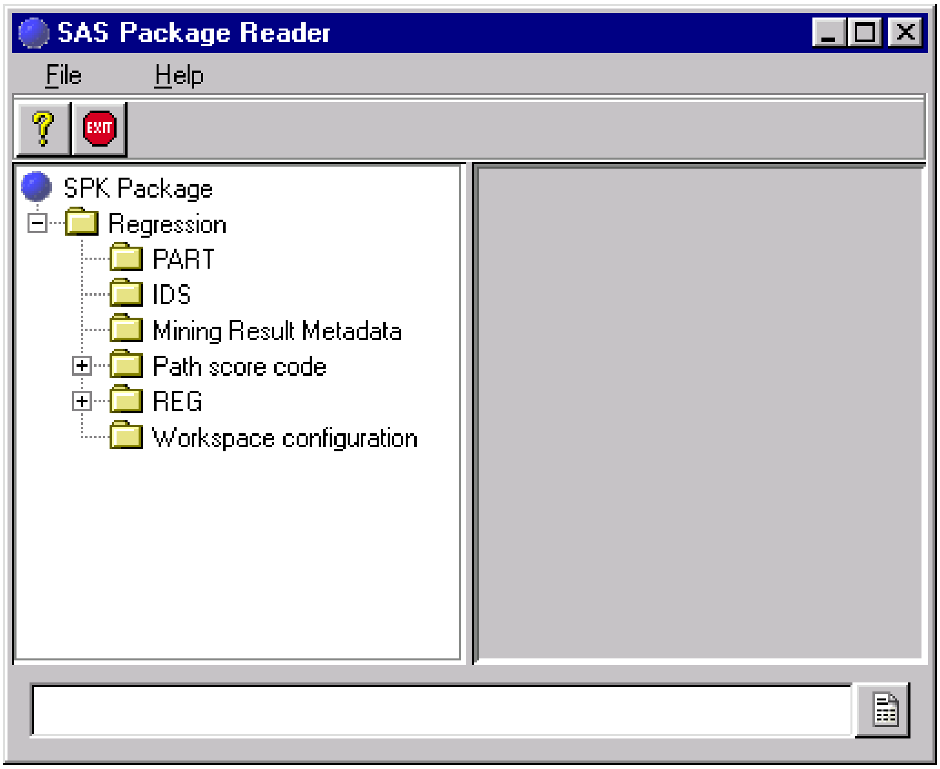
- Common PC file decompression or ZIP utilities, such as WinZip.
- SAS Package Reader. For more information about the SAS Package Reader, see **`http://support.sas.com/rnd/itech/doc9/dev_guide/reader/index.html`**
- The project panel popup menu of the SAS Enterprise Miner user interface. For more information about the project panel popup menu, see "Project Panel Pop-Up Menus" on page 233 .

SAS Package Reader software is shipped with SAS 9.2 and later.

What's in a Model Package File

A model package is a snapshot of a process flow from the perspective of the reporting node. Model Package files enable you to save results and settings for an entire process flow.

The following example is a model package that was created from a SAS Enterprise Miner Regression model. The process flow includes an Input Data (IDS) node, a Data Partition (PART) node, and a Regression (REG) modeling node.



SAS Enterprise Miner 12.3 Package files can contain the following:

- Mining Result Metadata
  - Input Table — XML file that describes the scoring input variables for the model. That is, the variables used in the score node.
  - Mining Result Metadata — the metadata about the flow results. Includes information such as the name of the workspace, the mining function, attributes of modeled target variable(s), mining algorithm, data source information.
  - Output Table — XML file that describes variables created by SAS Enterprise Miner.
  - Statistics Table — XML file that describes the fit statistics table.
  - Target Table — XML file that describes the target variable(s).
- Path Score Code
  - SAS Score Code
    - DATA Step Code — the SAS DATA step score code.
    - Format Code — the SAS formats code for the variables.
  - C Score Code

        - C Score Metadata — the metadata associated with the CScore file. This metadata describes the input variables, output variables, and the C function.
        - C Score for DB2 file — the C score for DB2 database.
        - Score File — the C score code.
    - Java Score Code
        - J Score Metadata — the metadata associated with the JScore file. This metadata describes the input variables, output variables, and the Java Class.
        - Score File — the Java score code.
        - User-Defined Format — the variable formats that are used in the score code.
- Workspace Configuration
    - Batch Code — SAS code that you can use to retrain the model. You can submit this code directly in SAS if you set up a SAS library and data set that correspond to the names in the XML file.
    - Batch Metadata — XML file that describes the datasource that is used to train the model.
    - Properties — XML file that describes the nodes, the positions of the nodes in the diagram, and the node connections in the process flow diagram. To import this file into SAS Enterprise Miner:
    1. Save the file from the SAS Package Reader to a separate directory.
    2. Select **File ⇨ Import Diagram from XML** from the main menu.
    3. select the saved XML file in the Open window.
- Node Results — the files that are stored in this folder depend on the modeling node that you use. For more information, see the Model Package File Results section.

Model Package File Results

The modeling node results files that are archived in SAS Package files vary according to the model. The following table shows all of the files that can be exported from any node. All SAS data sets are converted to CSV files for sharing with multiple applications across multiple systems.

File	Description
Assessment decile statistics	CSV file that contains the assessment statistics for each percentile of rank order measures such as Lift.
Assessment score distribution	CSV file that contains the score distribution statistics for each evenly spaced bucket.
Effects	CSV file that contains the effects table for the regression model.
English rules	Text file that contains decision tree English rules.
Fit Statistics	CSV file that contains the model fit statistics. The statistics are displayed in a single row.

File	Description
Flow score code	Text file that contains the process flow score code. This score code contains code to generate residual variables.
Leaf Statistics	CSV file that contains information about each leaf in a decision tree.
Log	Text file that contains the SAS log from the completed run.
Misclassification table	CSV file that contains the classification statistics.
Output	Text file that contains the SAS output from the node.
Properties	XML file that contains the property settings that were used by the tool during training.
Publish score code	The process flow score code. This score code does not contain code to generate residual variables.
Report metadata	XML file that describes the model package files.
Tabular fit statistics	CSV file that contains the fit statistics in tabular form. Statistic values are displayed in columns that correspond to train, validate, and test data sets.
Training code	The SAS code that is used to train the model.
Training variables	The training variables table in CSV format.
Tree_Plot	CSV file that contains data that is used to populate the decision tree within SAS Enterprise Miner 12.3.
TreePlot.XML	The Predictive Model Markup Language is an XML-based standard for data mining model storage. Several model nodes in SAS Enterprise Miner produce PMML files.

Note: CSV files are limited to 10,000 rows.

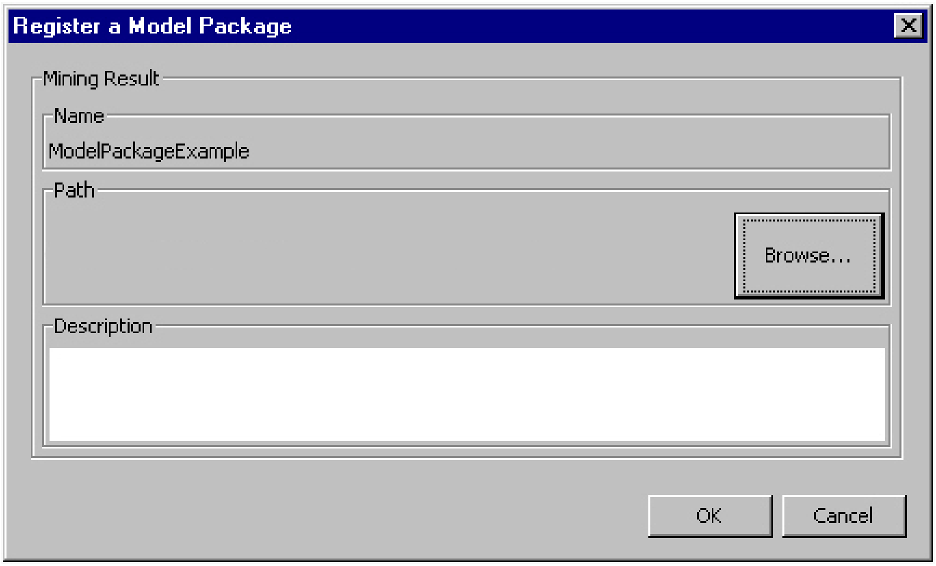
Registering SAS Enterprise Miner Models

You can register model packages that you want to save to the Model Repository.

Use one of the following methods to register your model package:

- Expand the Model Packages folder in the Project Panel and right-click the model package that you want to save. Select **Register** from the menu.
- To open the Model Package window, right-click the Model Package folder in the project tree and select **Open Model**.
- In the Model Package window, select the model that you want to register in the Results table and click **Register**.

  When you click **Register**, the Register a Model Package window appears.

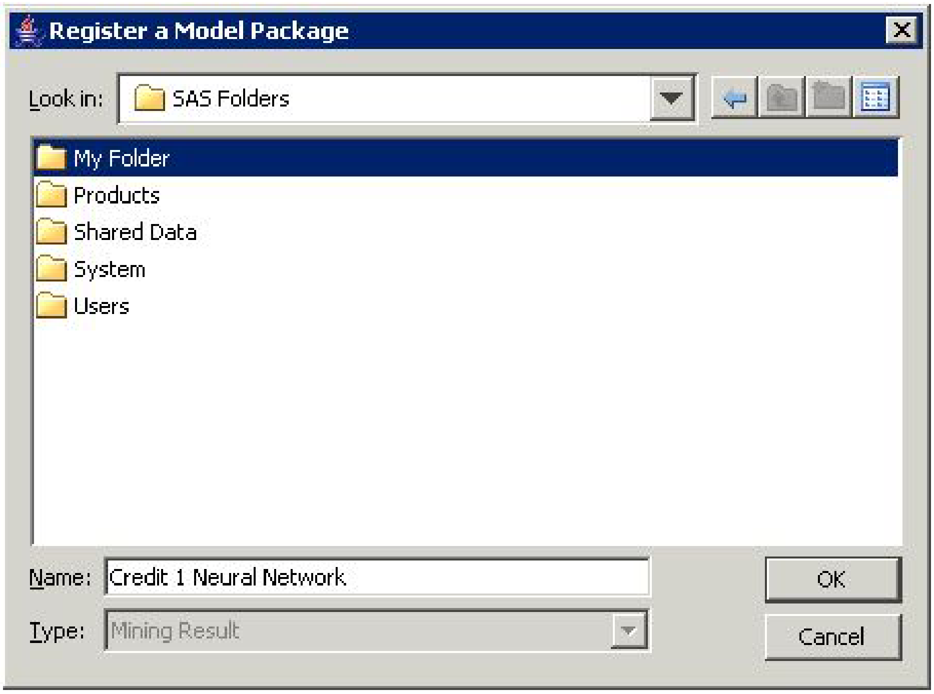


Name: ModelPackageExample

Path:

Description:

Click **Browse** to select a location in the SAS Folders in SAS Metadata. This is the same folder structure that is used to store data mining projects. You can choose your own folder, the Shared Data folder, or a dedicated project folder. SAS administrators can control access to these folders via SAS Management Console, resulting in a secure environment for managing data mining users, projects, and models.



Look in: SAS Folders

My Folder
Products
Shared Data
System
Users

Name: Credit 1 Neural Network

Type: Mining Result

Click **OK** to complete the registration.

When the registration is complete, the following information will be directly saved in the SAS Metadata server:

- Model Name as entered by the user
- Primary Model Algorithm
- Primary Modeling Node
- Name of the SAS Enterprise Miner user
- Date and Time of the model registration
- LIBNAME and Memname of the primary table used to build the model
- All input variables needed for scoring the model
- All output variables produced by the model
- Fit statistics computed for model comparison
- Gains tables computed for model comparison
- SAS Score code
- Location of the Model Archive file

Archiving SAS Enterprise Miner Models

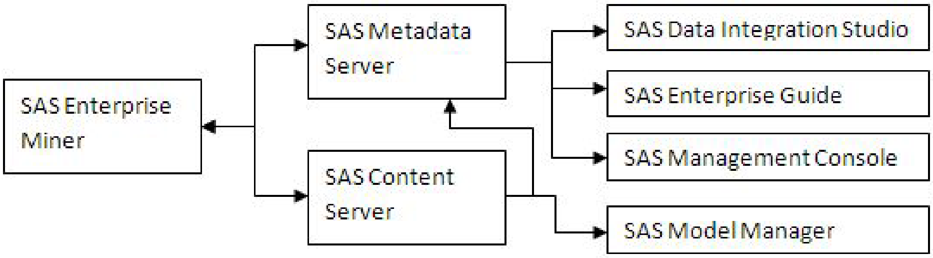
When you register a model, you can store the model package file in a central location used as an archive. Because model package files contain all of the information about how the model was created, SAS Package files can be used for corporate auditing purposes as well as archived storage.

After you create and archive your model package file, you can delete the SAS Enterprise Miner diagram and project from your SAS Enterprise Miner environment if you want. You can view or expand the contents of the model package file using a file compression or ZIP file utility such as WinZip. You can also use SAS Enterprise Miner main menu commands to import the saved model package into SAS Enterprise Miner or SAS Model Manager environments.

SAS Administrators should refer to the SAS Content Server documentation for more complete information about Content Server setup and configuration. Your organization can also use a WebDAV server to store model archives. After content server service is available, your SAS Administrator should configure the SAS Enterprise Miner plug-in for the SAS Management Console in order to make the archived files available to data mining users.

Sharing SAS Enterprise Miner Models with Users and Applications

After you register a model to SAS Metadata, the model can be shared with users of many other SAS applications. SAS Metadata serves as a central repository of information about SAS content and users. The following figure shows applications that can read (pull) and write (push) data mining models to SAS Metadata:



- SAS Enterprise Miner can be used to read and write models to the SAS Metadata server, and to read and write models to the SAS Content Server. You can use the following methods to import a model into SAS Enterprise Miner:
  - From the SAS Enterprise Miner main menu, select **File** ⇨ **Open Mode** to select a model from SAS Metadata. You can view model details and create a new process flow diagram if you want to continue the analysis.
  - From the SAS Enterprise Miner main menu, select **File** ⇨ **Open Model Package** to select a SAS Package file from your local file system. You can view the model details and create a new process flow diagram if you want to continue the analysis.
  - Use the **File Import** node to add the model function to an existing data mining diagram, for the purpose of interactively scoring new data and to compare model statistics with other models. This function does not require the model archive.
- SAS Data Integration Studio reads model information from SAS Metadata to create scoring jobs that are fully integrated with query, transformation, input, output, scheduling, and other features targeted at manipulating large scale enterprise data.
- SAS Enterprise Guide reads model information from SAS Metadata to integrate model scoring into business and analytical user processes.
- SAS Management Console manages security and access to data mining models.
- SAS Model Manager reads and writes models from SAS Metadata as well as reads and presents full model details from the model archive, as part of the process of moving a model from a development environment into a production system.

Scoring New Data

Overview

Data mining models are most often used to score new data records. Model training is defined as the process of using historical records to fit and build models. The score code that models produce is a scoring formula that creates new output values as a function of input values. For example, common marketing models use historical purchasing records, product information, and customer details to predict how customers will respond to a promotion.

SAS Code

SAS score code contains all of the steps that are performed in the SAS Enterprise Miner SEMMA (Sample, Explore, Modify, Model, Evaluate) data mining process, including all transformation, imputation, replacement, recoding, binning, clustering, projection, dimension reduction, regression, decision tree, neural network, and rules steps.

Most SAS Enterprise Miner functions produce score code. Base SAS, as well as an increasing number of SAS products, can read data mining models from SAS Metadata and automatically construct and manage the model scoring processes. The SAS products that can read data mining models include SAS Model Manager, SAS Enterprise Guide, and SAS Data Integration Studio.

C and Java Code

SAS Enterprise Miner model scoring formulas can be exported as C code or Java code, so that the model score code can be used on external systems where SAS is not available. Examples of these environments include real-time data processing systems and embedded micro devices.

Before you deploy exported C code or Java code, it is recommended that you review the *SAS Enterprise Miner C and Java Score Code Basics* documentation at **`http://support.sas.com/documentation/onlinedoc/miner/index.html`**.

Exceptions

You can score the overwhelming majority of functions that SAS Enterprise Miner creates using Base SAS, with the following exceptions. The following tool functions require a SAS Enterprise Miner license to support scoring processes. None of the SAS Enterprise Miner functions below can be exported in C or Java code.

- Associations
- Sequences
- Market Baskets
- Support Vector Machines with nonlinear kernels
- Memory Based Reasoning
- Text Miner

PMML

Predictive Model Markup Language (PMML) defines an XML schema that saves the logic embedded in a data mining model. The PMML standard format can be used to move a model from development to a production system that supports PMML.

The following SAS Enterprise Miner 12.3 nodes produce PMML files:

- Associations node
- Autoneural node
- Clustering node
- Decision Tree node
- DMINE Regression node (logistic and linear regression models)
- Neural Network node
- Regression node (multinomial, logistic, and linear regression models)

Scoring logic that uses unsupported functions (such as transformations and imputations) is not included in the PMML files.

For more detailed information about PMML deployment in SAS Enterprise Miner, see the SAS Enterprise Miner Reference Help chapter on PMML on page 55 .

Part 20

SAS Credit Scoring

Chapter 89
SAS Credit Scoring Overview

SAS Credit Scoring Overview

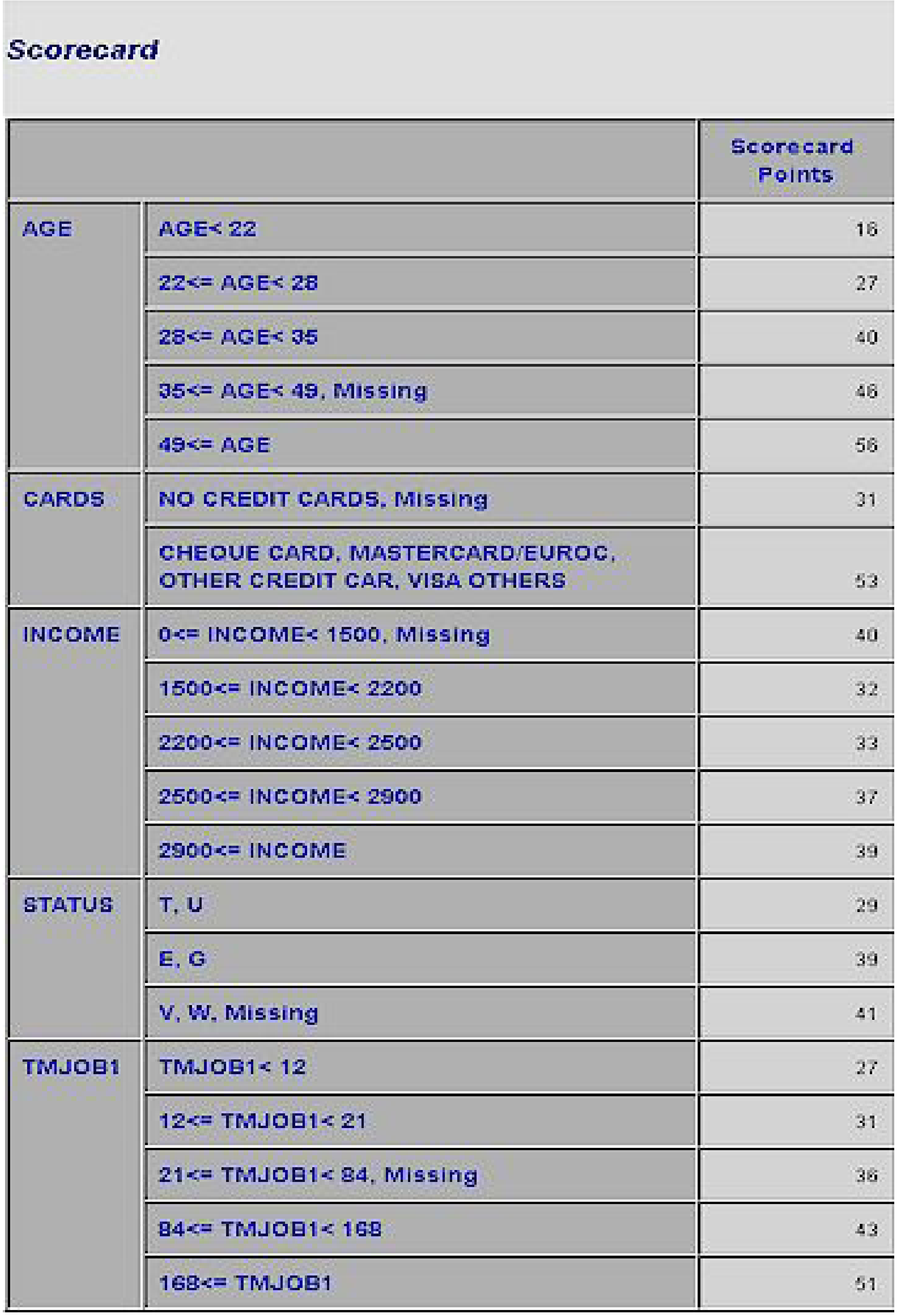
Overview

Introduction

Note: The Credit Scoring for SAS Enterprise Miner solution is not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner, the credit scoring node tools do not appear in your SAS Enterprise Miner software.

Risk Scoring, as with other predictive models, is a tool used to evaluate the level of risk associated with applicants or customers. It provides statistical odds, or probability, that an applicant with any given score will be "good" or "bad." These probabilities or scores, along with other business considerations such as expected approval rates, profit, churn, and losses, are then used as a basis for decision making.

In its simplest form, a scorecard consists of a group of characteristics, statistically determined to be predictive in separating good and bad accounts. An example scorecard is shown below.



Scorecard

		Scorecard Points
AGE	AGE< 22	16
	22<= AGE< 28	27
	28<= AGE< 35	40
	35<= AGE< 49, Missing	46
	49<= AGE	56
CARDS	NO CREDIT CARDS, Missing	31
	CHEQUE CARD, MASTERCARD/EUROC, OTHER CREDIT CAR, VISA OTHERS	53
INCOME	0<= INCOME< 1500, Missing	40
	1500<= INCOME< 2200	32
	2200<= INCOME< 2500	33
	2500<= INCOME< 2900	37
	2900<= INCOME	39
STATUS	T, U	29
	E, G	39
	V, W, Missing	41
TMJOB1	TMJOB1< 12	27
	12<= TMJOB1< 21	31
	21<= TMJOB1< 84, Missing	36
	84<= TMJOB1< 168	43
	168<= TMJOB1	51

Scorecard characteristics can be selected from any of the sources of data available to the lender at the time of the application. Examples of such characteristics are demographics (for example, age, time at residence, time at job, postal code), existing relationship (for example, time at bank, number of products, payment performance, and previous claims), credit bureau (for example, inquiries, trades, delinquency, and public records), real estate data, and so forth.

Each attribute ("Age" is a characteristic and "22<= Age<28" is an attribute) is assigned points based on statistical analyses. Point assignment takes into consideration various factors such as the predictive strength of the characteristics, correlation between characteristics, and operational factors. The total score of an applicant is the sum of the scores for each attribute present for that applicant.

Risk scoring is also used with existing clients on an ongoing basis. In this context, the client's behavioral data with the company is used to predict the probability of negative behavior. Scorecards can also be defined based on the type of data that is used to develop them. Custom scorecards are those developed using data for customers of one organization exclusively. Generic or pooled data scorecards are those built with data from multiple lenders.

Risk scoring, in addition to being a tool to evaluate levels of risk, has also been effectively applied in other operational areas such as the following:

- streamlining the decision-making process. Higher risk and borderline applications can be given to more experienced staff for more scrutiny, while low risk applications are assigned to junior staff. This can be done in branches, credit adjudication centers, and collection departments.
- reducing turnaround time for processing applications through automated decision-making.
- evaluating the quality of portfolios intended for acquisition.
- setting economic and regulatory capital allocation.
- setting pricing for securitization of receivables portfolios.
- comparing the quality of business from different channels, regions, or suppliers.

Risk scoring provides creditors with an opportunity for consistent and objective decision making, based on empirically derived information. Combined with business knowledge, predictive modeling technologies provide risk managers with added efficiency and control over the risk management process.

Before developing an application scorecard you must do the following:

- Identify the business objective for which the application scorecard is to be used.
- Identify applicant characteristics that are most likely to be relevant for assessing credit risk.
- Construct a database containing historical data about previous applicants.

The rudimentary steps for developing an application credit scorecard in Enterprise Miner are as follows:

1. “Create the Accepts Data Source” on page 1460
2. “Partition the Accepts Data into Training and Validation Samples” on page 1460
3. “Perform Univariate Characteristic Screening and Grouping on the Augmented Data Set” on page 1465
4. “Fit a Logistic Model and Generate a Scorecard” on page 1462

Optionally, you can perform the following additional steps:

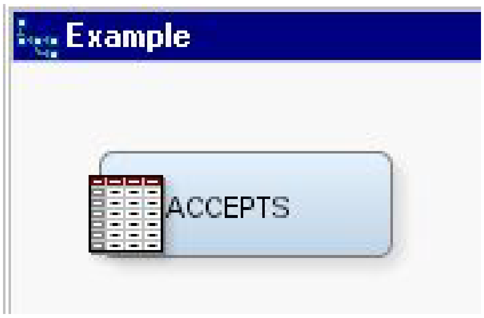
1. “Create a Rejects Data Source” on page 1463
2. “Perform Reject Inference and Create an Augmented Data Set” on page 1463
3. “Partition the Augmented Data Set into Training and Validation Samples” on page 1464
4. “Perform Univariate Characteristic Screening and Grouping on the Augmented Data Set” on page 1465

5. “Fit a Logistic Regression Model and Score the Augmented Data Set” on page 1466
6. “Export the Credit Reporting Data Sets” on page 1467

Your database likely contains records that represent applicants who were previously extended credit and records that represent individuals whose application for credit was rejected. Each observation in the database records an individual's attributes (that is, the observed value of the characteristics). You must develop a set of rules by which you can classify previous applicants who have been extended credit as being either "good" or "bad." The definitions of "good" and "bad" depend on the business objectives and are typically defined so as to segment the applicants according to profitability. In the output of the various credit scoring nodes, you do not see references to "good" or "bad." Instead, you see references to "non-event" and "event." Throughout the documentation and the output, you can equate "event" with "bad" and "non-event" with "good."

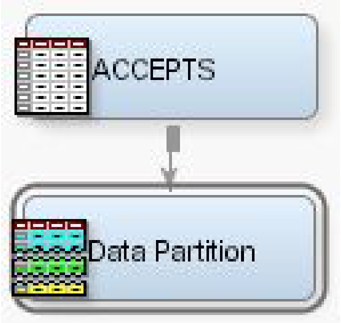
Create the Accepts Data Source

The accepts data set consists of just the applicants who have been previously extended credit. The accepts data set contains examples of non-event and event customers and is used to train the initial model. When you create the accepts data set, you must generate a binary target variable. Assign a value of zero to the target variable for records that represent non-event applicants. Assign a value of 1 to the target variable for records that represent event applicants. Once you have created an accepts data set, store it in a SAS library that is accessible by Enterprise Miner. See Allocating Libraries for SAS Enterprise Miner 12.3 for details. Use the Data Source wizard to configure the accepts data set for use in Enterprise Miner. Assign a role of TRAIN or RAW to your accepts data source. Finally, place the accepts data source to your process flow diagram.



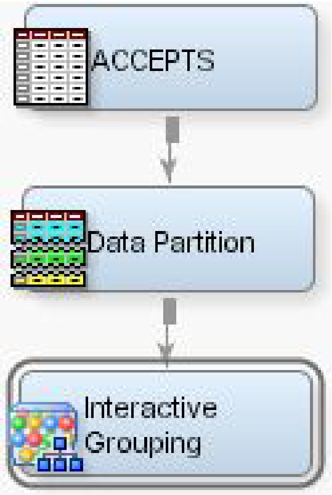
Partition the Accepts Data into Training and Validation Samples

In the process of developing a scorecard, you perform predictive modeling. Thus, it is advisable to partition your data set into training and validation samples. If the total sample size permits, having a separate test sample permits a more robust evaluation of the resulting scorecard. Use the Data Partition node in Enterprise Miner to partition your accepts data set.



Perform Univariate Characteristic Screening and Grouping

To perform univariate characteristic screening and grouping, add an Interactive Grouping node to your process flow diagram.



Grouping refers to the process of purposefully censoring your data. Grouping offers the following advantages:

- It offers an easier way to deal with rare classes and outliers with interval variables.
- It makes it easy to understand relationships, and therefore gain far more knowledge of the portfolio. A chart displaying the relationship between attributes of a characteristic and performance is a much more powerful tool than a simple variable strength statistic. It enables users to explain the nature of this relationship, in addition to the strength of the relationship.
- Nonlinear dependencies can be modeled with linear models.
- It gives the user control over the development process. By shaping the groups, you shape the final composition of the scorecard.
- The process of grouping characteristics enables the user to develop insights into the behavior of risk predictors and to increase knowledge of the portfolio, which can help in developing better strategies for portfolio management.

After the characteristics are grouped, initial characteristic screening is performed. Initial characteristic screening refers to the process of assessing the strength of each characteristic individually as a predictor of performance. The strength of a characteristic is gauged using four main criteria.

- predictive power of each attribute. The weight of evidence (WOE) measure is used for this purpose.
- the range and trend of WOE across grouped attributes within a characteristic.
- predictive power of the characteristic. The Information Value or Gini Statistic is used for this characteristic.
- operational and business considerations (for example, using some logic in grouping postal codes or grouping the debt service ratio to coincide with corporate policy limits).

See the Reference Help for the Interactive Grouping node for details on how WOE, the Information Value, and Gini Statistic are computed.

There is no single criterion that indicates when a grouping can be considered satisfactory. A monotone increase or decrease of the scorecard points or, at a minimum, a smooth variation, denotes a logical relationship between the attributes of a characteristic and the probability of a negative event. As discussed in the Reference Help for the Interactive Grouping node, the WOE of an attribute is inversely related to the event rate of the attribute. Thus, monotonically increasing or decreasing WOE is achieved by grouping the characteristics such that the attributes have monotonically increasing or decreasing event rates.

The Interactive Grouping node can automatically group the characteristics for you. However, the Interactive Grouping node also enables you to perform the grouping interactively. The ability to perform interactive grouping is important because the results of the grouping affects the predictive power of the characteristics, and the results of the screening often indicate the need for regrouping. Thus, the process of grouping and screening is iterative rather than a sequential set of discrete steps.

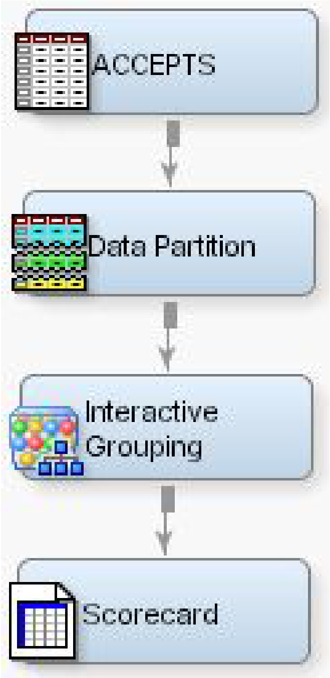
Prior Probabilities and the Interactive Grouping Node

If you enter prior probabilities in the accepts data source's decisions matrix, the Interactive Grouping node does not take the prior probabilities into account when grouping characteristics. Thus, the resulting groups and the WOE are not, in general, the same as when you use a frequency variable.

At the end of the initial characteristic analysis, the scorecard developer will have a set of strong, grouped characteristics, preferably representing independent information types, for use in the regression step that follows.

Fit a Logistic Model and Generate a Scorecard

After characteristic screening and grouping is completed, add a Scorecard node to your process flow diagram.



The Scorecard node fits a logistic regression model and computes the scorecard points for each attribute. You can use either the WOE variables or the group variables that are exported by the Interactive Grouping node as inputs for the logistic regression model. The Scorecard node provides four methods of model selection and seven selection criteria for the logistic regression model. The scorecard points of each attribute are based on the coefficients of the logistic regression model. Details for how the scorecard points are computed are provided in the Reference Help for the Scorecard node. The Scorecard node also enables you to manually assign scorecard points to attributes. The scaling of the scorecard points is also controlled by the three Scaling Options properties of the Scorecard node.

Prior Probabilities and the Scorecard Node

If you enter prior probabilities in the accepts data source's decisions matrix, the parameter estimates of the Scorecard node's logistic regression model are not affected by the prior probabilities. The estimated coefficients of the characteristics are consistent, but the intercept is biased. A prior offset is applied to correct this bias in the intercept. The prior offset is defined as follows:

$$prior\ offset = ln\left(\frac{\rho_1\pi_0}{\rho_0\pi_1}\right)$$

In the preceding equation, ρ_1 and ρ_0 are the proportion of target classes in the sample, and π1 and π0 are the proportion of target classes in the population. When WOE variables are used as inputs in a scorecard model, the scorecard point value for a characteristic's attribute is computed as follows:

$$scorecard\ points_{ij} = -\left(WOE_{ij} * \beta_j + \frac{(\alpha - prior\_offset)}{n}\right) * factor + \frac{offset}{n}$$

In the preceding equation, WOE_{ij} is the weight of evidence of the i^{th} attribute of the j^{th} characteristic; β_j is the regression coefficient on the j^{th} characteristic; α is the intercept of the logistic regression model; and n is the number of characteristics in the logistic regression model. When group variables are used as the inputs in a scorecard model, the scorecard point value for an attribute of a characteristic is computed as follows:

$$scorecard\ points_{ij} = -\left(\beta_{ij} + \frac{(\alpha - prior\_offset)}{n}\right) * factor + \frac{offset}{n}$$

For more details about the use of prior probabilities in Enterprise Miner, see the Prior Probabilities topic in Predictive Modeling.

Create a Rejects Data Source

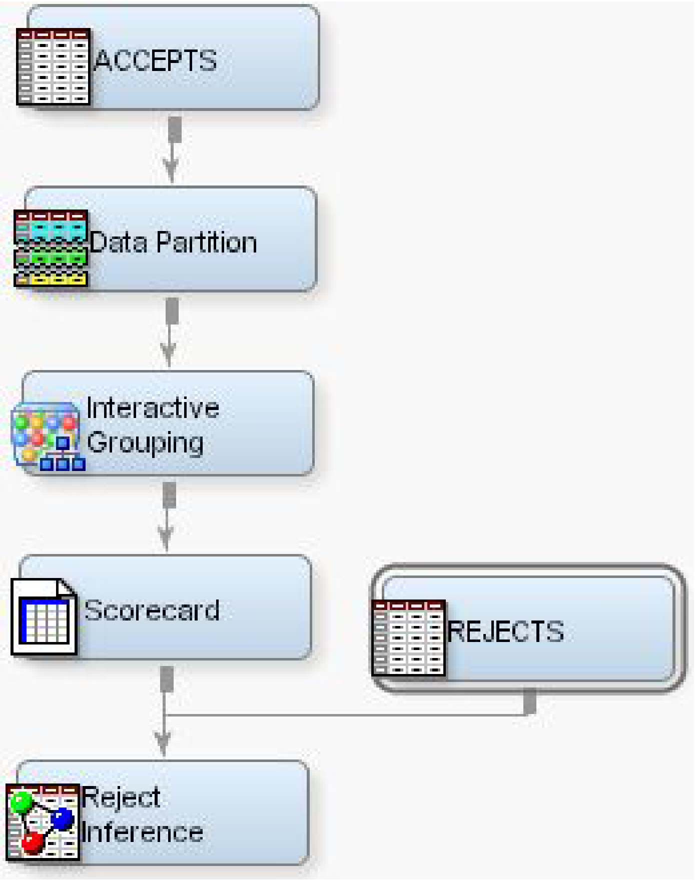
Create a rejects data set by selecting records from your database that represent previous applicants who were denied credit. The rejects data set does not have a target variable. The Reject Inference node automatically creates the target variable for the rejects data when it creates the augmented data set. The rejects data set must include the same characteristics as the accepts data. After you have created the rejects data set, store it in a SAS library that is accessible by Enterprise Miner. See Allocating Libraries for SAS Enterprise Miner 12.3 for details. Use the Data Source wizard to configure the rejects data set for use in Enterprise Miner. Assign a role of SCORE to your rejects data source.

Perform Reject Inference and Create an Augmented Data Set

Credit scoring models are built with a fundamental bias (selection bias). The sample data that is used to develop a credit scoring model is structurally different from the "through-the-door" population to which the credit scoring model is applied. The non-event or event target variable that is created for the credit scoring model is based on the records of applicants who were all accepted for credit. However, the population to which the credit scoring model is applied includes applicants who would have been rejected under the scoring rules that were used to generate the initial model.

One remedy for this selection bias is to use reject inference. The reject inference approach uses the model that was trained using the accepted applications to score the rejected applications. The observations in the rejected data set are classified as inferred non-event and inferred event. The inferred observations are then added to the accepts data set to form an augmented data set. This augmented data set, which represents the "through-the-door" population, serves as the training data set for a second scorecard model.

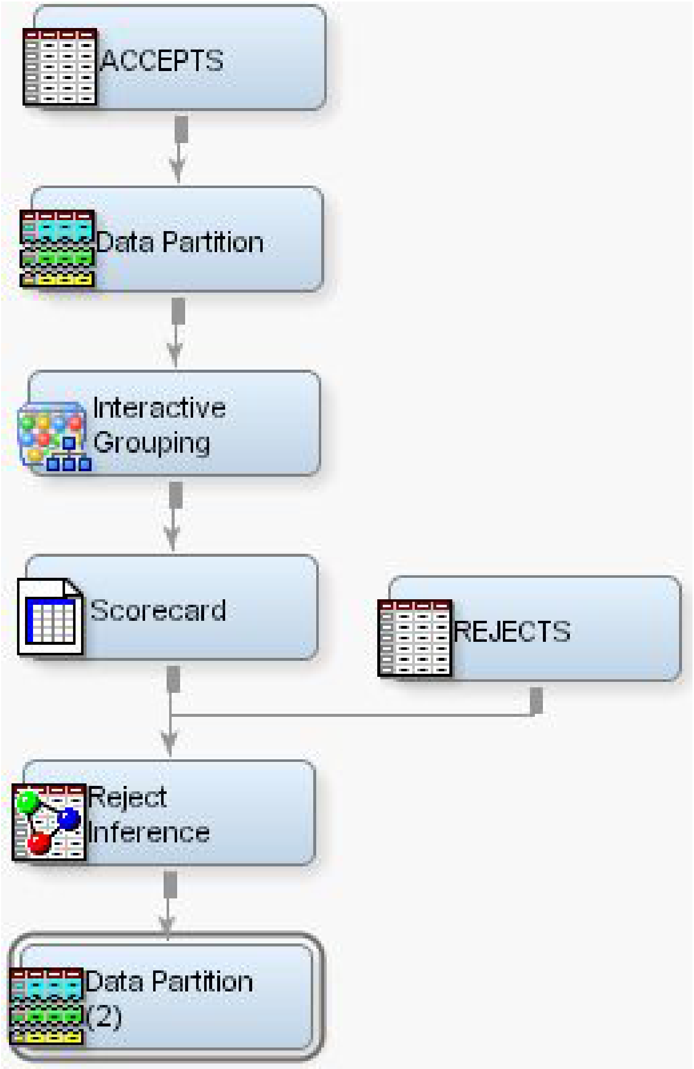
To perform reject inference, add a Reject Inference node and the rejects data source to the process flow diagram. Connect both the Scorecard node and the rejects data source to the Reject Inference node.



Use the Reject Inference node's properties to specify the inference method, the rejection rate, and the event rate increase. For more information about the inference methods and configuring the Reject Inference node's properties, see the Reference Help for the Reject Inference Node.

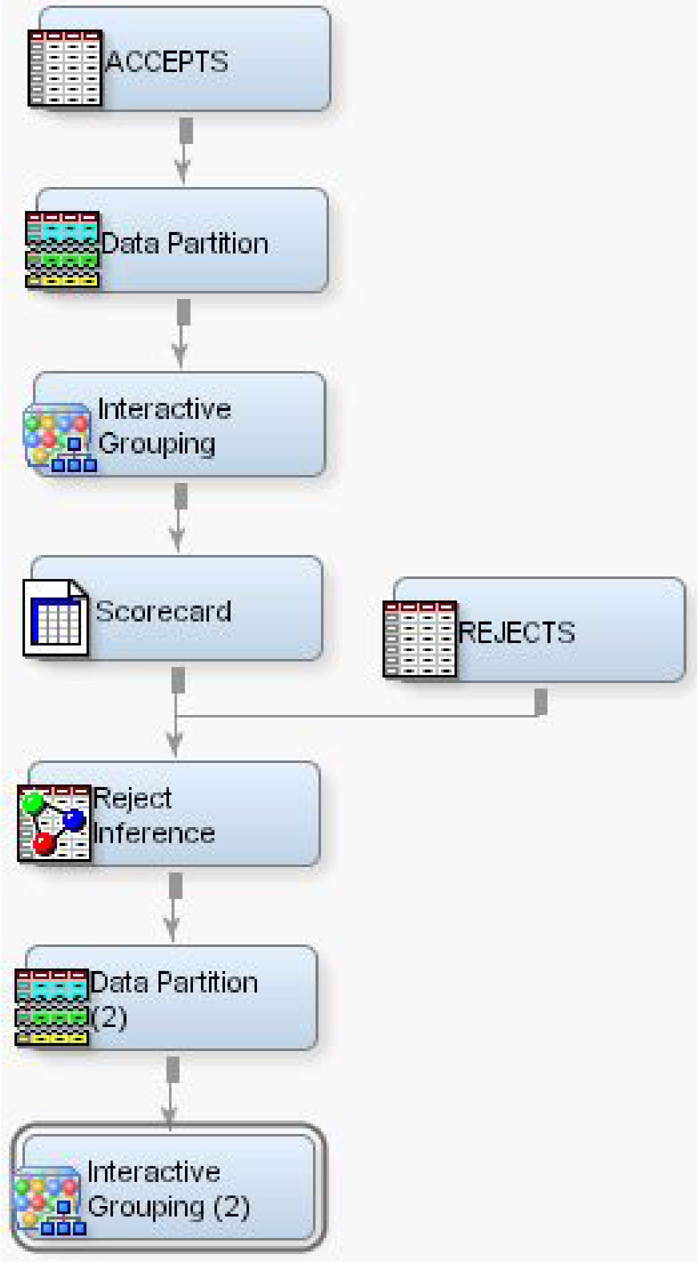
Partition the Augmented Data Set into Training and Validation Samples

The augmented data set that is exported by the Reject Inference Node is used to train a second scorecard model. Before training a model on the augmented data set, add a second Data Partition node to your process flow diagram, and partition the augmented data set into training and validation data sets.



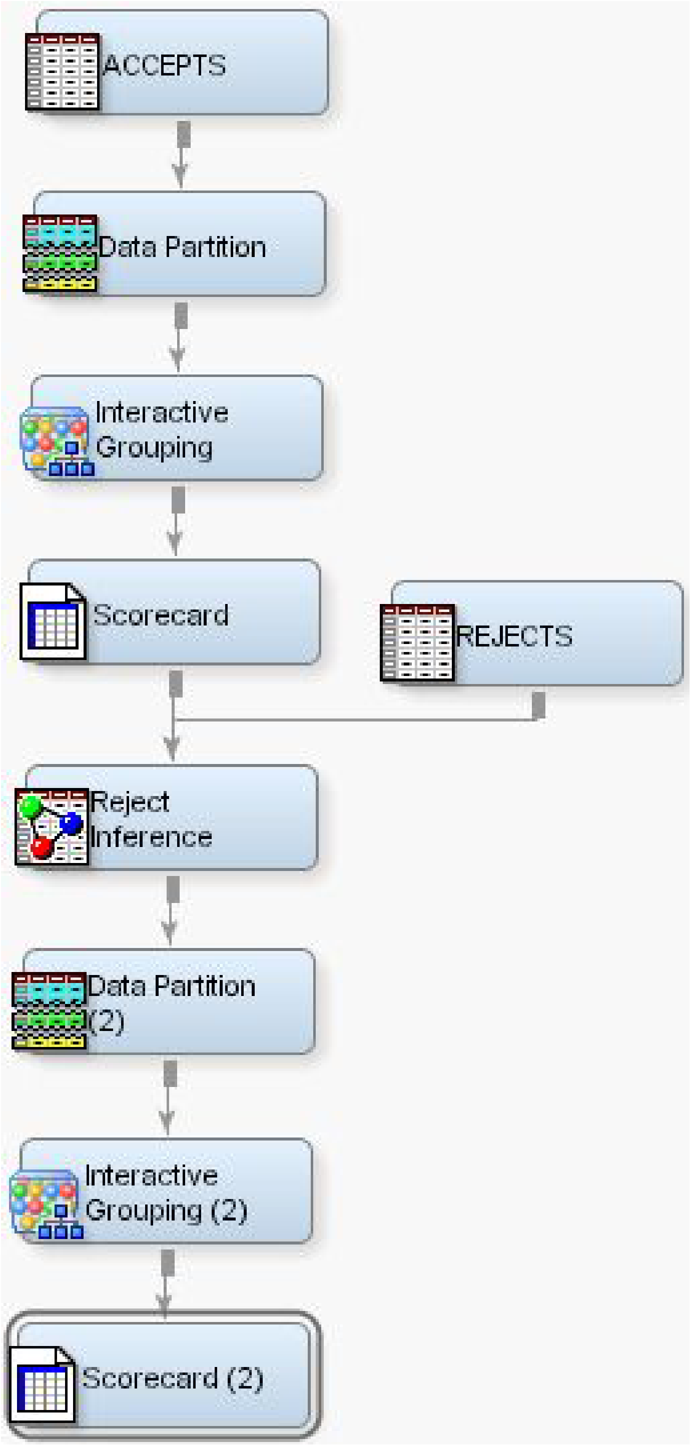
Perform Univariate Characteristic Screening and Grouping on the Augmented Data Set

Your sample has been altered by the addition of the scored rejects data. Add a second Interactive Grouping node to your process flow diagram to recompute the weights of evidence, information values, and Gini statistics. The event rates have changed, so regrouping the characteristics could be beneficial.



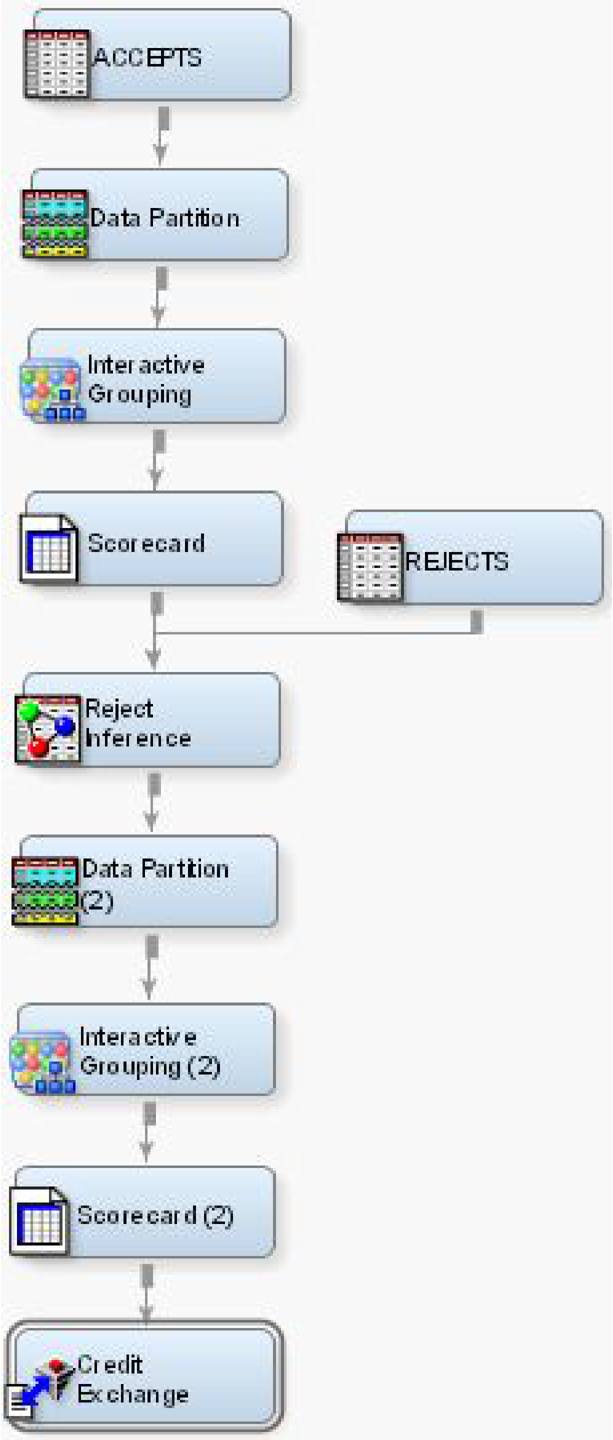
Fit a Logistic Regression Model and Score the Augmented Data Set

Add a second Scorecard node to your process flow diagram. This action enables you to fit a logistic regression on the augmented data set and to generate a score card that is appropriate for the "through-the-door" population of applicants.



Export the Credit Reporting Data Sets

Share your Enterprise Miner credit scoring data with other applications such as the SAS Credit Risk Solution by adding a Credit Exchange node to your process flow diagram.



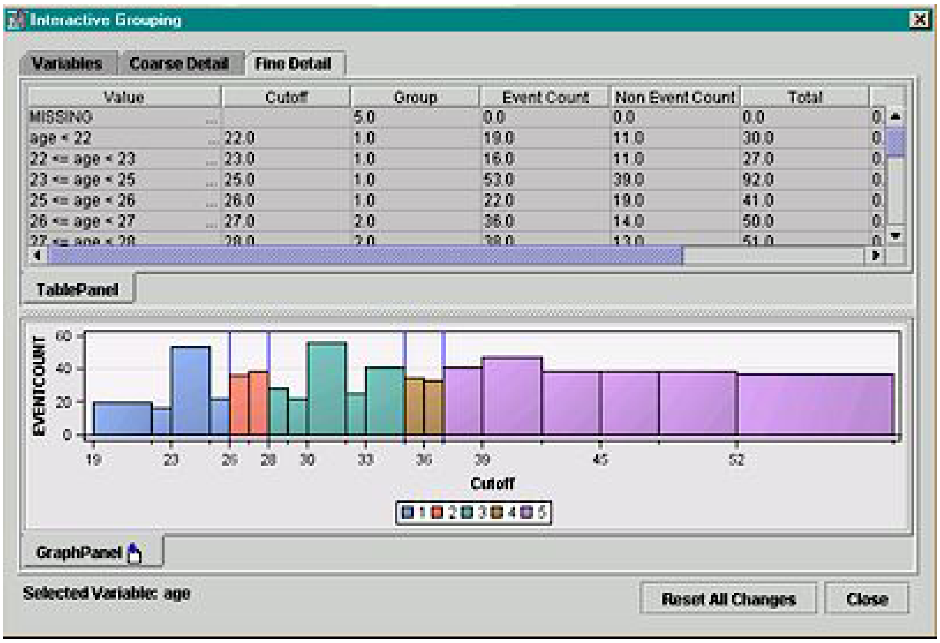
When you add a Credit Exchange node to your credit scoring model, you create a credit scoring statistics data set, a Mapping Table, and score code. The model can be registered to the Enterprise Miner Model Repository and can be used by other solutions, such as SAS Credit Risk. More information about using the Credit Exchange node to export data to the SAS Credit Risk solution is available through the Reference Help for the Credit Exchange node.

Appendix 1: Changes and Enhancements in SAS Credit Scoring

Interactive Grouping Node

The following enhancements have been added to the Interactive Grouping node:

- The Interactive Grouping node is able to handle mixed-type data. Special missing values in numeric columns are mapped to unique bins of the output variable instead of a separate variable. This mapping addresses many types of credit bureau data that contains both numeric values and special codes.
- A new interactive window displays "fine grain" bins that give the user more control over split points for continuous data.



Value	Cutoff	Group	Event Count	Non Event Count	Total
MISSING ...		5.0	0.0	0.0	0.0
age < 22 ...	22.0	1.0	19.0	11.0	30.0
22 <= age < 23 ...	23.0	1.0	16.0	11.0	27.0
23 <= age < 25 ...	25.0	1.0	53.0	39.0	92.0
25 <= age < 26 ...	26.0	1.0	22.0	19.0	41.0
26 <= age < 27 ...	27.0	2.0	36.0	14.0	50.0
27 <= age < 28	28.0	2.0	38.0	13.0	51.0

Selected Variable: age

- An initial high-resolution scan of the data is made to create the "fine grain" bins before a second pass is made to create the "coarse grain" candidate bins. You can interactively use the fine grain split points when constructing the final bins.
- WOE values can now be manually adjusted, and you can interactively override variable Input and Reject roles when performing interactive training.

Scorecard Node

The following enhancements have been added:

- The Scorecard node has been enhanced so that individual terms can be ordered for model selection algorithms such as the stepwise selection algorithm. Model selection options for STAY, STOP, and Term Ordering have been added. The Scorecard node also features Adverse Characteristic Scores for use with individual credit analysis.

Appendix 2: Differences in SAS Credit Scoring between SAS Enterprise Miner 4.3 and SAS Enterprise Miner 12.3

The following tables show the primary differences in SAS Credit Scoring between SAS Enterprise Miner 4.3 and SAS Enterprise Miner 12.3.

Interactive Grouping Node	SAS Enterprise Miner 4.3	SAS Enterprise Miner 12.3
Training Procedure	Uses the SPLIT procedure to generate splitting information. Requires one pass of data per input variable.	Uses the ARBORETUM procedure to generate splitting information. Requires only one pass of the data regardless of the number of input variables.
Control over Procedure Options	No.	Yes. You can specify options such as splitting criterion, missing value assignment, and default grouping assignments.
Export of New Variables	Based on a commit status of Yes or No; Only variables with a commit status of Yes are passed to subsequent nodes.	Based on a variable selection process; all new variables are passed to successor nodes with appropriate variable roles.
Response Rate Plots	No.	Yes. Graphical display shows the response rates across group values for each input variable.
Gini Statistic or Information Value Plots	No.	Yes. Graphical display shows either the Gini Statistic or Information Value across all input variables.
Output of Variable Mappings	No.	Yes. It indicates the mapping from the original input variable to the corresponding WOE and GRP variables.
Interactive Mode	Supported through a series of SAS/AF frames. Only those inputs that were examined during the interactive mode have information passed to successor nodes.	Supported through the application that is based on Windows: SAS Enterprise Miner Interactive Grouping Desktop Application. Generates default groupings for all inputs regardless of whether they are examined in interactive mode.

Scorecard Node	SAS Enterprise Miner 4.3	SAS Enterprise Miner 12.3
Analysis Variables		
Built-in Regression functionality		

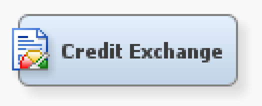
Scorecard Node	SAS Enterprise Miner 4.3	SAS Enterprise Miner 12.3
Score Distribution Chart Trade-off Chart Strategy Curve Event Frequency Chart Scorecard Strength Chart	User Selection	Always created at run time.
Characteristic Analysis Table Selection	Yes.	Yes.
Scorecard Display	Simple PROC PRINT of PROC TABULATE output.	HTML display.
Result Graphics	Static SAS/GRAPH output.	Interactive graphics enable control of look and feel, selection of plot variables, and creation of new plots.

Credit Exchange Node	SAS Enterprise Miner 4.3	SAS Enterprise Miner 12.3
Scorecode generation	No.	Yes. Contains the calculation of EM_PD, EM_LGD, EM_CCF, and EM_EXPOSURE variables.
Generation of Required Inputs Table, Output Table, Target Table, and Accumulation of Path Scorecode	Done at node runtime.	Done during the creation of the mining results package.
Integration Point for other solutions	All information is saved in a user-defined folder structure. Difficult to parse.	All information is saved to the Model Repository, which is easy for solutions to access and parse.
Audit Trail	No. Each time the node was run, the information in the file structure was overwritten.	Yes. Each time a model is registered to the Model Repository a new entry is created in the repository including timestamp, user ID, and a description field.
Access to results by other solutions	No. The folder structure contained no access back to the corresponding model results.	Yes. The model repository includes a link to the corresponding SPK package used to register the model. This package can be opened to view the results alone or it can be exported back into a SAS Enterprise Miner diagram to recreate the project.

Reject Inference Node	SAS Enterprise Miner 4.3	SAS Enterprise Miner 12.3
Classification Charts of Actual versus Inferred	No.	Yes.
Distribution Plots of Actual versus Inferred	No.	Yes.
Summary Statistics of Actual versus Inferred	No.	Yes.
Exported Data Sets	Augmented data only.	Augmented, Train, Validation, and Test data sets.
Augmented Data Variables	Contains WOE, GRP, and LBL variables from original credit scoring process.	WOE, GRP, and LBL variables from original credit scoring process have been removed.

Chapter 90
Credit Exchange Node

Credit Exchange Node



Overview of the Credit Exchange Node

Note: The Credit Scoring for SAS Enterprise Miner solution is not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner, the credit scoring node tools will not appear in your SAS Enterprise Miner software.

The **Credit Exchange** node enables you to exchange the data that is created in SAS Enterprise Miner with the SAS Credit Risk Management solution.

The **Credit Exchange** node creates the following data sources:

- Statistics Table
- Mapping Table

Data Set Requirements for the Credit Exchange Node

You must have the **Scorecard** node that precedes the Credit Exchange nodes in the diagram.

Credit Exchange Node Properties

Credit Exchange Node General Properties

The following general properties are associated with the Credit Exchange Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Credit Exchange node added to a diagram will have a Node ID of CreditEx. The second Credit Exchange node added to a diagram will have a Node ID of CreditEx2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Credit Exchange window. The Imported Data — Credit Exchange window contains a list of the ports that provide data sources to the **Credit Exchange** node. Select the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Credit Exchange window. The Exported Data — Credit Exchange window contains a list of the output data ports that the **Credit Exchange** node creates data for when it runs. Select the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Credit Exchange Node Train Properties

- **Variables** — Use the Variables property of the **Credit Exchange** node to specify the properties of each variable that you want to use in your data source. Select the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, as well as select which variables you want included in reports.

  In the Variables table, you can specify the following credit roles for variables. By default, the variables are not assigned a credit role.

- **LGD** (Loss Given Default) — the actual or predicted loss if an applicant defaults.
- **CCF** (Credit Conversion Factors) — the proportion of credit limit that is used assuming that the credit application is approved.
- **Exposure** — the amount that was loaned.
- **None**

Note: If you assign a credit role to a categorical variable, the assignment will be ignored.

Credit Exchange Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Credit Exchange Node Results

You can open the Results window of the **Credit Exchange** node by right-clicking the node and selecting **Results**. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Results window:

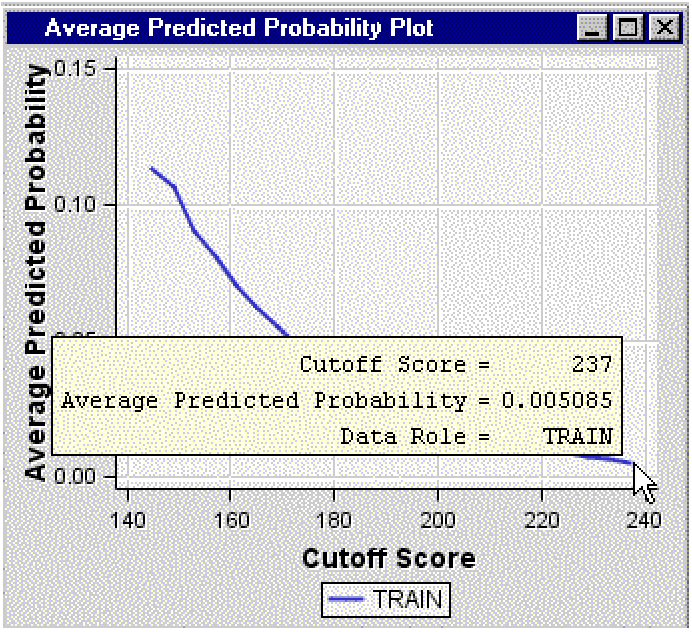
- **Properties**
  - **Settings** — displays a window that includes a read-only table of the **StatExplore** node properties configuration when the node was last run. Use the **Show Advanced Properties** check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the **Credit Exchange** node run. The Run Start Time, Run Duration, and information about whether the run was completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking the column headings.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Credit Exchange node run.

    - **Output** — the SAS output of the Credit Exchange node run. The Credit Exchange SAS output includes a variables summary.
    - **Flow Code** — the SAS code used to produce the output that the **Credit Exchange** node passes on to the next node in the process flow diagram. Flow Score code creates the EM_PD, EM_LGD, EM_CCF, and EM_EXPOSURE variables.
- **Scoring**
    - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the SAS Enterprise Miner environment in custom user applications.
    - **PMML Code** — the **Credit Exchange** node does not generate PMML code.
- **Graphs**
    - "Average Predicted Probability Plot" on page 1476
    - "Average LGD Plot" on page 1477
    - "Average CCF Plot" on page 1478
    - "Average Exposure Plot" on page 1479
- "Statistics Table" on page 1480 — displays the Statistics Table.
- "Mapping Table" on page 1481 — displays the Mapping Table.
- **Tables** — displays the data table for a graph that you select.
- **Plot** — opens the Graph Wizard for the table that you select.

Credit Exchange Node Plots and Tables

Average Predicted Probability Plot

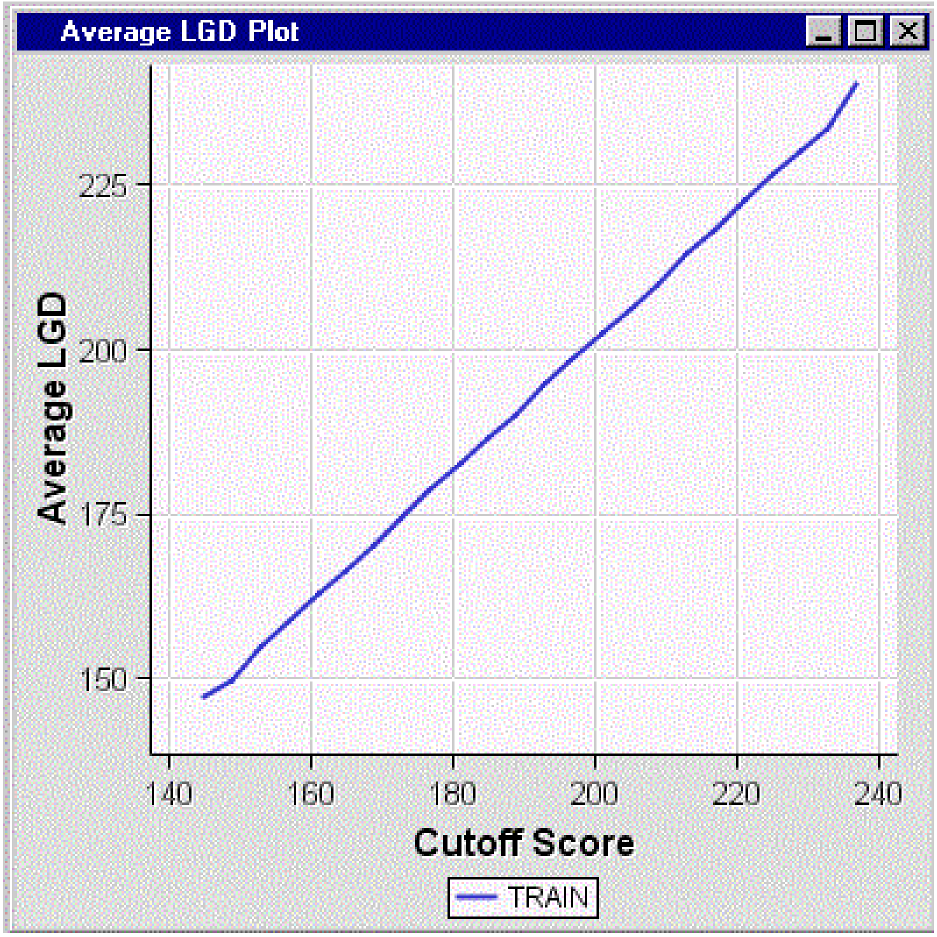
The Average Predicted Probability Plot displays the average value of the predicted probability of events over observations in a score bucket on the vertical axis, and the lower limit of the score points in a score bucket on the horizontal axis. You can see additional information in balloon text if you position your mouse pointer over a data point.


Average Predicted Probability Plot
Average Predicted Probability
0.15
0.10
0.05
0.00
140
160
180
200
220
240
Cutoff Score
TRAIN

```
                  Cutoff Score =      237
Average Predicted Probability = 0.005085
                     Data Role =    TRAIN
```

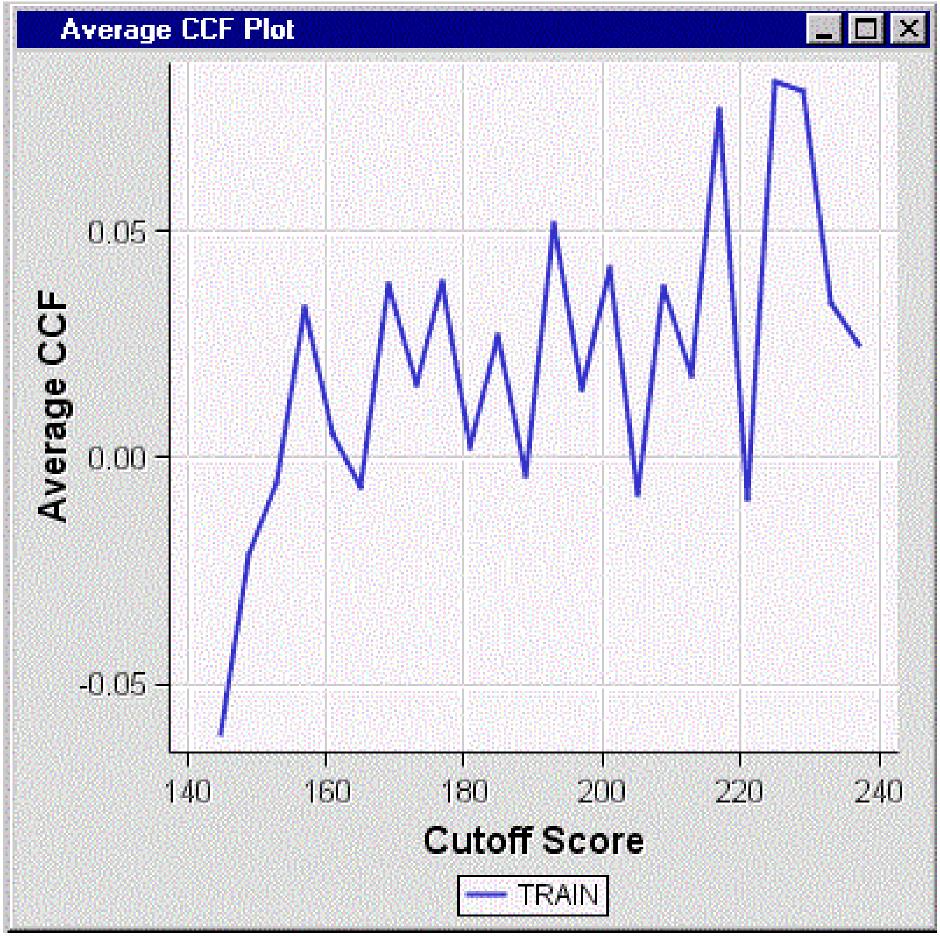
Average LGD Plot

The Average LGD (Loss Given Default) Plot displays the average value of the LGD variable over observations in a score bucket on the vertical axis, and the lower limit of the score points in a score bucket on the horizontal axis. The Average LGD Plot does not appear in the Results window menu unless the Variables property was used to assign the LGD Credit Role to an input variable prior to the node run.



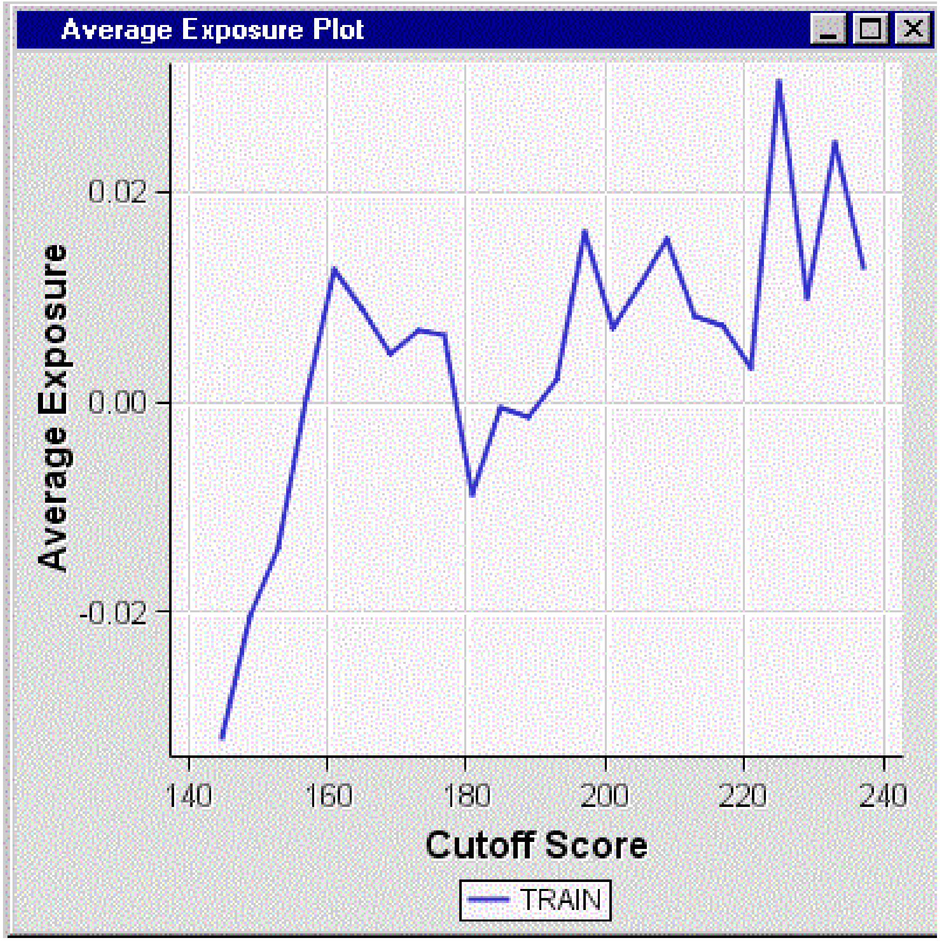
Average CCF Plot

The Average CCF (Credit Conversion Factor) Plot displays the average value of the CCF variable over observations in a score bucket on the vertical axis, and the lower limit of the score points in a score bucket on the horizontal axis. The Average CCF Plot does not appear in the Results window menu unless the Variables property was used to assign the CCF Credit Role to an input variable prior to the node run.



Average Exposure Plot

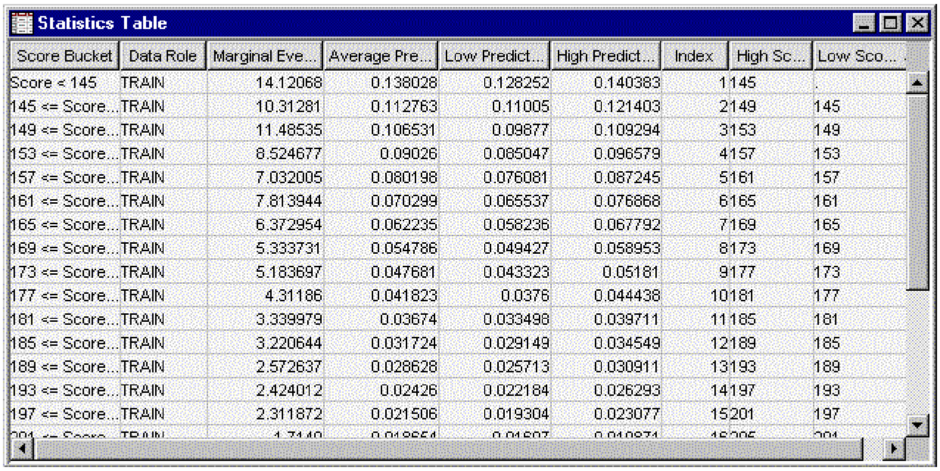
The Average Exposure Plot displays the average value of the Exposure variable for Loan Amount over observations in a score bucket on the vertical axis, and the lower limit of the score points in a score bucket on the horizontal axis. The Average Exposure plot does not appear in the Results window menu unless the Variables property was used to assign the Average Exposure Credit Role to an input variable prior to the node run.



Statistics Table

The Statistics Table displays the following information:

- **Score Bucket** — displays the range of the score points of a bin.
- **Marginal Event Rate** — displays the percentage of events in a bin.
- **Average Predicted Probability of Event** — displays the average posterior probability of events across observations in a bin.
- **Low Predicted Probability Threshold** — displays the lower limit of the posterior probabilities of events in a bin.
- **High Predicted Probability Threshold** — displays the upper limit of the posterior probabilities of events in a bin.
- **Index** — displays the ID of a bin (scorecard bucket).
- **Low Score Threshold** — displays the lower limit of the score points in a bin.
- **High Score Threshold** — displays the upper limit of the score points in a bin.
- **Average LGD** — displays the average value of LGD (Loss Given Default) in a bin if an LGD variable has been defined.
- **Average CCF** — displays the average value of CCF (Credit Conversion Factors) in a bin if a CCF variable has been defined.
- **Average Exposure** — displays the average value of exposure (lent amount) in a bin if an Exposure variable has been defined.



Statistics Table

Score Bucket	Data Role	Marginal Eve...	Average Pre...	Low Predict...	High Predict...	Index	High Sc...	Low Sco...
Score < 145	TRAIN	14.12068	0.138028	0.128252	0.140383	1	145	.
145 <= Score...	TRAIN	10.31281	0.112763	0.11005	0.121403	2	149	145
149 <= Score...	TRAIN	11.48535	0.106531	0.09877	0.109294	3	153	149
153 <= Score...	TRAIN	8.524677	0.09026	0.085047	0.096579	4	157	153
157 <= Score...	TRAIN	7.032005	0.080198	0.076081	0.087245	5	161	157
161 <= Score...	TRAIN	7.813944	0.070299	0.065537	0.076868	6	165	161
165 <= Score...	TRAIN	6.372954	0.062235	0.058236	0.067792	7	169	165
169 <= Score...	TRAIN	5.333731	0.054786	0.049427	0.058953	8	173	169
173 <= Score...	TRAIN	5.183697	0.047681	0.043323	0.05181	9	177	173
177 <= Score...	TRAIN	4.31186	0.041823	0.0376	0.044438	10	181	177
181 <= Score...	TRAIN	3.339979	0.03674	0.033498	0.039711	11	185	181
185 <= Score...	TRAIN	3.220644	0.031724	0.029149	0.034549	12	189	185
189 <= Score...	TRAIN	2.572637	0.028628	0.025713	0.030911	13	193	189
193 <= Score...	TRAIN	2.424012	0.02426	0.022184	0.026293	14	197	193
197 <= Score...	TRAIN	2.311872	0.021506	0.019304	0.023077	15	201	197
201 <= Score...	TRAIN	1.7140	0.018654	0.01607	0.019871	16	205	201

The score code from the **Scorecard** node contains the calculation of scorecard points for each observation, but not the score code that was used to create the scorecard buckets. You must use the values (_low_score_threshold_, _high_score_threshold_, and _index_) from the Statistics Table to generate the score code that is used to create the scorecard buckets. Then, you apply the score code to the scored data set. For example, suppose that you have the following values for the scorecard buckets from the Statistics Table.

index	_low_score_threshold	_high_score_threshold
1	.	113.05
2	113.05	119.65
2	119.65	.

You apply the following score code to the scored data set, which contains the score points information for each observation, in order to assign the observations to the score buckets.

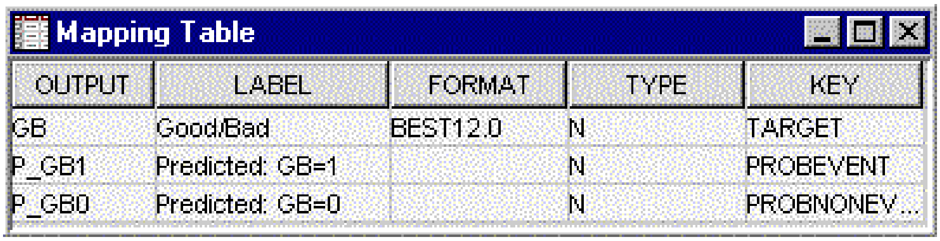
```
if SCORECARD_POINTS <= 113.05
   then SCORECARD_BIN = 1;

if SCORECARD_POINTS >  113.05
   and SCORECARD_POINTS <= 119.65
   then SCORECARD_BIN=2;

if SCORECARD_POINTS >  119.65
   then SCORECARD_BIN=3;
```

Mapping Table

The Mapping Table displays the following information:

- **Output** — the names of the output variables.
- **Label** — the labels of the output variables.
- **Format** — the formats of the output variables.
- **Type** — the variable types, such as N for numerical variable.

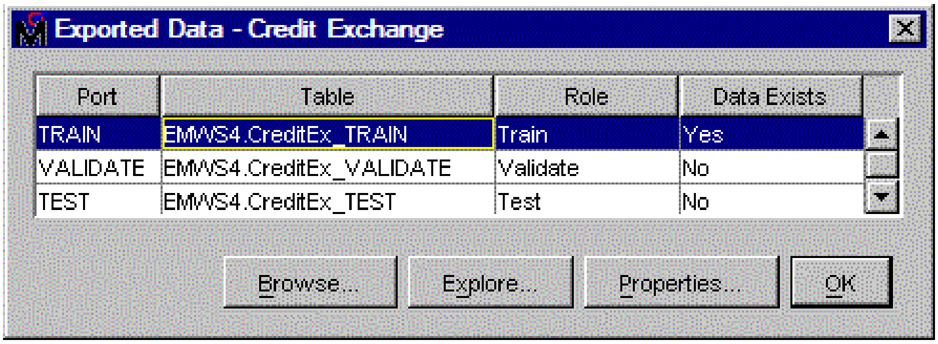
- **Key** — a hardcoded tag value for each observation. This value can be used for further programming.



OUTPUT	LABEL	FORMAT	TYPE	KEY
GB	Good/Bad	BEST12.0	N	TARGET
P_GB1	Predicted: GB=1		N	PROBEVENT
P_GB0	Predicted: GB=0		N	PROBNONEV...

Output Data Sources of the Credit Exchange Node

To view the output Data Sources of the **Credit Exchange** node, select the node in the Diagram Workspace after the node has run. In the Credit Exchange node Properties panel, select the ... button to the right of the Exported Data property to open a table that lists the exported data sets that the **Credit Exchange** node creates.



Port	Table	Role	Data Exists
TRAIN	EMWS4.CreditEx_TRAIN	Train	Yes
VALIDATE	EMWS4.CreditEx_VALIDATE	Validate	No
TEST	EMWS4.CreditEx_TEST	Test	No

If data exists for an exported data set, you can select the row in the table and click:

- **Browse** to open a window where you can browse the first hundred observations of the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contains summary information (metadata) about the table and variables.

The output training, validation, or test data sources from the **Credit Exchange** node contain the original input variables and the following variables:

- EM_CCF — the average of the CCF variable for the score bucket.
- EM_EXPOSURE — the average of the Exposure variable for the score bucket.
- EM_LGD — the average of the LGD variable for the score bucket.
- EM_PD — the average of the posterior probability for the score bucket.

Credit Exchange Node Results

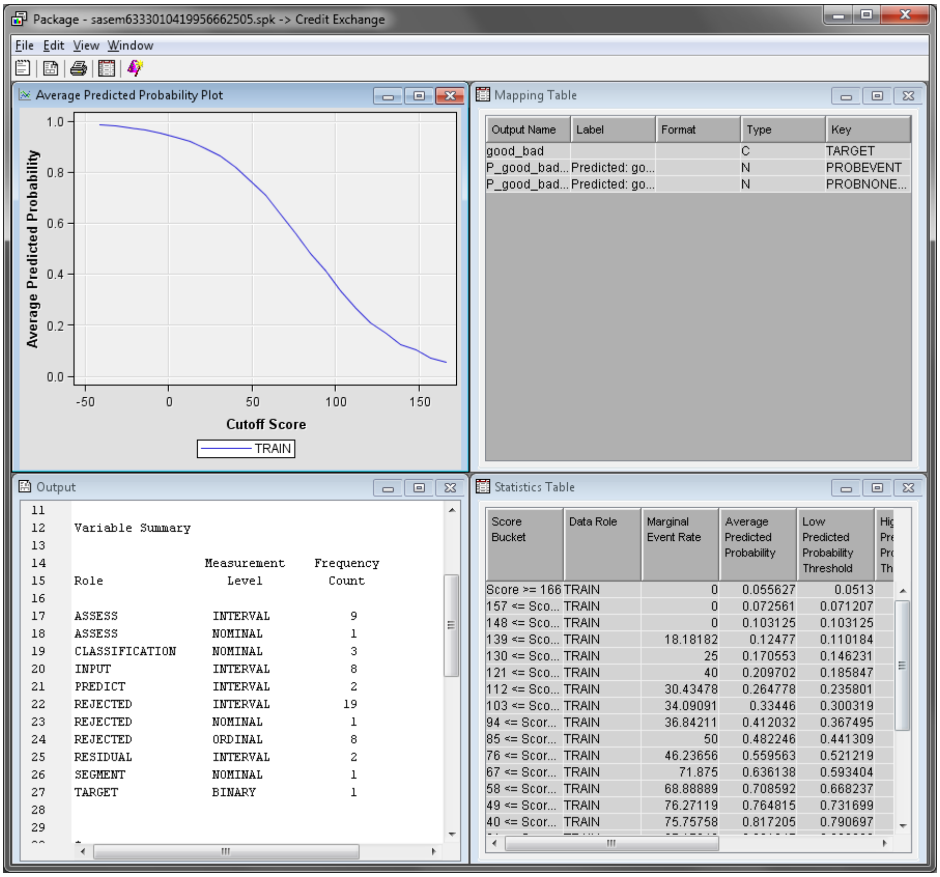
Follow these steps to register and to view the results of the **Credit Exchange** node in the SAS Enterprise Miner Model Repository. The results in the SAS Enterprise Miner

Model Repository can be used to exchange information with other products, such as SAS Credit Risk solution.

1. After the **Credit Exchange** node is run successfully, right-click the node and select **Create Model Package**. In the Input window, enter a **Model Package Name**.

2. Expand the Model Packages folder in the Project panel. Right-click the model package that you created in step 1, and then select **Register**. The Register a Model Package on page 1449 window appears. Provide a description for the model package and click **OK**.

3. After the registration is completed, right-click the model package and select **Open**. In the SPK Viewer window, right-click the **Credit Exchange** node and select **Open**.


Package - sasem6333010419956662505.spk -> Credit Exchange
File Edit View Window
Average Predicted Probability Plot
Average Predicted Probability
1.0
0.8
0.6
0.4
0.2
0.0
-50
0
50
100
150
Cutoff Score
TRAIN
Mapping Table
Output
Statistics Table

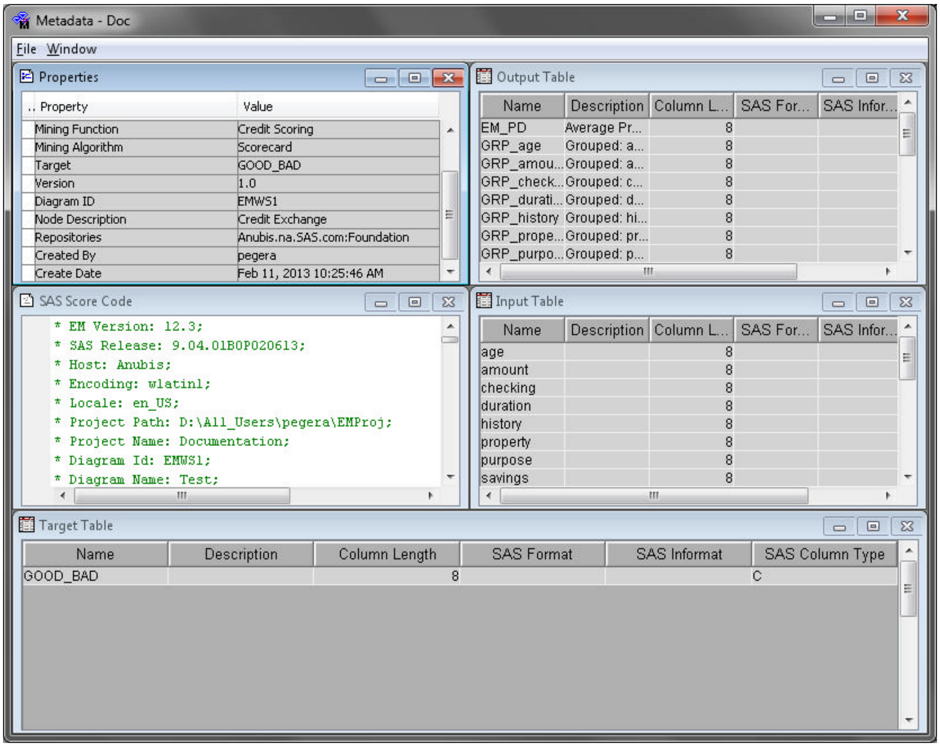
Output Name	Label	Format	Type	Key
good_bad			C	TARGET
P_good_bad...	Predicted: go...		N	PROBEVENT
P_good_bad...	Predicted: go...		N	PROBNONE...

```

Variable Summary

                      Measurement    Frequency
Role                     Level         Count

ASSESS                 INTERVAL          9
ASSESS                 NOMINAL           1
CLASSIFICATION         NOMINAL           3
INPUT                  INTERVAL          8
PREDICT                INTERVAL          2
REJECTED               INTERVAL         19
REJECTED               NOMINAL           1
REJECTED               ORDINAL           8
RESIDUAL               INTERVAL          2
SEGMENT                NOMINAL           1
TARGET                 BINARY            1


```

Score Bucket	Data Role	Marginal Event Rate	Average Predicted Probability	Low Predicted Probability Threshold	Hig Pre Pro Th
Score >= 166	TRAIN	0	0.055627	0.0513	
157 <= Sco...	TRAIN	0	0.072561	0.071207	
148 <= Sco...	TRAIN	0	0.103125	0.103125	
139 <= Sco...	TRAIN	18.18182	0.12477	0.110184	
130 <= Sco...	TRAIN	25	0.170553	0.146231	
121 <= Sco...	TRAIN	40	0.209702	0.185847	
112 <= Sco...	TRAIN	30.43478	0.264778	0.235801	
103 <= Sco...	TRAIN	34.09091	0.33446	0.300319	
94 <= Scor...	TRAIN	36.84211	0.412032	0.367495	
85 <= Scor...	TRAIN	50	0.482246	0.441309	
76 <= Scor...	TRAIN	46.23656	0.559563	0.521219	
67 <= Scor...	TRAIN	71.875	0.636138	0.593404	
58 <= Scor...	TRAIN	68.88889	0.708592	0.668237	
49 <= Scor...	TRAIN	76.27119	0.764815	0.731699	
40 <= Scor...	TRAIN	75.75758	0.817205	0.790697	

Close the Results window. Close the SPK Viewer window.

To view the model summary information, click the **Credit Exchange** node in the process flow diagram. From the toolbar, click the **Metadata** icon. In the Metadata window, double-click your model package.



.. Property	Value
Mining Function	Credit Scoring
Mining Algorithm	Scorecard
Target	GOOD_BAD
Version	1.0
Diagram ID	EMWS1
Node Description	Credit Exchange
Repositories	Anubis.na.SAS.com:Foundation
Created By	pegera
Create Date	Feb 11, 2013 10:25:46 AM

Name	Description	Column L...	SAS For...	SAS Infor...
EM_PD	Average Pr...	8		
GRP_age	Grouped: a...	8		
GRP_amou...	Grouped: a...	8		
GRP_check...	Grouped: c...	8		
GRP_durati...	Grouped: d...	8		
GRP_history	Grouped: hi...	8		
GRP_prope...	Grouped: pr...	8		
GRP_purpo...	Grouped: p...	8		

```
* EM Version: 12.3;
* SAS Release: 9.04.01B0P020613;
* Host: Anubis;
* Encoding: wlatin1;
* Locale: en_US;
* Project Path: D:\All_Users\pegera\EMProj;
* Project Name: Documentation;
* Diagram Id: EMWS1;
* Diagram Name: Test;
```

Name	Description	Column L...	SAS For...	SAS Infor...
age		8		
amount		8		
checking		8		
duration		8		
history		8		
property		8		
purpose		8		
savings		8		

Name	Description	Column Length	SAS Format	SAS Informat	SAS Column Type
GOOD_BAD		8			C

The Metadata window contains the following scoring information about the Credit Exchange node results:

- training variables that were used to create the model. These are the required variables for scoring.
- scoring output variables that are generated and their properties.
- path score code.
- target variable information.

Chapter 91

Interactive Grouping Node

Interactive Grouping Node

Overview of the Interactive Grouping Node

Overview

Note: The Credit Scoring for SAS Enterprise Miner solution is not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner, the credit scoring node tools do not appear in SAS Enterprise Miner.

The **Interactive Grouping** node belongs to the Credit Scoring category. You use the **Interactive Grouping** node to perform grouping and initial characteristic screening. A characteristic is an observable trait, quality, or property of a credit applicant. Scorecard characteristics can be selected from any of the sources of data available to the lender. Examples of such characteristics are demographics (for example, age, time at residence, time at job, and postal code), existing relationship (for example, time at bank, number of products, payment performance, and previous claims), credit bureau (for example, inquiries, trades, delinquencies, and public records), real estate data, and so on. Thus, characteristics can be represented in your data as interval, nominal, ordinal, or binary variables. Each possible value of a characteristic is called an attribute of the characteristic. Thus, before any binning or grouping is performed, binary, ordinal, and nominal variables have a fixed, finite number of attributes while interval characteristics have an infinite number of attributes.

Grouping refers to the process of purposefully censoring your data.

Grouping offers the following advantages:

- It offers an easier way to deal with rare classes and outliers with interval variables.
- It makes it easy to understand relationships and therefore gain far more knowledge of the portfolio. A chart that displays the relationship between attributes of a characteristic and performance is a much more powerful tool than a simple variable strength statistic. It enables users to explain the nature of this relationship, in addition to the strength of the relationship.
- Nonlinear dependencies can be modeled with linear models.
- It enables user control over the development process. By shaping the groups, you shape the final composition of the scorecard.
- The process of grouping characteristics enables the user to develop insights into the behavior of risk predictors and increase knowledge of the portfolio, which can help in developing better strategies for portfolio management.

Initial characteristic screening refers to the process of assessing the strength of each characteristic individually as a predictor of performance. The strength of a characteristic is gauged using four main criteria.

- Predictive power of each attribute. The weight of evidence measure on page 1487 (WOE) is used for this purpose.
- The range and trend of WOE across grouped attributes within a characteristic.
- Predictive power of the characteristic. The Information Value on page 1488 or Gini Statistic on page 1488 is used for this.
- Operational and business considerations (for example, using some logic in grouping postal codes or grouping the debt service ratio to coincide with corporate policy limits).

The **Interactive Grouping** node first performs binning on the interval characteristic. You can choose between two binning methods: quantile and bucket. The quantile method generates groups formed by ranked quantities with approximately the same frequency in each group. The bucket method generates groups by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.

After the interval variables have been pre-binned, a decision tree model is fitted for each characteristic. PROC ARBOR or PROC OPTBIN (if constrained optimal) is used to produce the groups. You can choose between four grouping methods: optimal criterion, quantile, monotonic event rate, and constrained optimal. The optimal criterion method uses one of two criteria: reduction in entropy measure or the p-value of the Pearson Chi-square statistic. The quantile method generates groups with approximately the same frequency in each group. The monotonic event rate method generates groups that result in a monotonic distribution of event rates across all attributes. The event rate is equal to P(event | attribute). This is the conditional probability of an event given that an applicant exhibits a particular attribute. The constrained optimal method finds an optimal set of groups while simultaneously imposing additional constraints, as specified in the node property panel settings.

The **Interactive Grouping** node also provides additional properties that enable you to set parameter values that control the automatic grouping process. Details are provided below under the heading “Interactive Grouping Node Train Properties: Grouping Options” on page 1493 . The node also provides an interactive grouping editor that enables you to manually group variables. The group variables are exported as ordinal variables by default, but you have the option of exporting them as nominal variables.

Note: The **Interactive Grouping** node uses normalized values of categorical variables and considers two categorical values the same if the normalized values are the same. Normalization removes any leading blank spaces from a value, converts lowercase characters to uppercase characters, and truncates the value to 32 characters.

After the characteristics have been grouped, the **Interactive Grouping** node computes the WOE for each attribute for every characteristic.

Weight of Evidence

Weight of evidence (WOE) measures the relative risk of an attribute or group level. The value depends on the value of the binary target variable, which is either "non-event" (target = 0) or "event" (target = 1). An attribute's WOE is defined as follows:

$$WOE_{attribute} = \ln \frac{p^{nonevent}_{attribute}}{p^{event}_{attribute}}$$

The definitions of the quantities in the formula above are as follows:

- $N^{attribute}_{non-event}$ — the number of non-event records that exhibit the attribute
- $N^{total}_{non-event}$ — the total number of non-event records
- $N^{attribute}_{event}$ — the number of event records that exhibit the attribute
- N^{total}_{event} — the total number of event records

For example, consider the following table:

Attributes for AGE	Non-event Count	Event Count	WOE
Age < 24	1440	141	–1.07756
24 <= AGE < 28	2490	138	–0.50841
28 <= AGE < 35	4590	149	0.02649
35 <= AGE < 47	5400	124	0.37268
47 <= AGE	4080	48	1.04145
Total	18000	600	

For the attribute Age<24, WOE = ln{(1440/18000)/(141/600)} = -1.07756.

When used in scorecard modeling, a monotonically increasing or decreasing WOE is desired. WOE is inversely related to the event rate. Thus, when you use the Interactive Grouping node's interactive grouping editor, grouping so that the event rate is monotonically increasing or decreasing ensures a monotonically decreasing or increasing WOE, respectively.

The **Interactive Grouping** node generates and exports the WOE variables as interval variables.

Information Value

The predictive power of a characteristic (that is, its ability to separate high-risk applicants from low-risk ones) is assessed by its Information Value or Gini Index. You can choose to use either method for variable selection, or you can choose not to perform variable selection at all.

The Information Value is a weighted sum of the WOE of the characteristic's attributes. The weight is the difference between the conditional probability of an attribute given an event and the conditional probability of an attribute given a non-event.

$$Information\ Value = \sum_{i=1}^{m} \left(P(attribute_i \mid event) - P(attribute_i \mid non-event)\right) * WOE_i$$

Information values can be any real number. Generally, the higher the information value, the more predictive a characteristic is likely to be. However, there is no way to determine an optimal cutoff value for characteristic selection, so ad hoc rules developed from anecdotal evidence are typically used.

For example, some practitioners use the following criteria:

- less than 0.02: unpredictive
- 0.02 to 0.1: weak
- 0.1 to 0.3: medium
- 0.3 +: strong

The **Interactive Grouping** node uses a default cutoff value of 0.1, but a user can specify another value.

Gini Index

The Gini Index is a Gini coefficient expressed as a percentage. A Gini coefficient ranges from 0 to 1, so the Gini Index ranges from 0 to 100. A Gini Index is a measure of the uniformity or non-uniformity of a distribution. The lower the Gini Index, the more uniformly distributed is a variable. In the context of scorecard modeling, the Gini Index is used to measure how equal the event rates are across the attributes of a characteristic.

- Sort the attributes in descending order of their event rates. For example, suppose a characteristic has m attributes. Then, the sorted attributes are numbered 1, 2, ..., m and attribute 1 has the highest event rate, attribute 2 has the second highest event rate, and so on.
- For each of these sorted attributes, count the number of events (n_i^{event}) and non-events ($n_i^{non\text{-}event}$) in attribute i. Next, count the total number of events (N^{event}) and the total number of non-events ($N^{non\text{-}event}$) in the data. The Gini Index is then computed as follows:

$$Gini\ Index = \left(1 - \frac{2 * \sum_{i=2}^{m}\left(n_i^{event} * \sum_{j=1}^{i-1} n_j^{non-event}\right) + \sum_{k=1}^{m}\left(n_k^{event} * n_k^{non-event}\right)}{N^{event} * N^{non-event}}\right) * 100$$

The node enables you to select a cutoff value for both the information value and the Gini Index. Depending on which variable selection method you choose, the node compares the computed statistic to the specified cut-off value for that statistic. When the computed statistic for a characteristic is less than the specified cut-off value, that characteristic's role is set to Rejected in the node's exported data set. Otherwise, the characteristic's role is set to input in the exported data set.

Entering Prior Probabilities in a Decision Matrix

If you enter prior probabilities in the decisions matrix, the **Interactive Grouping** node does not take the prior probabilities into account when computing WOEs. Thus, the WOEs are not, in general, the same as when you use a frequency variable.

Data Set Requirements for the Interactive Grouping Node

The **Interactive Grouping** node requires a binary or interval target variable.

Interactive Grouping Node Properties

Interactive Grouping Node General Properties

The following general properties are associated with the Interactive Grouping Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Interactive Grouping node that is added to a diagram has a Node ID of Ign. The second Interactive Grouping node added to a diagram has a Node ID of Ign2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Interactive Grouping window. The Imported Data — Interactive Grouping window contains a list of the ports that provide data sources to the **Interactive Grouping** node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Interactive Grouping window. The Exported Data — Interactive Grouping window contains a list of the output data ports that the **Interactive Grouping** node creates data for when it runs. Click the ... button to the right of the Exported Data property to open a table of the exported data.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.

  - **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Interactive Grouping Node Train Properties

The following train properties are associated with the **Interactive Grouping** node:

- **Variables** — Use the Variables property of the **Interactive Grouping** node to specify the properties of each variable that you want to use in your data source. Click the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, as well as select which variables you want included in reports.
- **Interactive Grouping** — Use the Interactive Grouping property to launch the Interactive Grouping Window. Click the ... button to the right of the Interactive Grouping property to open the Interactive Grouping window.

Interactive Grouping Node Train Properties: Interval Target Options

- **Binary Transformation** — specifies the binary transformation method that is used. You can choose from the following options:
  - **Weighting Method** — This option duplicates each observation, assigns each duplicate a binary target variable value, and creates a new frequency variable for each duplicate observation. The frequency variable has values `Interval Target` and `1 – Interval Target` that represent the weights. Note that if the imported data set contains a frequency variable, the new weights are actually a product of the new frequency variable and existing frequency variable.
  - **Mean Cutoff** — The mean value of the interval target variable is used as the cutoff when the node creates a new binary target variable. If a frequency variable exists, then that variable is used when determining the mean value.
  - **Random Cutoff** — Each observation is assigned a randomly generated value from a Uniform(0, 1) distribution. This value is used for the cutoff when the node creates a new binary target variable.
  - **User-Specified Cutoff** — The value that you specify in the **Cutoff Value** property is used as the cutoff when the node creates a new binary target variable.
- **Use Default Min/Max for Scaling** — Specify **Yes** to use the minimum and maximum target variable values when the target variable is scaled to the [0, 1] interval. If you specify **No**, then you must use the **Minimum for Scaling** and **Maximum for Scaling** properties to determine the range of the target variable. This property is enabled only when you use **Weighting Method** or **Random Cutoff** as the **Binary Transformation** method.
- **Minimum for Scaling** — Specifies the minimum value for the range of the target variable. This property is enabled only when you use **Weighting Method** or **Random Cutoff** as the **Binary Transformation** method.
- **Maximum for Scaling** — Specifies the maximum value for the range of the target variable. This property is enabled only when you use **Weighting Method** or **Random Cutoff** as the **Binary Transformation** method.
- **Allow Out-of-Range Values** — Specifies whether values that fall outside of the range determined by the **Minimum for Scaling** and **Maximum for Scaling**

properties are allowed. If you specify **Weighting Method** as the **Binary Transformation**, then values greater than 1 are truncated to 1 and values less than 0 are truncated to 0.

- **Random Seed** — Specifies the initial random seed value that you want to use for random number generation. The default value is 12345.
- **Cutoff Value** — Specifies the cutoff value that is used when **User-Specified** is selected as the **Binary Transformation** method.

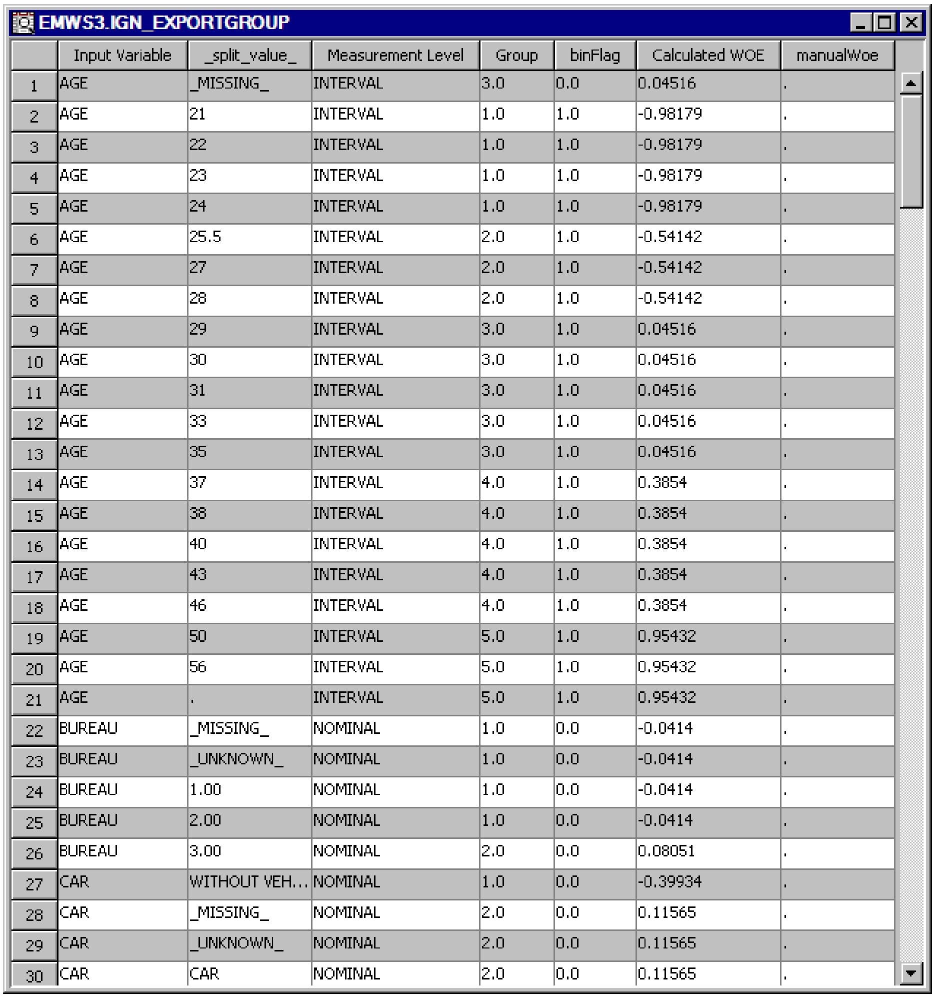
Interactive Grouping Node Train Properties: Pre-Defined Groupings

- **Use Frozen Groupings** — When the Use Frozen Groupings property is set to **Yes**, automatic grouping is not performed on variables where grouping definitions have already been created. When this property is set to **No**, previously defined groupings are ignored; new groupings are created based on the current values of the node properties.
- **Import Grouping Data** — Use the Import Grouping Data property to specify whether to use a table of predefined grouping parameters to perform the grouping on your input data. You can create a grouping data set by setting the Interactive Grouping **Create Grouping Data** property (Report properties) to Yes. Then run the **Interactive Grouping** node with your configured groupings. Your predefined grouping data set will be saved in your SAS Enterprise Miner EMWS*n* project work library, with the name IGN_EXPORTGROUP.

  You can either import a SAS data set that was generated by a SAS Enterprise Miner Interactive Grouping node, or create your own SAS data set to import. This grouping data set must be in the following format.

Variable Name	Type	Format	Description
VARIABLE	Character	$32	variable name
_SPLIT_VALUE_	Character	$200	the upper bound of the values in a group if the variable is interval; individual values if the variable is categorical (ordinal, nominal, or binary)
LEVEL	Character	$8	measure level of the variable, such as interval, ordinal, nominal, or binary
GROUP	Interval	8	group ID

  Here is an example of a predefined grouping data set that was created by the **Interactive Grouping** node:



	Input Variable	_split_value_	Measurement Level	Group	binFlag	Calculated WOE	manualWoe
1	AGE	_MISSING_	INTERVAL	3.0	0.0	0.04516	.
2	AGE	21	INTERVAL	1.0	1.0	-0.98179	.
3	AGE	22	INTERVAL	1.0	1.0	-0.98179	.
4	AGE	23	INTERVAL	1.0	1.0	-0.98179	.
5	AGE	24	INTERVAL	1.0	1.0	-0.98179	.
6	AGE	25.5	INTERVAL	2.0	1.0	-0.54142	.
7	AGE	27	INTERVAL	2.0	1.0	-0.54142	.
8	AGE	28	INTERVAL	2.0	1.0	-0.54142	.
9	AGE	29	INTERVAL	3.0	1.0	0.04516	.
10	AGE	30	INTERVAL	3.0	1.0	0.04516	.
11	AGE	31	INTERVAL	3.0	1.0	0.04516	.
12	AGE	33	INTERVAL	3.0	1.0	0.04516	.
13	AGE	35	INTERVAL	3.0	1.0	0.04516	.
14	AGE	37	INTERVAL	4.0	1.0	0.3854	.
15	AGE	38	INTERVAL	4.0	1.0	0.3854	.
16	AGE	40	INTERVAL	4.0	1.0	0.3854	.
17	AGE	43	INTERVAL	4.0	1.0	0.3854	.
18	AGE	46	INTERVAL	4.0	1.0	0.3854	.
19	AGE	50	INTERVAL	5.0	1.0	0.95432	.
20	AGE	56	INTERVAL	5.0	1.0	0.95432	.
21	AGE	.	INTERVAL	5.0	1.0	0.95432	.
22	BUREAU	_MISSING_	NOMINAL	1.0	0.0	-0.0414	.
23	BUREAU	_UNKNOWN_	NOMINAL	1.0	0.0	-0.0414	.
24	BUREAU	1.00	NOMINAL	1.0	0.0	-0.0414	.
25	BUREAU	2.00	NOMINAL	1.0	0.0	-0.0414	.
26	BUREAU	3.00	NOMINAL	2.0	0.0	0.08051	.
27	CAR	WITHOUT VEH...	NOMINAL	1.0	0.0	-0.39934	.
28	CAR	_MISSING_	NOMINAL	2.0	0.0	0.11565	.
29	CAR	_UNKNOWN_	NOMINAL	2.0	0.0	0.11565	.
30	CAR	CAR	NOMINAL	2.0	0.0	0.11565	.

In this example, five groups are created for the interval variable AGE, and two groups are created for the categorical variable CAR.

- For AGE, group 1 contains the applicants who are younger than 25.
- For AGE, group 2 contains the applicants who are at least 25 but younger than 29.
- For AGE, group 3 contains the applicants who are at least 29 but younger than 37, and the applicants whose age information is missing.
- For AGE, group 4 contains the applicants who are at least 37 but younger than 50.
- For AGE, group 5 contains the applicants who are at least 50.
- For CAR, group 1 contains the applicants who have no cars.
- For CAR, group 2 contains the applicants who have cars, and the applicants for whom information about cars is either unknown or missing from the data.

- **Import Data Set** — If you have an existing predefined grouping data set, and you set the Import Grouping Data property to Yes, you use the Import Data Set property to select the grouping data set that you want to use. Click the ... button to the right of the **Import Data Set** property to open a Select a SAS Table window. Use the Select a SAS Table window to navigate to the EMWS project work library for your diagram (or to the location where you stored a previously created grouping data set), and select the IGN_EXPORTGROUP table that you want to use.

- **Use Pre-Defined WOE Values** — Specifies which pre-defined WOE values are used when either Use Frozen Groupings or Import Grouping Data are set to **Yes**. Valid values are **None**, **Manual**, and **All**.

Interactive Grouping Node Train Properties: Interval Variable Binning Options

- **Apply Level Rule** — Specifies whether the number of unique levels of an interval input should be taken into consideration when determining how the variable should be treated. If Apply Level Rule property is set to **Yes**, the number of unique levels is compared to the number of levels in the Number of Bins property. If the number of unique levels is less than the number of levels specified in the Number of Bins property, the input variable is treated as an ordinal input. If the number of unique levels is greater than the number of levels specified in the Number of Bins property, the input variable is treated as an interval input. If the Apply Level Rule property is set to **No**, all interval input variables will be treated as interval, regardless of the number of unique levels.
- **Binning Method** — Use the Binning Method property to specify the method to use for pre-binning of interval variables.

  Choose one of the following methods:

  - **Quantile** — generates groups formed by ranked quantities with approximately the same frequency in each group.
  - **Bucket** — generates groups by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
- **Number of Bins** — Specify the number of bins that you want to use when pre-binning interval input variables.

Interactive Grouping Node Train Properties: Special Code Options

- **Use Special Codes** — Use the Use Special Codes property to specify whether a mapping data set that contains special codes exists for training.
- **Special Code Data Set** — Use this property to specify the name of the data set used to map values to the corresponding special missing value. Click the ... button to the right of the Special Code Data Set property to open the Select a SAS Table window. This button is available only when you set the Use Special Codes property to yes.

Interactive Grouping Node Train Properties: Grouping Options

- **Interval Grouping Method** — Use the Interval Grouping Method property to specify the method for grouping interval inputs. Choose one of the following methods:
  - **Optimal Criterion** — generates the groupings that are best based on the Criterion property.
  - **Quantile** — generates groups with approximately the same frequency in each group.
  - **Monotonic Event Rate** — generates groups that result in a monotonic distribution of event rates across all levels.
  - **Constrained Optimal** — generates groups based on pre-defined constraints.
- **Ordinal Grouping Method** — Use the Ordinal Grouping Method property to specify the method for grouping ordinal inputs. Choose one of the following methods:

  - **Optimal** — generates the groupings based on optimizing the Splitting Criterion.
  - **Quantile** — generates groups with approximately the same frequency in each group.
  - **Monotonic Event Rate** — generates groups that result in a monotonic distribution of event rates across all levels.
  - **Constrained Optimal** — generates groups based on pre-defined constraints.
- **Tree Based Grouping Options** — Select the ... to specify the Tree Based Grouping Options.
  - **Criterion** — Use the Criterion property to specify the criterion for evaluating candidate splitting rules. Choose one of the following criteria:
    - **Entropy** — reduction in entropy measure.
    - **ProbChisq** — p-value of Pearson Chi-square statistic for target versus the branch node.
  - **Missing Values** — Use the Missing Values property to specify how splitting rules handle observations that contain missing values for a variable. The default value is Separate Branch if Any. Select from the following available missing value policies:
    - **Separate Branch if Any** — assigns the observation to a separate branch, if missing values are found in the training data. If missing observations are not found in the training data, the branch is not created.
    - **Use in Search** — uses missing values during the split search.
    - **Largest Branch** — assigns the observation to the largest branch.
    - **Branch with Smallest Residual** — assigns the observation to the branch that minimizes SSE among observations with missing values.
  - **Minimum Categorical Size** — Use the Minimum Categorical Size property to specify the minimum number of training observations that a categorical value must have before the category can be used in a split search. Permissible values are integers greater than or equal to 1. The default value is 5.
  - **Node Sample** — Use the Node Sample property to specify the maximum within-node sample size n that you want to use to find splits. If the number of training observations in a node is larger than n, then the split search for that node is based on a random sample of size n. Permissible values are integers greater than or equal to 2. The default value for the Node Sample property is 20000.
- **Constrained Optimal Options** — Select the ... to specify the Constrained Optimal Options. This property is available only if the Interval Grouping Method or Ordinal Grouping Method property is set to **Constrained Optimal**.
  - **Apply WOE Monotonicity** — specifies whether a monotonic distribution of Weight of Evidence should be imposed.
  - **Apply Minimum Number of Groups** — specifies whether a minimum number of groups should be created during the generation of the groups.
  - **Minimum Number of Groups** — specifies the minimum number of groups to create.
  - **Apply Minimum Non-Event** — specifies whether a minimum number of non-event observations should be included during the generation of the groups.

  - **Minimum Non-Event Count** — specifies the minimum number of non-event observations that should be included during the generation of the groups.
  - **Apply Minimum Event** — specifies whether a minimum number of event observations should be included during the generation of the groups.
  - **Minimum Event Count** — specifies the minimum number of event observations that should be included during the generation of the groups.
  - **Apply Maximum Total Count** — specifies whether a maximum number of observations should be included during the generation of the groups.
  - **Maximum Total Count** — specifies the maximum number of observations that are included during the generation of the groups.
  - **Apply Minimum WOE Difference** — specifies whether the **Minimum WOE Difference** should be imposed during the generation of the groups.
  - **Minimum WOE Difference** — specifies the minimum difference in WOE values from one generated group to the next.

- **Advanced Constrained Optimal** — Select the ... to open the Advanced Constrained Optimal window. This window contains a table that enables you to set the Constrained Optimal Options for each variable individually. The variables table only contains rows for variables that will use the advanced constrained optimal method. For example, if the **Interval Grouping Method** is set to Constrained Optimal, then only interval inputs will appear in the table.

  This property is available only if the Interval Grouping Method or Ordinal Grouping Method property is set to **Constrained Optimal**.

  The following advanced options are available:

  - **Apply WOE Monotonicity** — specifies whether a monotonic distribution of Weight of Evidence should be imposed.
  - **Apply Minimum Groups** — specifies whether a minimum number of groups should be created during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
  - **Minimum Number of Groups** — specifies the minimum number of groups to create when **Apply Minimum Groups** is set to **User**.
  - **Apply Maximum Groups** — specifies whether a maximum number of groups should be created during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
  - **Maximum Number of Groups** — specifies the maximum number of groups to create when **Apply Maximum Groups** is set to **User**.
  - **Apply Minimum Event Count** — specifies whether a minimum number of event observations should be included during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.

- **Minimum Event Count** — specifies the minimum number of event observations to include when **Apply Minimum Event Count** is set to **User**.
- **Apply Minimum Non-Event Count** — specifies whether a minimum number of non-event observations should be included during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
- **Minimum Non-Event Count** — specifies the minimum number of non-event observations to include when **Apply Minimum Non-Event Count** is set to **User**.
- **Apply Minimum Total Count** — specifies whether a minimum number of observations should be included during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
- **Minimum Total Count** — specifies the minimum number of observations to include when **Apply Minimum Total Count** is set to **User**.
- **Apply Maximum Total Count** — specifies whether a maximum number of observations should be included during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
- **Maximum Total Count** — specifies the maximum number of observations to include when **Apply Maximum Total Count** is set to **User**.
- **Apply Minimum WOE Difference** — specifies whether the **Minimum WOE Difference** should be imposed during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
- **Minimum WOE Difference** — specifies the minimum difference in WOE values from one generated group to the next when **Apply Minimum WOE Difference** is set to **User**.
- **Apply Minimum Cutoff Difference** — specifies whether the **Minimum Cutoff Difference** should be imposed during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.
- **Minimum Cutoff Difference** — specifies the minimum difference between the lower bound and upper bound of the input variable for each generated group when **Apply Minimum Cutoff Difference** is set to **User**.
- **Apply Maximum Cutoff Difference** — specifies whether the **Maximum Cutoff Difference** should be imposed during the generation of the groups. When **Default** is selected, the values from the **Constrained Optimal** options are used. When **None** is selected, the property is not used. When **User** is selected, the corresponding user-entered value is used. If **User** is selected but the corresponding value is entered, then this is equivalent to selecting **None**.

  - **Maximum Cutoff Difference** — specifies the maximum difference between the lower bound and upper bound of the input variable for each generated group when **Apply Maximum Cutoff Difference** is set to **User**.

- **Maximum Number of Groups** — Use the Maximum Number of Groups property to specify the maximum number of groups that are generated.

- **Significant Digits** — Use the Significant Digits property to specify the precision up to which interval values are grouped. The default value is 2. The valid values are integers from 0 to 8. If the Precision property is set to 0, the lower and upper bounds of an interval group are integers.

- **Apply Restrictions** — Use the Apply Restrictions property to specify whether to apply the restriction of minimum group size when automatic grouping is performed. Note that the minimum group size that is determined by either **Count** or **Percent** might be too large to allow splits for a variable. In this case the Gini value would be 0 for that variable. If you notice such behavior, you should decrease the value for **Count** or **Percent**, or change **Apply Restrictions** to **No**.

- **Type** — Use the Type property to specify whether the restriction of minimum group size is based on a percentage or a count.

- **Percent** — Use the Percent property to specify the minimum group size as a percent value. The default value is 5. The valid range is between 0 and 10.

- **Count** — Use the Count property to specify the minimum group size as a count. The minimum group size count is restricted to integer values no larger than 10% of the number of observations in the training data set. Any values specified that exceed the 10% threshold are reset to 10%.

- **Adjust WOE** — Set the Adjust WOE property to Yes to activate the WOE adjustment for a group in which all observations have the same target value. In this case, the adjustment factor that you specify is added to the number of events and the number of non-events in order to calculate WOE:

$$WOE_{attribute}^{adjusted} = \ln\left(\frac{\dfrac{n_{non-event}^{attribute} + adjustment\ factor}{n_{non-event}^{population}}}{\dfrac{n_{event}^{attribute} + adjustment\ factor}{n_{event}^{population}}}\right)$$

- **Adjustment Factor** — Use the Adjustment Factor property to specify the adjustment factor if the Adjust WOE property is set to Yes. A valid value must be between 0 and 1.

Interactive Grouping Node Score Properties

The following score properties are associated with the **Interactive Grouping** node:

- **Group Level** — Use the Group Level property to specify the level assigned to group variables. Possible values are **Ordinal** (default) and **Nominal**.

- **Variable Selection Method** — Use the Variable Selection Method to specify the criteria for variable selection. Possible values are **None**, **Gini Statistic**, and **Information Value**.

- **Gini Cutoff** — Use the Gini Cutoff property to specify a minimum cutoff value for the Gini Statistic. Variables with Gini statistic greater than the specified cutoff are selected as inputs to successor nodes. Otherwise, the variables are assigned a role of Rejected.

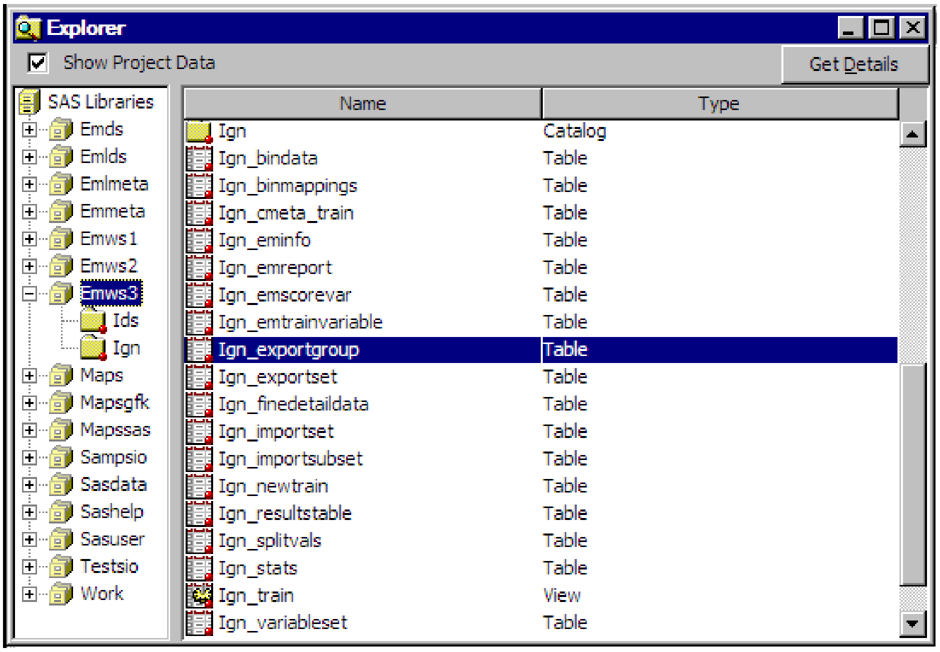
- **Information Value Cutoff** — Use the Information Value Cutoff property to specify a minimum cutoff value for Information Value. Variables with an Information Value greater than the specified cutoff are selected as inputs to successor nodes. Otherwise, the variables are assigned a role of Rejected.

Interactive Grouping Node Report Properties

- **Create Grouping Data** — Use the Create Grouping Data property to specify whether to create a grouping data set. You can use grouping data sets to save your custom groupings for one or more model deployments. You must run the **Interactive Grouping** node with the Create Grouping Data property set to Yes to create a grouping data set.

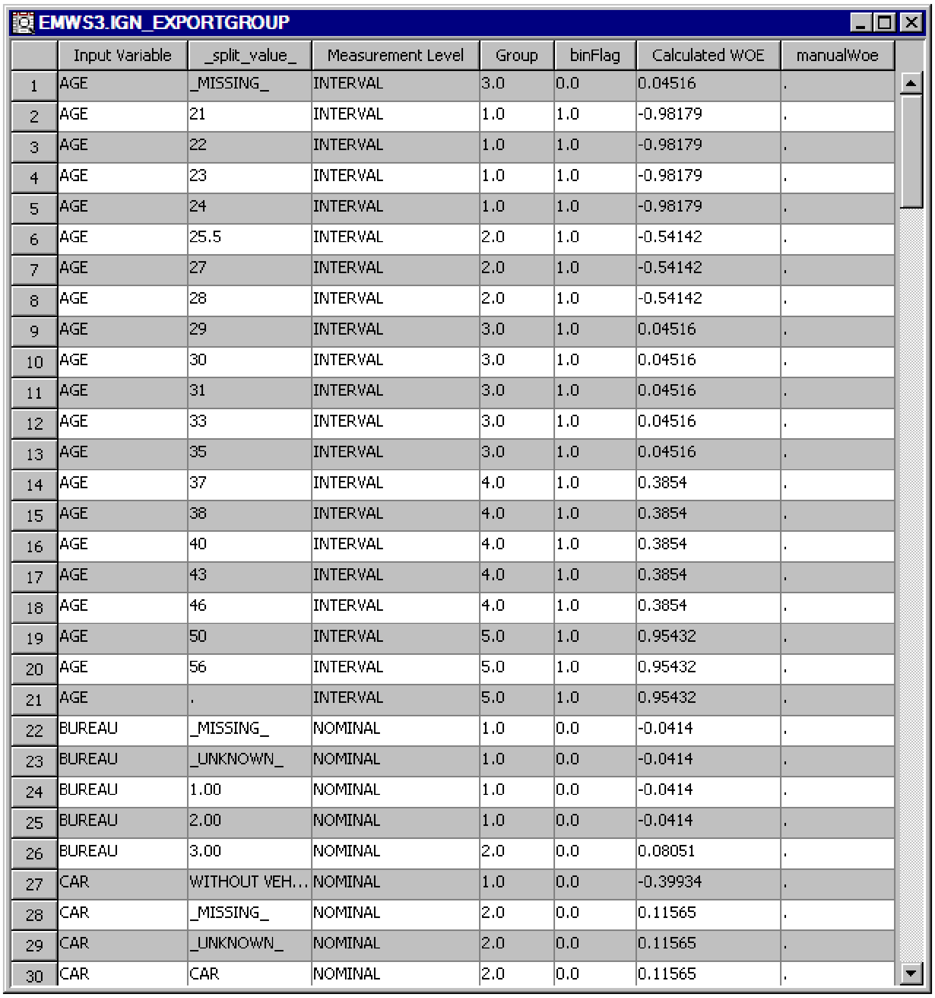
  Grouping data sets are saved in the EMWS*n* project data library that is associated with your process flow diagram. Grouping data sets are named IGN_EXPORTGROUP. You can use the SAS Enterprise Miner file Explorer utility (**View** ⇨ **Explorer** from the main menu) to view the grouping data set that you create.

  Note: Select the **View Project Data** check box in the Explorer window so that the EMWS*n* project data library folders will not remain hidden from view.



Name	Type
Ign	Catalog
Ign_bindata	Table
Ign_binmappings	Table
Ign_cmeta_train	Table
Ign_eminfo	Table
Ign_emreport	Table
Ign_emscorevar	Table
Ign_emtrainvariable	Table
Ign_exportgroup	Table
Ign_exportset	Table
Ign_finedetaildata	Table
Ign_importset	Table
Ign_importsubset	Table
Ign_newtrain	Table
Ign_resultstable	Table
Ign_splitvals	Table
Ign_stats	Table
Ign_train	View
Ign_variableset	Table

You can double-click the **Ign_exportgroup** file in the Explorer window to view the contents of the grouping data table.



EMWS3.IGN_EXPORTGROUP

	Input Variable	_split_value_	Measurement Level	Group	binFlag	Calculated WOE	manualWoe
1	AGE	_MISSING_	INTERVAL	3.0	0.0	0.04516	.
2	AGE	21	INTERVAL	1.0	1.0	-0.98179	.
3	AGE	22	INTERVAL	1.0	1.0	-0.98179	.
4	AGE	23	INTERVAL	1.0	1.0	-0.98179	.
5	AGE	24	INTERVAL	1.0	1.0	-0.98179	.
6	AGE	25.5	INTERVAL	2.0	1.0	-0.54142	.
7	AGE	27	INTERVAL	2.0	1.0	-0.54142	.
8	AGE	28	INTERVAL	2.0	1.0	-0.54142	.
9	AGE	29	INTERVAL	3.0	1.0	0.04516	.
10	AGE	30	INTERVAL	3.0	1.0	0.04516	.
11	AGE	31	INTERVAL	3.0	1.0	0.04516	.
12	AGE	33	INTERVAL	3.0	1.0	0.04516	.
13	AGE	35	INTERVAL	3.0	1.0	0.04516	.
14	AGE	37	INTERVAL	4.0	1.0	0.3854	.
15	AGE	38	INTERVAL	4.0	1.0	0.3854	.
16	AGE	40	INTERVAL	4.0	1.0	0.3854	.
17	AGE	43	INTERVAL	4.0	1.0	0.3854	.
18	AGE	46	INTERVAL	4.0	1.0	0.3854	.
19	AGE	50	INTERVAL	5.0	1.0	0.95432	.
20	AGE	56	INTERVAL	5.0	1.0	0.95432	.
21	AGE	.	INTERVAL	5.0	1.0	0.95432	.
22	BUREAU	_MISSING_	NOMINAL	1.0	0.0	-0.0414	.
23	BUREAU	_UNKNOWN_	NOMINAL	1.0	0.0	-0.0414	.
24	BUREAU	1.00	NOMINAL	1.0	0.0	-0.0414	.
25	BUREAU	2.00	NOMINAL	1.0	0.0	-0.0414	.
26	BUREAU	3.00	NOMINAL	2.0	0.0	0.08051	.
27	CAR	WITHOUT VEH...	NOMINAL	1.0	0.0	-0.39934	.
28	CAR	_MISSING_	NOMINAL	2.0	0.0	0.11565	.
29	CAR	_UNKNOWN_	NOMINAL	2.0	0.0	0.11565	.
30	CAR	CAR	NOMINAL	2.0	0.0	0.11565	.

- **Create Method** — Use the Create Method property to specify whether to append or to overwrite the saved information of groupings for the grouped variables. You can set this property when the Create Grouping Data property is set to **Yes**.
- **Number of Variables** — Use the Number of Variables property to specify the maximum number of variables to be plotted in the Event Rate plot. Only the variables that exceed the information value are plotted.

Interactive Grouping Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Using a Special Codes Data Set for Special Missing Values

The **Interactive Grouping** node can handle mixed-type data and special coded or holdout values. Set the Use Special Codes property to Yes and specify the Special Codes Data Set. Add an entry to the Special Codes data set for each special code or holdout value that you want to map to a special missing value.

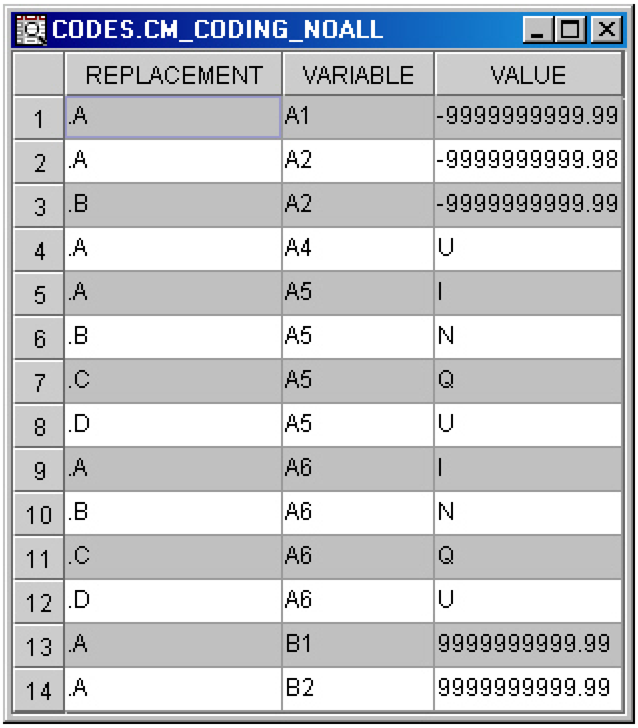
Define your Special Codes data set with the following variables:

- REPLACEMENT — special missing value to which the corresponding holdout value is mapped. This must be a special missing value and this must be a character field.
- VARIABLE — the name of the input variable to be mapped or _ALL_. A value of _ALL_ implies that this mapping applies for any input variable that contains the corresponding value. Note that _ALL_ mapping definitions are applied first, followed by specific variable definitions. If a mapping exists for a value in both a variable mapping and an _ALL_ mapping, the variable mapping supersedes the _ALL_ mapping.

  The length of VARIABLE must be $32. If VARIABLE is a different length, then the Special Codes data set values are not used by the **Interactive Grouping** node.
- VALUE — the original holdout or special coded value. This must be a character field.

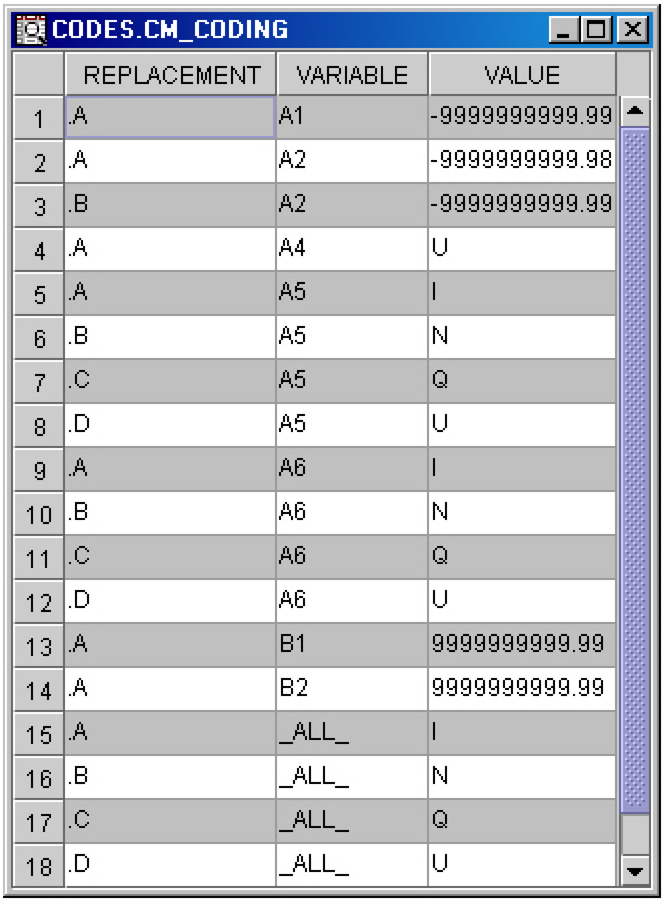
The **Interactive Grouping** node first processes all values not included in the Special Codes data set, creating a default set of bins based on property values. It then creates an additional bin for each unique replacement value specified in the Special Codes data set.

The display below shows a Special Codes data set with the replacement, variable, and value columns.



	REPLACEMENT	VARIABLE	VALUE
1	.A	A1	-9999999999.99
2	.A	A2	-9999999999.98
3	.B	A2	-9999999999.99
4	.A	A4	U
5	.A	A5	I
6	.B	A5	N
7	.C	A5	Q
8	.D	A5	U
9	.A	A6	I
10	.B	A6	N
11	.C	A6	Q
12	.D	A6	U
13	.A	B1	9999999999.99
14	.A	B2	9999999999.99

The display below shows a Special Codes data set that specifies the _ALL_ variable. The **Interactive Grouping** node applies _ALL_ mapping definitions first. If specific definitions for a particular variable exist, that definition supersedes the _ALL_ definition.



	REPLACEMENT	VARIABLE	VALUE
1	.A	A1	-9999999999.99
2	.A	A2	-9999999999.98
3	.B	A2	-9999999999.99
4	.A	A4	U
5	.A	A5	I
6	.B	A5	N
7	.C	A5	Q
8	.D	A5	U
9	.A	A6	I
10	.B	A6	N
11	.C	A6	Q
12	.D	A6	U
13	.A	B1	9999999999.99
14	.A	B2	9999999999.99
15	.A	_ALL_	I
16	.B	_ALL_	N
17	.C	_ALL_	Q
18	.D	_ALL_	U

When you set the Special Codes Data Set property to Yes, the **Fine Detail** tab of the Interactive Grouping application displays a distribution histogram for the missing values.

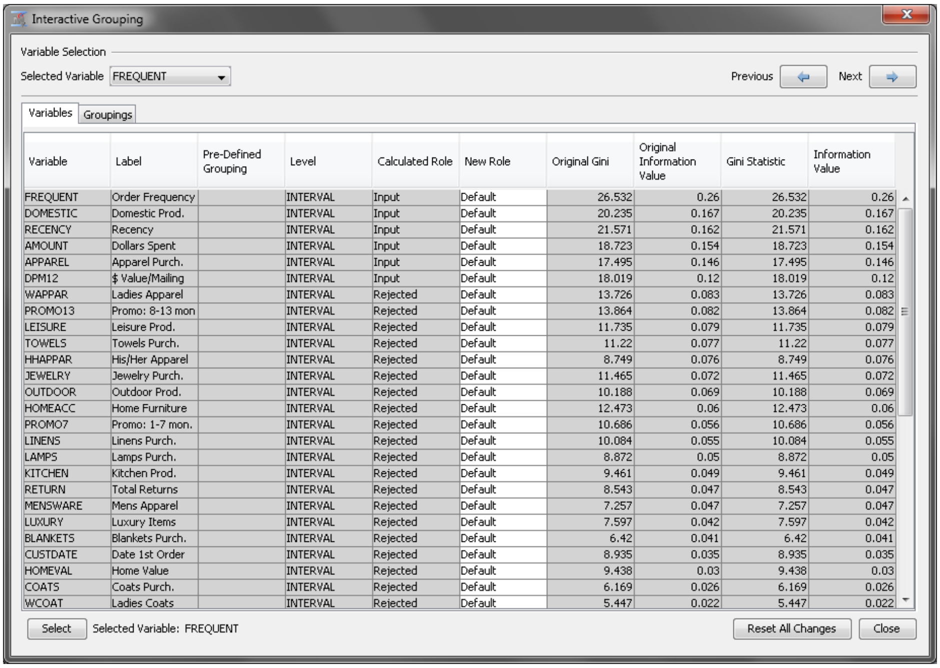
Interactive Grouping Application

Overview

To launch the Interactive Grouping application, select the ... button to the right of the Interactive Binning property. The Interactive Binning window displays "fine grain" bins that give users more control over split points for continuous data. An initial high-resolution scan of the data is made to create the "fine grain" bins before a second pass is made to create the "coarse grain" candidate bins. You can interactively use the fine grain split points when constructing the final bins.

Variables Tab

The **Variables** tab displays a table of variable name, level, role, Gini statistic, information value, and other information. You can select the variable that is plotted in the **Groupings** tab by selecting that variable and clicking the **Select** button in the lower left corner of the window. You can interactively override each variable's role, either **Input**, **Rejected**, or **Default**, when performing interactive training as opposed to having to change variable roles in successor nodes. To change a variable's role, select that variable and right-click the anywhere in the **New Role** column. This action opens a drop-down menu that enables you to select the variable's new role.



Selected Variable FREQUENT

Variable	Label	Pre-Defined Grouping	Level	Calculated Role	New Role	Original Gini	Original Information Value	Gini Statistic	Information Value
FREQUENT	Order Frequency		INTERVAL	Input	Default	26.532	0.26	26.532	0.26
DOMESTIC	Domestic Prod.		INTERVAL	Input	Default	20.235	0.167	20.235	0.167
RECENCY	Recency		INTERVAL	Input	Default	21.571	0.162	21.571	0.162
AMOUNT	Dollars Spent		INTERVAL	Input	Default	18.723	0.154	18.723	0.154
APPAREL	Apparel Purch.		INTERVAL	Input	Default	17.495	0.146	17.495	0.146
DPM12	$ Value/Mailing		INTERVAL	Input	Default	18.019	0.12	18.019	0.12
WAPPAR	Ladies Apparel		INTERVAL	Rejected	Default	13.726	0.083	13.726	0.083
PROMO13	Promo: 8-13 mon		INTERVAL	Rejected	Default	13.864	0.082	13.864	0.082
LEISURE	Leisure Prod.		INTERVAL	Rejected	Default	11.735	0.079	11.735	0.079
TOWELS	Towels Purch.		INTERVAL	Rejected	Default	11.22	0.077	11.22	0.077
HHAPPAR	His/Her Apparel		INTERVAL	Rejected	Default	8.749	0.076	8.749	0.076
JEWELRY	Jewelry Purch.		INTERVAL	Rejected	Default	11.465	0.072	11.465	0.072
OUTDOOR	Outdoor Prod.		INTERVAL	Rejected	Default	10.188	0.069	10.188	0.069
HOMEACC	Home Furniture		INTERVAL	Rejected	Default	12.473	0.06	12.473	0.06
PROMO7	Promo: 1-7 mon.		INTERVAL	Rejected	Default	10.686	0.056	10.686	0.056
LINENS	Linens Purch.		INTERVAL	Rejected	Default	10.084	0.055	10.084	0.055
LAMPS	Lamps Purch.		INTERVAL	Rejected	Default	8.872	0.05	8.872	0.05
KITCHEN	Kitchen Prod.		INTERVAL	Rejected	Default	9.461	0.049	9.461	0.049
RETURN	Total Returns		INTERVAL	Rejected	Default	8.543	0.047	8.543	0.047
MENSWARE	Mens Apparel		INTERVAL	Rejected	Default	7.257	0.047	7.257	0.047
LUXURY	Luxury Items		INTERVAL	Rejected	Default	7.597	0.042	7.597	0.042
BLANKETS	Blankets Purch.		INTERVAL	Rejected	Default	6.42	0.041	6.42	0.041
CUSTDATE	Date 1st Order		INTERVAL	Rejected	Default	8.935	0.035	8.935	0.035
HOMEVAL	Home Value		INTERVAL	Rejected	Default	9.438	0.03	9.438	0.03
COATS	Coats Purch.		INTERVAL	Rejected	Default	6.169	0.026	6.169	0.026
WCOAT	Ladies Coats		INTERVAL	Rejected	Default	5.447	0.022	5.447	0.022

Selected Variable: FREQUENT

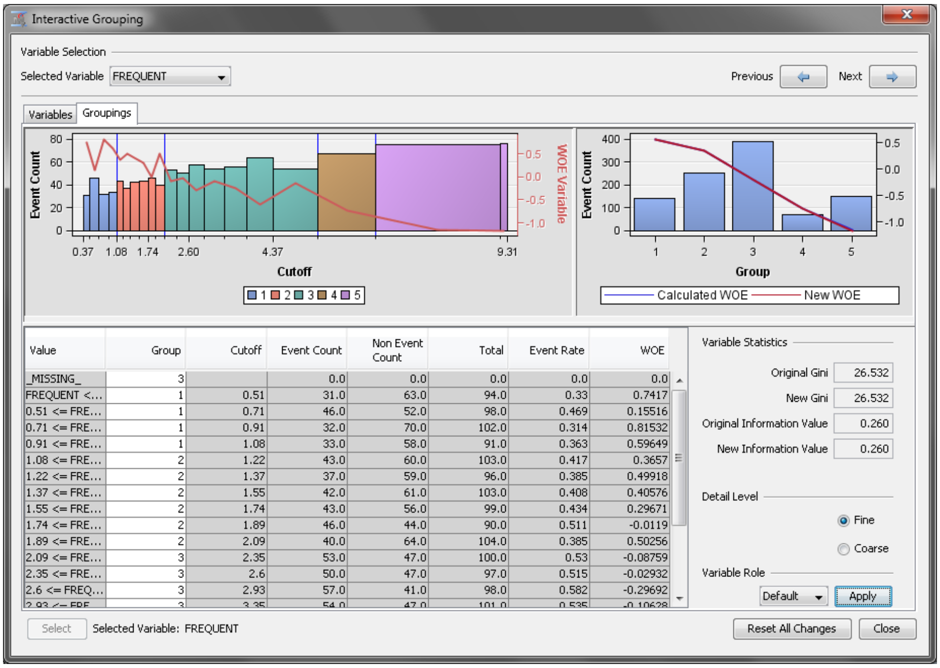
Groupings Tab

The **Groupings** tab is a combination of the **Fine Detail** tab and the **Coarse Detail** tab that were available in older versions of SAS Enterprise Miner. Certain features of the **Fine Detail** tab and the **Coarse Details** tab have been combined, and is described in depth below. In addition, changes that you apply to the groupings are updated automatically.

The **Groupings** tab display tables and plots for one variable at a time. You are able to select the variable of interest in a drop down menu located in the upper left corner of the Interactive Grouping window, or use the selector buttons in the upper right corner to scroll through each variable individually.

Note: The drop down menu and selector buttons are available in the **Variables** tab as well.

By default, the **Groupings** tab will display the **Fine** detail table, as shown in the image below. In addition to the fine detail variables table, there are two plots in the **Groupings** tab. In the upper left, the Cutoff plot is shown. In the upper right, the Weight of Evidence plot and the Event Count plot have been combined. Notice that the scale for the **Event Count** can be found on the left side of the graph and the scale for the **WOE Variable** can be found on the right side of the graph.



Variable Selection

Selected Variable: FREQUENT

Value	Group	Cutoff	Event Count	Non Event Count	Total	Event Rate	WOE
MISSING	3		0.0	0.0	0.0	0.0	0.0
FREQUENT <...	1	0.51	31.0	63.0	94.0	0.33	0.7417
0.51 <= FRE...	1	0.71	46.0	52.0	98.0	0.469	0.15516
0.71 <= FRE...	1	0.91	32.0	70.0	102.0	0.314	0.81532
0.91 <= FRE...	1	1.08	33.0	58.0	91.0	0.363	0.59649
1.08 <= FRE...	2	1.22	43.0	60.0	103.0	0.417	0.3657
1.22 <= FRE...	2	1.37	37.0	59.0	96.0	0.385	0.49918
1.37 <= FRE...	2	1.55	42.0	61.0	103.0	0.408	0.40576
1.55 <= FRE...	2	1.74	43.0	56.0	99.0	0.434	0.29671
1.74 <= FRE...	2	1.89	46.0	44.0	90.0	0.511	-0.0119
1.89 <= FRE...	2	2.09	40.0	64.0	104.0	0.385	0.50256
2.09 <= FRE...	3	2.35	53.0	47.0	100.0	0.53	-0.08759
2.35 <= FRE...	3	2.6	50.0	47.0	97.0	0.515	-0.02932
2.6 <= FREQ...	3	2.93	57.0	41.0	98.0	0.582	-0.29692
2.93 <= FRE...	3	3.35	54.0	47.0	101.0	0.535	-0.10628

Variable Statistics

Original Gini: 26.532

New Gini: 26.532

Original Information Value: 0.260

New Information Value: 0.260

Detail Level

● Fine

○ Coarse

Variable Role: Default Apply

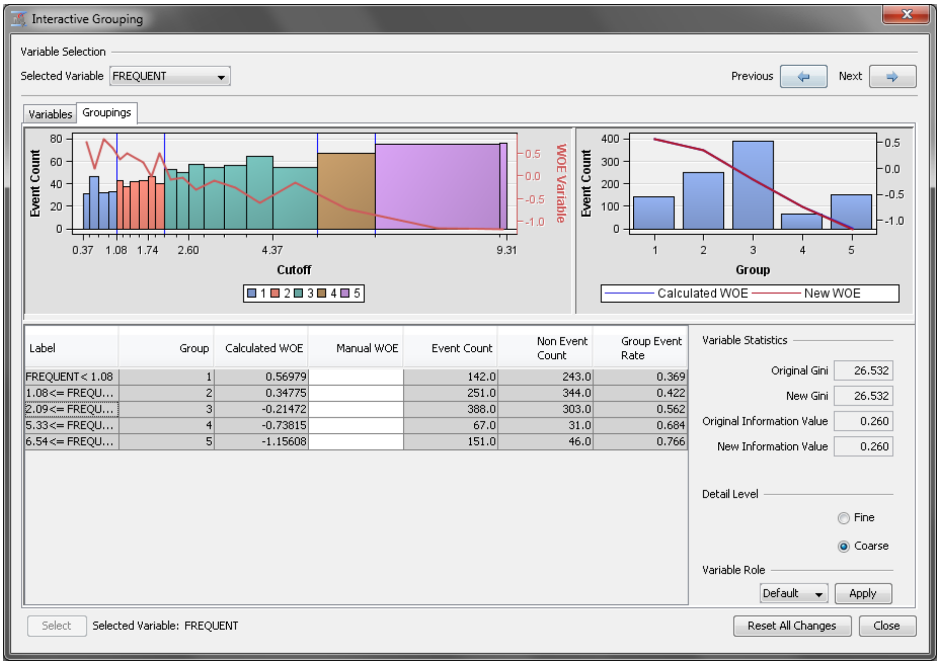
Selected Variable: FREQUENT

In the fine detail view, you are able to perform certain actions, determined by the variable type.

Here are the following actions that you can perform:

- **Merge Bin** — select two or more adjacent rows that have cutoff values on the Table Panel, right-click and select **Merge Bin**. The tables and displays will update to reflect this change.. Available for interval variables.
- **Split Bin** — select a row, right-click, and select **Split Bin** to open the Split Bin window. Enter the new cutoff value in the **Enter New Cutoff Value** field and select **OK**. The table and displays will update to reflect this change. Available for interval variables.
- **New Group** — select one or more rows, right-click and select **New Group**. The table and displays will update to reflect this change. Available for interval variables.
- **Group = #** — select one or more rows, right-click and select **Group = #** to reassign the selected rows to a numbered group. The table and displays will update to reflect this change. Available for interval variables.
- **Assign New Group** — select one or more rows, right-click and select **Assign New Group** to open the Group Selection window. In this window, you can assign the selected rows to an existing group number of a new group number. The table and displays will update to reflect this change. Available for categorical variables.

To view the coarse detail table, locate the **Detail Level** option in the bottom right corner of the screen and select the **Coarse** option. This is shown in the image below.



Selected Variable FREQUENT

Label	Group	Calculated WOE	Manual WOE	Event Count	Non Event Count	Group Event Rate
FREQUENT< 1.08	1	0.56979		142.0	243.0	0.369
1.08<= FREQU...	2	0.34775		251.0	344.0	0.422
2.09<= FREQU...	3	-0.21472		388.0	303.0	0.562
5.33<= FREQU...	4	-0.73815		67.0	31.0	0.684
6.54<= FREQU...	5	-1.15608		151.0	46.0	0.766

Variable Statistics

Original Gini	26.532
New Gini	26.532
Original Information Value	0.260
New Information Value	0.260

Detail Level

○ Fine
● Coarse

Variable Role

Default Apply

Selected Variable: FREQUENT

In the coarse detail table, you are able to manually override the weight of evidence statistic. In the image above, the variable AGE has been selected, and the weight of evidence has been changed for the third bin. This bin contains values of AGE between 28 and 35 and all missing values. Notice in the Weight of Evidence/Event Count plot, that the **New WOE** graph shows this change.

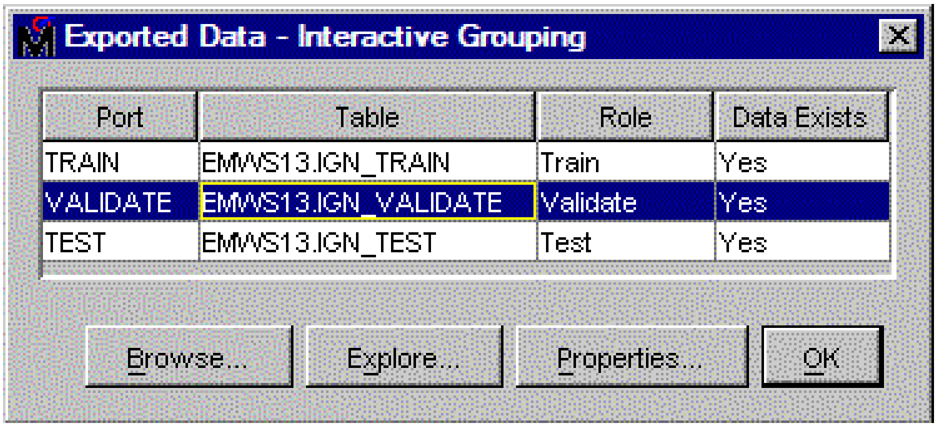
There is a vertical division between the Cutoff plot and the WOE/Event Count plot that enables you to adjust the width of these plots. Similarly, there is a horizontal division that enables you to adjust the height of the variables table.

The **Variable Role** section enables you to modify the role of the current variable. Use the drop down box to set the variable's role without needing to switch back to the **Variables** tab.

Clicking the **Reset All Changes** button reverses any changes that you have made in the current session.

Output Data Sources of the Interactive Grouping Node

You can view the **Interactive Grouping** node output data sources after you successfully run the node. Click the ... button to the right of the Exported Data property in the Properties panel to open a table that lists the exported Interactive Grouping data sets.



Port	Table	Role	Data Exists
TRAIN	EMWS13.IGN_TRAIN	Train	Yes
VALIDATE	EMWS13.IGN_VALIDATE	Validate	Yes
TEST	EMWS13.IGN_TEST	Test	Yes

If data exists for an exported data set, you can select the row in the table and select one of the following buttons:

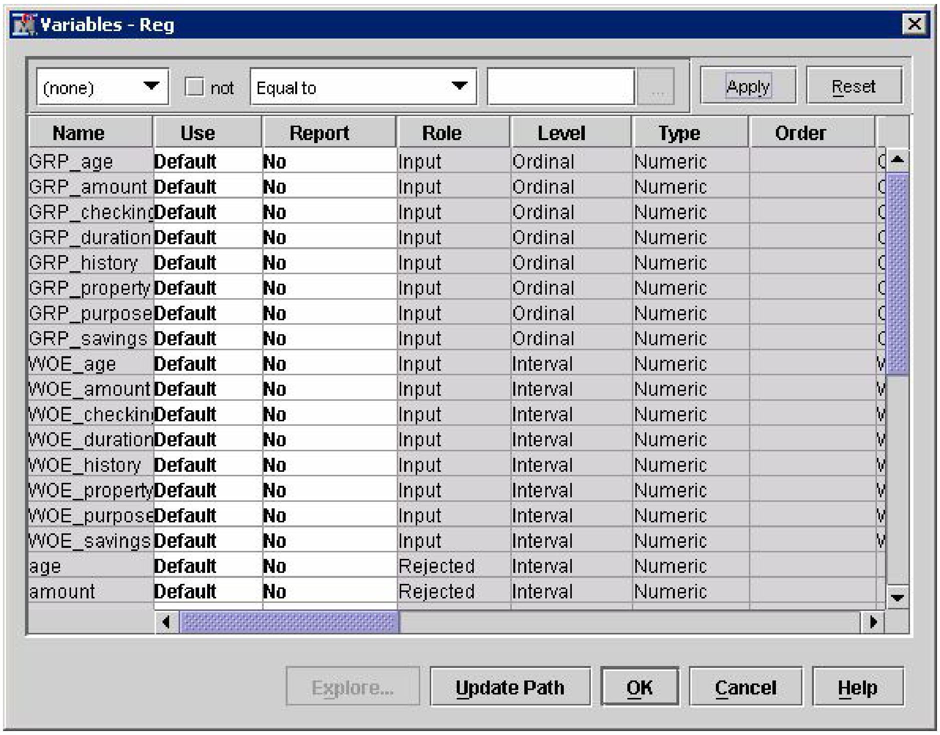
- Select **Browse** to open a window where you can browse the data set.
- Select **Explore** to open the Explore window, where you can sample and plot the data.
- Select **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

The output data sets from the **Interactive Grouping** node contain the original input variables and the following variables for each input variable:

- GRP_variable-name — the identification number of the group to which an observation belongs
- WOE_variable-name — the weight of evidence of an observation

The roles of all of the input and output variables that do not meet the variable selection criterion are set to Rejected.

To view the exported variables from your Interactive Grouping node, select the node that follows Interactive Grouping and click the ... button to the right of the Variables property. The input data to the successor node is the Interactive Grouping exported data.



Name	Use	Report	Role	Level	Type	Order	
GRP_age	Default	No	Input	Ordinal	Numeric		C
GRP_amount	Default	No	Input	Ordinal	Numeric		C
GRP_checking	Default	No	Input	Ordinal	Numeric		C
GRP_duration	Default	No	Input	Ordinal	Numeric		C
GRP_history	Default	No	Input	Ordinal	Numeric		C
GRP_property	Default	No	Input	Ordinal	Numeric		C
GRP_purpose	Default	No	Input	Ordinal	Numeric		C
GRP_savings	Default	No	Input	Ordinal	Numeric		C
WOE_age	Default	No	Input	Interval	Numeric		V
WOE_amount	Default	No	Input	Interval	Numeric		V
WOE_checkin	Default	No	Input	Interval	Numeric		V
WOE_duration	Default	No	Input	Interval	Numeric		V
WOE_history	Default	No	Input	Interval	Numeric		V
WOE_property	Default	No	Input	Interval	Numeric		V
WOE_purpose	Default	No	Input	Interval	Numeric		V
WOE_savings	Default	No	Input	Interval	Numeric		V
age	Default	No	Rejected	Interval	Numeric		
amount	Default	No	Rejected	Interval	Numeric		

Interactive Grouping Node Results

You can open the Results window of the **Interactive Grouping** node by right-clicking the node and selecting **Results** from the pop-up menu. For general information about the Results window, see "Using the Results Window" on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following information in the Results window:

- **Properties**
  - **Settings** — displays a window that has a read-only table of the Interactive Grouping node properties configuration when the node was last run.
  - **Run Status** — indicates the status of the Interactive Grouping node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — displays a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
  - **Notes** — displays the information entered by the user for the node.
- **SAS Results**
  - **Log** — the SAS log of the Interactive Grouping node run
  - **Output** — the SAS output of the Interactive Grouping node run

  - **Flow Code** — the SAS code used to produce the output that the **Interactive Grouping** node passes on to the next node in the process flow diagram

- **Scoring**

  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications. SAS Code contains Interactive Grouping values of input variables only. These values include the Interactive Grouping of unknown levels (if specified) and Interactive Grouping of specific levels. Score code is generated for only those variables that meet the variable selection criteria. If no Interactive Grouping values were generated, the **SAS Code** menu item is dimmed and unavailable.

  - **PMML Code** — the **Interactive Grouping** node does not generate PMML code.

- **Model**

  - **Output Variables** — the columns of the Output Variables table display Gini Statistic, Exported role, Level, Variable Labels, and Gini Ordering information for each of the input variables. Variables that have Gini Statistic of less than 20.0 are assigned the role "Rejected."

  - **Statistics Plot** — The Statistics Plot displays the value of Gini statistic for each of the input variables. Position your pointer over a vertical bar to see the variable name, Gini statistic, and exported role.

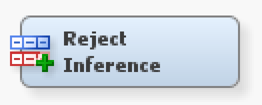
  - **Event Rate Plot** — The Event Rate Plot is a lattice plot that shows the event rates across group values for each of the input variables. Each lattice represents the group values for an input variable.

- **Table** — enables you to open a table of the data that is associated with the graph that you have open in the Results window. Data table column headings display variable labels by default. To display the variable names that are used in SAS code as column headings, right-click in any table cell and select **Show Variable Names**. To view labels in the column headings again, right-click in any table cell and select **Show Variable Labels**.

- **Plot** — opens the Graph wizard, enabling you to create ad hoc plots based on the data that is used to produce the table that is selected in the Results window. The Graph wizard menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Chapter 92
Reject Inference Node

Reject Inference Node



Overview of the Reject Inference Node

Overview

Note: Credit Scoring for SAS Enterprise Miner solution is not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner, the credit scoring node tools do not appear in your SAS Enterprise Miner software.

Credit scoring models are built with a fundamental bias (selection bias). The sample data that was used to develop a credit scoring model is structurally different from the "through-the-door" population to which the credit scoring model is applied. The non-event/event target variable that is created for the credit scoring model is based on the records of applicants who were all accepted for credit. However, the population to which the credit scoring model is applied is composed includes applicants who would have been rejected under the scoring rules that were used to generate the initial model.

One remedy for this selection bias is to use reject inference. The reject inference approach uses the model that was trained using the accepted applications to score the rejected applications. The observations in the rejected data set are classified as inferred non-event and inferred event using one of three available methods. The inferred observations are then added to the accepts data set, which contains the actual non-event and event records, to form an augmented data set. This augmented data set, which represents the "through-the-door" population, serves as the training data set for a second scorecard model.

When you create inferred data from the rejects data set, you predict how the rejected customer would have performed if they had been accepted. The Enterprise Miner Reject Inference node provides three different methods that you can use to create inferred data from the rejects data set: Fuzzy, Hard Cutoff, and Parceling.

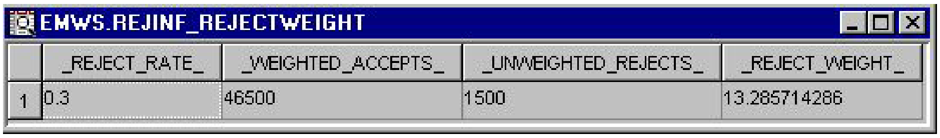
Once the rejects data has been scored and the observations have been classified as non-events or events, the rejects data is appended to the accepts data to form an augmented data set. If you over-sampled the events in the accepts data and generated frequency weights, those weights are used in the augmented data set for the accepts observations. If you did not use frequency weights for the accepts data, the frequency weight is set to 1 for the accepts observations in the augmented data set.

Fuzzy Method

The Fuzzy method is the default reject inference method. The Fuzzy method uses partial classifications of non-event and event to weight each reject observation. Instead of classifying observations as either non-event or event, the Reject Inference node creates two observations in the augmented data set for each original observation in the rejects data set. In the first observation, a target value of 0 is assigned. In the second observation, a target value of 1 is assigned. The two observations are then individually weighted by the posterior probabilities, P(non-event) and P(event), respectively. The posterior probabilities, P(non-event) and P(event), are estimated from the model that was trained on the accepts data set. Using these probabilities indicates the tendency for the original observation to be either a non-event or an event. A common frequency weight, called the reject weight, is then assigned to both observations to account for any over-sampling or under-sampling of the rejects data. The reject weight is computed as follows:

$$\text{reject weight} = \frac{\dfrac{\text{Rejection Rate}}{(1 - \text{Rejection Rate})}}{\dfrac{N_{rejects}}{N_{accepts}}}$$

In the preceding equation, $N_{accepts}$ is the weighted number of observations in the accepts data set. That is, it is the number of observations in the accepts data set after frequency weights have been applied. $N_{rejects}$ is the number of observations in the rejects data set; it is unweighted. The Rejection Rate is the value that you specify in the Reject Inference node's Properties panel and equals the prior probability of rejection. These values are stored in your project data in the EMWS.Rejectinf_rejectweight data set. For example, consider the following data set:



EMWS.REJINF_REJECTWEIGHT

	_REJECT_RATE_	_WEIGHTED_ACCEPTS_	_UNWEIGHTED_REJECTS_	_REJECT_WEIGHT_
1	0.3	46500	1500	13.285714286

The reject weight is computed as follows:

$$\_REJECT\_WEIGHT\_ = \frac{\dfrac{\_REJECT\_RATE\_}{(1 - \_REJECT\_RATE\_)}}{\dfrac{\_UNWEIGHTED\_REJECTS\_}{\_WEIGHTED\_ACCEPTS\_}} = \frac{\dfrac{0.3}{(1 - 0.3)}}{\dfrac{1500}{46500}} = 13.285714286$$

The posterior probabilities and the reject weights are combined to generate a frequency weight for each observation as follows:

$$frequency\ weight_{(target=0)} = reject\ weight * P(non - event)$$

$$frequency\ weight_{(target=1)} = reject\ weight * P(event)$$

The event rate increase is the value that is specified in the Event Rate Increase property. The default value for the event rate increase is 1.

Hard Cutoff Method

The Hard Cutoff method uses a cutoff score to classify observations as non-event or event. If you choose Hard Cutoff as your inference method, you must specify a value for the Cutoff Score property in the Reject Inference node's Hard Cutoff properties. In this context, the score refers to the number of scorecard points computed by the Scorecard node. Any score below the hard cutoff value is allocated a status of event. The Hard Cutoff method generates a frequency weight to account as follows:

$$reject\ weight = \frac{\dfrac{Rejection\ Rate}{(1 - Rejection\ Rate)}}{\dfrac{N_{rejects}}{N_{accepts}}}$$

Parceling Method

Parceling distributes scored rejects into equal-sized buckets that are defined by the Score Range method that you choose. The scored rejects within a bucket are then randomly classified as non-event or event. The proportion of observations that are classified as events is determined by the observed event rate in the accepts data multiplied by the value of the Event Rate Increase property. The default value for the Event Rate Increase property is 1. Thus, by default, the rejects are classified as non-events and events in the same proportion as what is observed in the accepts data. Setting the Event Rate Increase property to a value that is greater than 1 causes the event rate in the rejects data to be greater than what is observed in the accepts data.

Consider the following distribution of non-events and events in an accepts and rejects data set:

Score	# Event	# Non-event	% Event	% Non-event	Scored Rejects	Allocated Rejects Event	Allocated Rejects Non-event
0 — 99	24	10	70.5%	29.5%	342	241	101
100 — 199	54	196	21.6%	78.4%	654	141	513
200 — 299	43	331	11.5%	88.5%	345	40	305
300 — 399	32	510	5.9%	94.1%	471	28	443
400+	29	1232	2.3%	97.7%	778	18	780

The rejects data is scored with the model that was built on the accepts data and allocated to 5 bins. The table above was generated using an Event Rate Increase of 1.

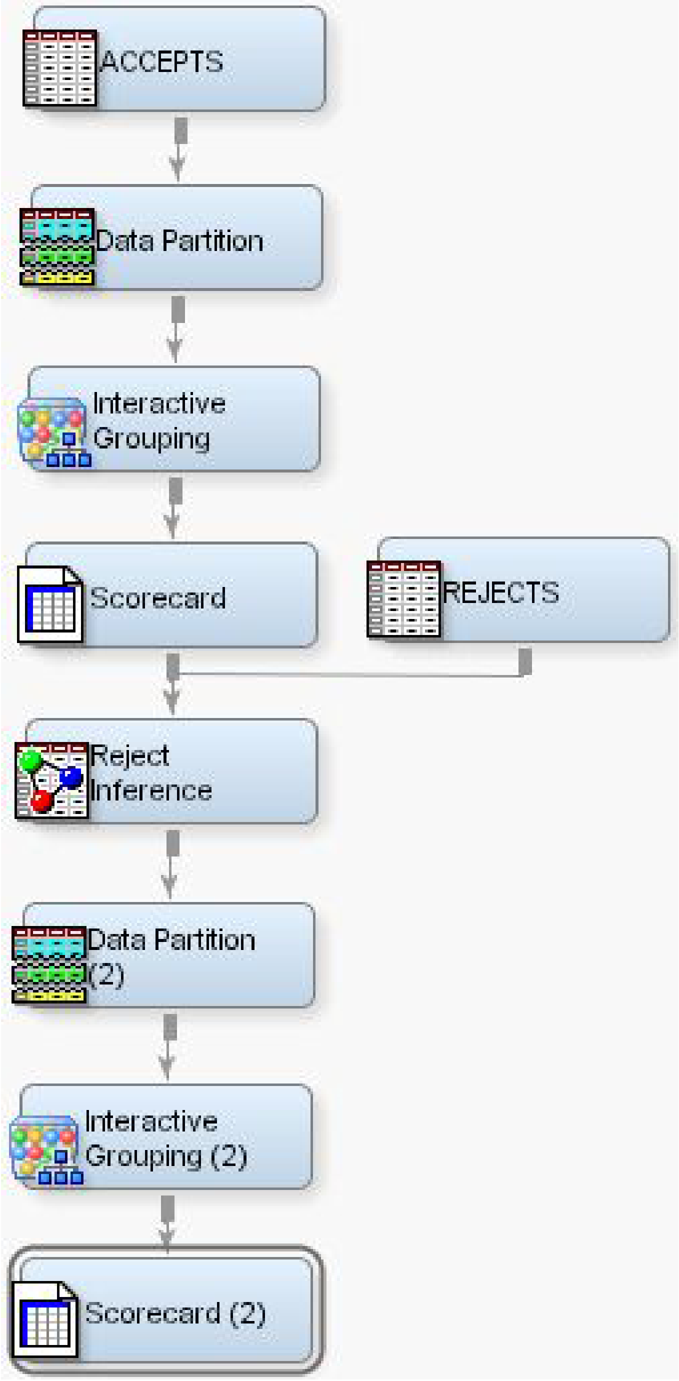
In the table above, consider the observations with Score = (0-99). Multiplying the % Event value (70.5%) by the number of Scored Rejects value (342) and using an Event Rate Increase of 1.0 results in an Allocated Rejects Event of 70.5 * 342 * 1.0 = 241. That is, of the 342 observations in the rejects data that have a score between 0 and 99, 241 are randomly selected and classified as events (assigned a target value of 1). The remaining 101 observations in that bin are classified as non-events (assigned a target value of 0). However, some practitioners assert that it is unrealistic for the proportion of non-events and events in the rejects data set to be the same as that in the accepts data set. They argue that rejects should have a larger proportion of events. To account for this, you can specify a higher Event Rate Increase value (1.2, for example) so that the event rate of the rejects is higher than what is observed in the accepts data.

The Parceling method computes a frequency weight as follows:

$$reject\ weight = \frac{\dfrac{Rejection\ Rate}{(1 - Rejection\ Rate)}}{\dfrac{N_{rejects}}{N_{accepts}}}$$

Data Set Requirements for the Reject Inference Node

The Reject Inference node requires a scored accepts data set that has been output by the Scorecard node and a scored rejects data set. These data sets must have their Role set to **Score**. Use the Input Data Source node to flow your rejects data set to the Reject Inference node as score data. A typical credit scoring process flow diagram that uses a rejects data set is shown below:



Reject Inference Node Properties

Reject Inference Node General Properties

The following general properties are associated with the Reject Inference Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Reject Inference window. The Imported Data — Reject Inference window contains a list of the ports that provide data sources to the Reject Inference node. Click the … button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Reject Inference window. The Exported Data — Reject Inference window contains a list of the output data ports that the Reject Inference node creates data for when it runs. Click the … button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the … button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Reject Inference Node Train Properties

The following train properties are associated with the Reject inference node:

- **Variables** — Use the Variables property of the Scorecard node to specify the properties of each variable that you want to use in your data source. Click the … button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, and you can select which variables you want included in reports.

Reject Inference Node Train Properties: General

- **Inference Method** — Use the Inference Method property to specify the method that you want to use to classify rejects data set observations. The three choices are **Fuzzy**, **Hard Cutoff**, and **Parceling**.
- **Rejection Rate** — The Rejection Rate represents the probability of rejection in the population. The Reject Inference node uses the Rejection Rate property to generate a sampling weight that is used to account for oversampling of the rejects. Rejection rates are real numbers between 0 and 1, exclusive. The default rejection rate is 0.3.
- **Event Rate Increase** — The Event Rate Increase is a scaling entity that differs according to the selected Inference Method. When the Inference Method is set to Hard Cutoff, Event Rate Increase is dimmed and unavailable. When the Inference Method is set to Fuzzy, the Event Rate increase allows the user to increase the

weighting value used with rejected applicants. When the Inference Method is set to Parceling, the Event Rate increase specifies the adjustment value for the reject event rate. When using Parceling, the observed event rate of the accepts data is multiplied by the value of the Event Rate Increase property to determine the event rate for the rejects data. For example, to exhibit a reject rate that is 20% greater than what is observed in the accepts data, set the Event Rate Increase property to 1.2. Permissible values for the Event Rate Increase property are real numbers from 0 to 100. The default setting for the Event Rate Increase property is 1.0.

Reject Inference Node Train Properties: Hard Cutoff

The Reject Inference node uses the Cutoff Score property when the selected inference method is Hard Cutoff.

- **Cutoff Score** — The Hard Cutoff reject inference method compares the scored observations from the rejects data set to the Cutoff Score value to classify observations as non-event or event. Scores below the Cutoff Score value are assigned a status of event; otherwise, the observation is classified as non-event. The Cutoff Score threshold is an integer between -1,000,000 and 1,000,000. The default setting for the Cutoff Score property is 200.

Reject Inference Node Train Properties: Parceling

You must configure Parceling properties when you choose Parceling as the selected inference method. You must also specify a Rejection Rate value in the General properties when you are parceling.

- **Score Range Method** — Use the Score Range Method property to specify the way that you want to define the range of scores to be bucketed.
  - **Accepts** — The Accepts score range method distributes the rejects into equal-sized buckets based on the score range of the Accepts data set. The Accepts method requires a scored accepts data set. The Accepts method also requires you to provide values for the Score Buckets and Event Rate Increase property settings. Accepts is the default setting for the Score Range Method property.
  - **Rejects** — The Rejects score range method distributes the rejects into equal-sized buckets based on the score range of the rejects data set. The Rejects method requires you to also specify values for the Score Buckets and Event Rate Increase property settings.
  - **Scorecard** — The Scorecard score range method distributes the rejects into equal-sized buckets based on the score range that is output by the augmented data set. The Scorecard method requires you to also specify values for the Score Buckets and Event Rate Increase property settings.
  - **Manual** — The Manual score range method distributes the rejects into equal-sized buckets based on the range that you input. When you choose Manual as your Score Range Method, you must also provide values for the Max Score and Min Score properties to specify your manual range parameters.
- **Min Score** — When you choose the Manual score range method, use the Min Score property to specify the lower parameter of your manual score range. Permissible Min Score property values are integers between -1,000,000 and 1,000,000. The default setting for the Min Score parceling property is 0.
- **Max Score** — When you choose the Manual score range method, use the Max Score property to specify the upper parameter of your manual score range. Permissible Max Score property values are integers between -1,000,000 and 1,000,000. The default setting for the Max Score parceling property is 250.

- **Score Buckets** — Use the Score Buckets property to specify the number of buckets that you want to use to parcel the data set into during non-event and event classification. Permissible Score Buckets property values are integers between 1 and 100. The default setting for the Score Buckets property is 25.
- **Random Seed** — Use the Random Seed property to specify the initial Random Seed value that you want to use for random number generation. The default Random Seed value is 12345.

Reject Inference Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Reject Inference Node Results

After the Reject Inference node successfully runs, you can open the Results - Reject Inference window by right-clicking the node in the Diagram Workspace and selecting **Results**. For general information about the Results window, see "Using the Results Window" on page 247 in the Enterprise Miner Help.

Select **View** from the main menu to view the following information in the Results window:

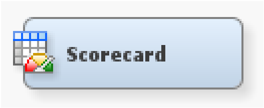
- **Properties**
  - **Settings** — displays a window with a read-only table of the Segment Profile node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
  - **Run Status** — indicates the status of the Segment Profile node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — the code that Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Segment Profile node run.

  - **Output** — the SAS output of the Segment Profile node run. The output displays the summary statistics for the original partitioned data set and for each resulting partition.
  - **Flow Code** — the SAS code used to produce the output that the Segment Profile node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications.
  - **PMML Code** — the Reject Inference node does not generate PMML code.
- **Model**
  - **Classification Chart** — displays the classification charts for the accepts and inferred data sets.
  - **Interval Variables Plots** — displays the plots of the scorecard points, and the p(good) and p(bad) values for inferred target variables for the accepts and augmented data sets.
- **Tables** — displays the data table for a graph that you select.
- **Plot** — opens the Graph Wizard for the table that you select.

Chapter 93

Scorecard Node

Scorecard Node



Overview of the Scorecard Node

Overview

Note: Credit Scoring for SAS Enterprise Miner is not included with the base version of SAS Enterprise Miner. If your site has not licensed Credit Scoring for SAS Enterprise Miner, the credit scoring node tools do not appear in your SAS Enterprise Miner software.

Credit scores are designed to reflect the odds of an applicant being a "good" credit risk versus being a "bad" credit risk, however "good" and "bad" are defined. The Scorecard node computes credit scores in a two-step process.

In the first step, the Scorecard node fits a logistic regression model, which estimates the ln(odds) as a linear function of the characteristics. The Scorecard node uses the data set that is exported by the Interactive Grouping node as the model's input data set. You can choose to use either the characteristics' Weight Of Evidence variables or the group variables as inputs for the logistic regression model. The node's Model Selection properties enable you to choose from a variety of model selection methods and selection criteria.

In the second step, the Scorecard node applies a linear transformation to the predicted ln(odds) to compute a score for each attribute of each characteristic. The score for an attribute of a characteristic satisfies the following equation: score = ln(odds) * factor + offset.

The node's Scaling Options properties enable you to control the factor and offset values of the score equation. Scaling enables you to control the range of the scores as well as the rate of change in odds for a given increase in the score. The Points to Double Odds property determines the value of factor. The Scorecard Points and Odds properties jointly determine the value of offset. To understand how this is accomplished, consider what happens to the score when you double the odds. The score equation must satisfy the following: score + Points to Double Odds = ln(2*odds) * factor + offset.

Subtracting the original score equation and solving for Points to Double Odds yields this result: Points to Double Odds = factor * ln(2).

Solving for factor yields this result: factor = Points to Double Odds / ln(2).

Thus, when you set the value of the Points to Double property, you are, in fact, controlling the value of factor in the score equation. The default value for Points to Double Odds is 20. This value is interpreted to mean that a 20 point increase in an applicant's score means the odds of the applicant being a "good" risk is doubled.

If you substitute the result for factor into the original score equation and rearrange to isolate offset, you get the following result: offset = score - (Points to Double Odds / ln(2)) * ln(odds).

To solve for offset, you need one fixed pair of values for score and odds. You use the Scorecard Points and Odds properties to specify these two values. Rewriting the offset equation using the property names you get the following result: offset = Scorecard Points - (Points to Double Odds / ln(2)) * ln(Odds).

The values you choose for Scorecard Points and Odds are arbitrary. The default values are 200 and 50, respectively. These values are interpreted to mean that a score of 200 represents odds of 50 to1 (that is, when P("good")/P("bad") = 50, the score = 200).

Because the logistic regression models the ln(odds) as a linear function of the characteristics, it is easy to see each characteristic's contribution to the ln(odds), and thus, the score. The weight of evidence (WOE) variables enter the model as interval variables and the group variables enter the model as ordinal variables. Thus, when you use the WOE variables as inputs for the regression model, the score points for each characteristic i are calculated as follows:

$$-(woe_i * \beta_i + \frac{\alpha}{n}) * factor + \frac{offset}{n}$$

where α is the intercept from the logistic regression, the βi is the parameter estimate associated with the ith characteristic, and n is the number of characteristics. An observation's total score is the sum of the score points across all n characteristics.

When group variables are used as inputs into the logistic regression model, the ordinal group variables are automatically replaced with a set of binary indicator variables by the DMREG procedure. Thus, when you use the group variables as inputs for the regression model, the score points for each attribute is calculated as follows:

$$-(\beta_{ij} + \frac{\alpha}{n}) * factor + \frac{offset}{n}$$

where α is the intercept from the logistic regression, βij is the parameter estimate associated with the jth attribute of the ith characteristic, and n is the number of characteristics. When a characteristic has k attributes and k > 2, the DMREG procedure generates k-1 binary indicator variables to represent the characteristic's ordinal group variable. One indicator variable must be omitted to prevent collinearity. Therefore, one attribute of such a characteristic does not have a regression coefficient associated with it. In such cases, the coefficient is assumed to be 0 in the preceding equation. An observation's total score is the sum of the score points across all n characteristics.

Prior Probabilities and the Scorecard Node

If you enter prior probabilities in the accepts data source's decisions matrix, the parameter estimates of the Scorecard node's logistic regression model are not affected by the prior probabilities. The estimated coefficients of the characteristics are consistent but the intercept is biased. A prior offset is applied to correct this bias in the intercept. The prior offset is defined as follows:

$$prior\ offset = ln\left(\frac{\rho_1\pi_0}{\rho_0\pi_1}\right)$$

In the preceding equation, ρ1 and ρ0 are the proportion of target classes in the sample, and π1 and π0 are the proportion of target classes in the population. When WOE variables are used as inputs in a scorecard model, the scorecard point value for a characteristic's attribute is computed as follows:

$$scorecard\ points_{ij} = -\left(WOE_{ij} * \beta_j + \frac{(\alpha - prior\_offset)}{n}\right) * factor + \frac{offset}{n}$$

In the preceding equation, WOEij is the Weight of Evidence of the ith attribute of the jth characteristic, βj is the regression coefficient on the jth characteristic, α is the intercept of the logistic regression model, and n is the number of characteristics in the logistic regression model. When group variables are used as the inputs in a scorecard model, the scorecard point value for an attribute of a characteristic is computed as follows:

$$scorecard\ points_{ij} = -\left(\beta_{ij} + \frac{(\alpha - prior\_offset)}{n}\right) * factor + \frac{offset}{n}$$

For more details about the use of prior probabilities in SAS Enterprise Miner see the Prior Probabilities on page 166 in the Predictive Modeling documentation.

After the scorecard points have been computed, the Scorecard node generates a tabular representation of the scorecard model in an .html format. The node also generates numerous diagnostic tables and graphs.

If a Scorecard node is used in a process flow diagram that specifies prior probabilities, then the Scorecard node Gains Table will contain the following adjusted variables:

- _adj_non_event_count_
- _adj_cumulative_non_event_count_
- _adj_count_
- _adj_cumulative_count_
- _adj_marginal_event_rate_

- _adj_marginal_non_event_rate_
- _adj_cumulative_event_rate_
- _adj_cumulative_non_event_rate_
- _adj_population_count_
- _adj_approval_rate_
- _adj_average_marginal_profit_
- _adj_average_total_profit_
- _adj_population_percentage_

The adjusted variables, listed above, are calculated as follows:

- First, an offset is computed with the following information:
  - TRAINPRIOR — the prior probability value that is proportional to the data.
  - DECPRIOR — the adjusted probability value that you specified.
  - E_ — prefix that indicates an event occurred.
  - NE_ — prefix that indicates a nonevent occurred.

  This information is used to compute the offset as OFFSET = (E_TRAINPRIOR*NE_DECPRIOR)/(NE_TRAINPRIOR*E_DECPRIOR)
- Next, the offset is used to adjust the non-event count so that _adj_non_even_count_ = _non_event_count_ * OFFSET.
- Finally, the adjusted non-event count is used during any of the adjusted calculations that are displayed in the gains table. For instance, the adjusted count is calculated as _adj_count_ = _event_count_ + _adj_non_event_count_.

Data Set Requirements for the Scorecard Node

The Scorecard node requires a data source with a binary target variable. There must also be an "Interactive Grouping Node" on page 1485 preceding the Scorecard node in your process flow diagram. The WOE variables or the group variables that are generated by an Interactive Grouping node are used as inputs for the Scorecard node's logistic regression model.

Scorecard Node Properties

Scorecard Node General Properties

The following general properties are associated with the Scorecard Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Scorecard node added to a diagram will have a Node ID of Scorecard. The second Scorecard node added to a diagram will have a Node ID of Scorecard2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Scorecard window. The Imported Data — Scorecard window contains a list of the ports that provide data sources to the Scorecard node. Click the ... button to the right of the Imported Data property to open a table of the imported data.

  If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — The Exported Data property provides access to the Exported Data — Scorecard window. The Exported Data — Scorecard window contains a list of the output data ports that the Scorecard node creates data for when it runs. Click the ... button to the right of the Exported Data property to open a table that lists the exported data sets.

  If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

  - **Browse** to open a window where you can browse the data set.
  - **Explore** to open the Explore window, where you can sample and plot the data.
  - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the ... button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Scorecard Node Train Properties

The following train properties are associated with the Scorecard node:

- **Variables** — Use the Variables property of the Scorecard node to specify the properties of each variable that you want to use in your data source. Click the ... button to the right of the Variables property to open a variables table. You can set the variable status to either Use or Don't Use in the table, and you can select which variables you want included in reports.
- **Scorecard Points** — Click the ... button to the right of the Scorecard Points property to open the Scorecard Editor window. You can manually overwrite the generated scorecard points for each grouping in the "Scorecard Editor Window" on page 1531 .
- **Score Ranges** — Click the ... button to the right of the Score Ranges property to open the Score Ranges window. In the Score Ranges window, you can modify the upper limit of the score range for each score bucket. You must run the Scorecard node before opening the "Score Ranges Window" on page 1532 .
- **Analysis Variables** — Use the Analysis Variables property to specify the type of variables that are used to build the logistic regression model.

  Choose one of the following types:

  - **WOE** — the WOE variables that the Interactive Grouping node generates. The WOE variables enter the node's logistic regression model as interval inputs.
  - **Group** — the Group variables that the Interactive Grouping node generates. The group variables enter the node's logistic regression model as ordinal input variables.

- **Freeze Scorecard Points** — Use the Freeze Scorecard Points property to specify if and how frozen scorecard points should be applied.
  - **None** — scorecard points are calculated for all variables.
  - **Frozen Group** — scorecard points are retained for variables that were created from frozen grouping definitions. Scorecard points are calculated for all other variables.
  - **All** — scorecard points for all variables will be retained regardless of their grouping definitions.

Scorecard Node Train Properties: Publish Score Code

- **Output Variables** — Use the Output Variables property to specify the variables that are generated by the publish score code.
  - **Scorepoints** — The publish score code generates the scorecard points.
  - **Scorepoints and Posterior Probabilities** — The publish score code generates the scorecard points and the posterior probabilities from the regression model (that is, P_ variables).
  - **Complete** — The publish score code includes the publish score code from the regression model and the computation of the scorecard points.

Scorecard Node Train Properties: Scaling Options

- **Intercept Based Scorecard** — Set the Intercept Based Scorecard property to **Yes** to generate an intercept based scorecard. The default value is **No**.

  An intercept-based scorecard creates an initial scorecard that is identical to the normal scorecard, with 0 points assigned to the intercept term. After building the initial scorecard, the **Scorecard** node parses each variable to determine the least points assigned for that variable. Next, all values within a variable are adjusted so that the lowest value is zero; the intercept is adjusted by an equal amount.

  For example, consider a data set with the two variables, Variable A and Variable B. Assume that Variable A has the scorecard values as given below. Remember that the default scorecard has 0 points assigned to the intercept

Group	Scorecard Points
1	150
2	200
3	275
4	325

The smallest value associated with Variable A is 150, so 150 points are subtracted from the value of each group in Variable A and 150 points is added to the intercept term. The new scorecard for Variable A is given in the table below, with an updated intercept value.

Group	Scorecard Points
1	0
2	50
3	125
4	175
Intercept: 150	

Now, let Variable B have the scorecard given below. Note that Variable B starts with an intercept value of 150, carried over from Variable A.

Group	Scorecard Points
1	–50
2	–25
3	75

The smallest value associated with Variable B is –50, so 50 points are added to the value of each group in Variable B and 50 points are subtracted from the intercept term. The new scorecard for Variable B is given in the table below, with an updated intercept value.

Group	Scorecard Points
1	0
2	25
3	125
Intercept: 100	

The end result is that each variable in the scorecard has a reference value of 0 and contains only positive values.

- **Reverse Scorecard** — When you set the Reverse Scorecard property to **Yes**, the scorecard points increase as the proportion of good applicants increases. Typically, scorecards that are built for credit scoring are decreasing as the event rate increases. This makes sense when you want to model a bad customer because you want to see an increased risk (lower score) associated with a higher proportion of bad applicants. However, you can use the Scorecard node to model good applicants, in which case you want to see scorecard points increase as the event rate increases. Set the Reverse Scorecard property to **No** in order to sort the scorecard points in descending order as the event rate increases.

- **Odds** — Use the Odds property to specify the non-event to event odds that correspond to the score value that you specify in the Scorecard Points property. For example, assume that your input data has about 30 good customers for every bad customer and you want to model the bad customers. In this case, you should set the Reverse Scorecard property to **No** and the Odds property to **30**. Using the same example data, if you wanted to model the good customers, you would set Reverse Scorecard to **Yes** and the Odds property to **0.033**, which is 1/30. The default value is 50 and valid values are positive, real numbers.

- **Scorecard Points** — Use the Scorecard Points property to specify a specific score that is associated with the odds that are specified in the Odds property. For example, if you use the default values of 200 for the Scorecard points and 50 for the Odds, a score of 200 represents odds of 50 to1 (that is, P(non-event)/P(event) = 50).

- **Points to Double Odds** — Use the Points to Double Odds property to specify the increase in score points that results in the score that corresponds to twice the odds. The Points to Double Odds property accepts integers greater than or equal to 1. The default value is 20.

- **Scorecard Type** — Use the Scorecard Type property to specify the type of scorecard to generate: either **Summary** or **Detailed**. A summary scorecard displays the attribute names and the scorecard points for each attribute. A detailed scorecard contains additional information such as group number, weight of evidence, percentage of population, and the regression coefficient associated with each attribute. The default setting for the Scorecard Type property is **Summary**.

- **Precision** — Use the Precision property to specify how many decimal places are used to calculate the scorecard points of an attribute. Permissible values are integers between 0 and 4. The default setting for the Precision property is 0, which produces integer scorecard points.

- **Bucketing Method** — The Bucketing Method property specifies the method that is used for creating the buckets of scorecard points. The bucketing method will affect the appearance of plots and tables and in the Scorecard node results window. The methods are **Min/Max Distribution** and **Quantiles**.

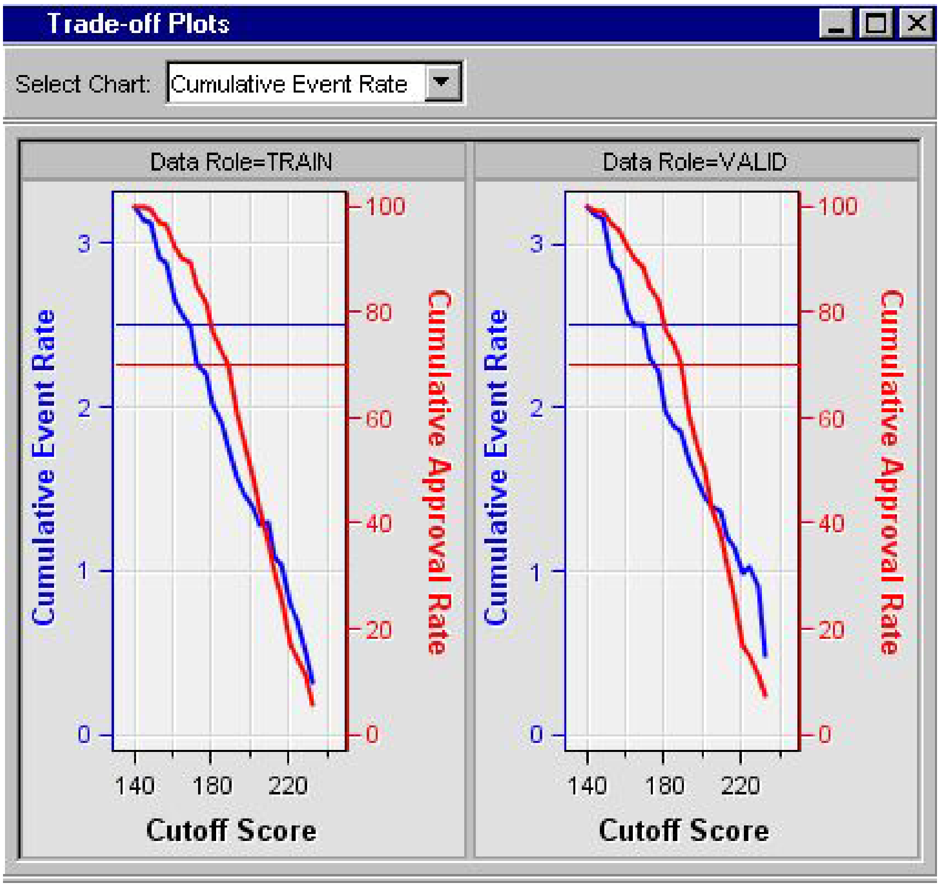
- **Number of Buckets** — Use the Number of Buckets property to specify the number of bins into which the score range is divided in a Gains table or Trade-Off chart. The default value is 25.

- **Use Indeterminate Values** — Specifies whether indeterminate values should be included in the Gains table calculations.

- **Revenue Accepted Good** — Use the Revenue Accepted Good property to specify the assumed average revenue from accepting applicants who have good credit status. This value is used to provide vertical reference lines in a trade-off plot in the node's results. The default value is 1,000.

- **Cost Accepted Bad** — Use the Cost Accepted Bad property to specify the assumed average cost of accepting applicants who have bad credit status. This value is used to provide vertical reference lines in a trade-off plot in the node's results. The default value is 50,000.

- **Current Approval Rate** — Use the Current Approval Rate property to specify the percentage of applicants who are approved. The default value is 70.0. The value that you specify is used to generate a vertical reference line (a VREF in SAS/GRAPH terminology) in the Trade-off plot depicted below. A vertical reference line is a horizontal line that intersects the vertical axis of a plot at a specified value.



- **Current Event Rate** — Use the Current Event Rate property to specify the percentage of applicants who have bad credit status. The default value is 2.5. The value that you specify is used to generate a vertical reference line (a VREF in SAS/GRAPH terminology) in the Trade-off plot depicted above. A vertical reference line is a horizontal line that intersects the vertical axis of a plot at a specified value.
- **Generate Characteristic Analysis** — Use the Generate Characteristic Analysis property to specify whether characteristic analysis tables are to be created. When the Generate Characteristic Analysis property is set to Yes, Attribute Event Rate and Average Score, Attribute Proportion by Score, and Cumulative Event Rate by Score tables are created for each characteristic in the node's output. The default value is No.

Scorecard Node Train Properties: Adverse Characteristic Options

It is required by law to explain why an application is rejected. The Adverse Characteristics properties enable you to prepare an explanation by comparing the actual value of a variable to the weighted average score or neutral score in order to identify the characteristics that are deemed adverse. For example, the following table lists the weighted average score and actual value of variables of an applicant who was rejected. The characteristics are listed based on the values of difference in ascending order. In this example, the characteristics that are deemed adverse in the order from most severe to least severe are OWN_RENT, INCOME, AGE, and then EDUCATION.

Characteristics (Variables)	Weighted Average Score	Actual Value	Difference
OWN_RENT	64	57	–7

Characteristics (Variables)	Weighted Average Score	Actual Value	Difference
INCOME	54	52	–2
AGE	33	35	2
EDUCATION	59	65	6

- **Method** — Use the Method property to specify which method is used to identify adverse characteristics. Choose from the two methods below.
  - **Weighted Average Score** — the weighted average of the scorecard points of observations in the groups. The number of observations in a group corresponds to the weight of a group. For example, suppose that there are three groups of the variable AGE, and the average score points of the groups are 36, 42, and 45, with 20, 30, and 40 observations in each of the groups, respectively. The weighted average score of AGE is (36*20+42*30+45*40)/90=42.

    When the Analysis Variables property is set to Group, the Method property is set to Weighted Average Score by default and the Neutral Score method is unavailable.
  - **Neutral Score** — A neutral score is defined as being the score points for an attribute when the attribute's WOE is equal to 0. Thus, the formula for computing a neutral score is as follows:

    $$-\left(\frac{\alpha}{n}\right) * factor + \frac{offset}{n}$$

    An attribute that has a WOE of 0 implies that a person with that attribute has the same odds as the population as a whole. An applicant whose attributes are all neutral has a total score of n * neutral score, where n is the number of characteristics in the scorecard. If you substitute this total score into the score formula, you can derive the implied ln(odds) for the applicant. The implied ln(odds) for a neutral total score are approximately equal to the ln(odds) of the population as a whole. Neutral Score is the default setting for the Method property when the Analysis Variables property is set to WOE. When the Analysis Variables property is set to Group, the Method property is set to Weighted Average Score by default and the Neutral Score method is unavailable.
- **Display Value** — Use the Display Value property to specify whether the Adverse Characteristic value should be displayed in the Scorecard node results. The default setting of the Display Value property is **No**.
- **Generate Report** — Set the Generate Report property to **Yes** to create a report for the top three adverse characteristics.
- **Number of Characteristics** — specifies the number of adverse characteristics that are included in the generated report. The **Number of Characteristics** property is available only when the **Generate Report** property is set to **Yes**.
- **Adverse Variables** — Click the ... button to the right of the **Adverse Variables** property to open the Adverse Variables window. The Adverse Variables window enables you to identify whether a variable is included in the adverse variable

calculations and reports. If you want to exclude a variable, set the value of the **Use** column to **No**.

The **Adverse Variables** property is available only when the **Generate Report** property is set to **Yes**.

Scorecard Node Train Properties: Model Selection Options

- **Selection Model** — Use the Selection Model property to specify the model selection method that you want to use during training.

  You can choose from the following selection methods:

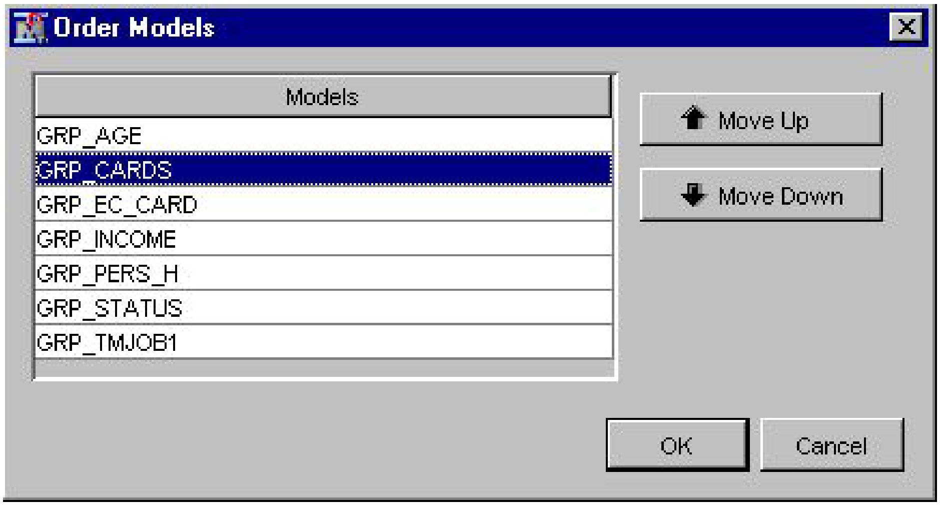
  - **None** — (default setting) all inputs are used to fit the model.
  - **Backward** — begins with all candidate effects in the model and removes effects that do not satisfy the Stay Significance Level criterion.
  - **Forward** — begins with no candidate effects in the model and adds effects that satisfy the Entry Significance Level criterion.
  - **Stepwise** — begins as in the forward model but might remove effects already in the model. Continues until Stay Significance Level or stepwise stopping criteria are met. For more information, see Model Selection Methods on page 928 in the Regression node documentation.

- **Criterion** — If you choose the forward, backward, or stepwise selection method for the Selection Model property, the Selection Criterion property specifies the criterion used to select the final model.

  The available criteria are as follows:

  - **Default** — uses the None criterion.
  - **None** — chooses the standard variable selection based on the entry or stay p-values.
  - **Akaike's Information Criterion** — The model with the smallest criterion value is chosen.
  - **Schwarz's Bayesian Criterion** — The model with the smallest criterion value is chosen.
  - **Validation Error** — the error rate for the validation data set. The error is the negative log-likelihood for logistic regression. The model with the smallest error rate is chosen.
  - **Validation Misclassification** — the misclassification rate for the validation data set. The model with the smallest misclassification rate is chosen.
  - **Cross-Validation Error** — the error rate for cross-validation. The error is the negative log-likelihood for logistic regression. The model with the smallest error rate is chosen.
  - **Cross-Validation Misclassification** — the misclassification rate for cross-validation. The model with the smallest misclassification rate is chosen.

- **Model Ordering** — Click the ... button to the right of the Model Ordering property to open the Order Models window:



Order Models

Models
GRP_AGE
GRP_CARDS
GRP_EC_CARD
GRP_INCOME
GRP_PERS_H
GRP_STATUS
GRP_TMJOB1

The Order Models window enables you to manually control the order in which variables enter the model. The Use Selection Defaults property must be set to No before you can use the Model Ordering property.

- **Use Selection Default** — Set the Selection Default property to **No** to use non-default values for your model selection criteria. You must also set the corresponding individual selection criteria properties to your user-defined values. The default setting for the Selection Default property is **Yes**.
- **Entry Significance Level** — When the Selection Default property is set to **No**, the Entry Significance Level property specifies the significance level to add variables in forward or stepwise regressions. The default value for the Entry Significance Level is 0.05. Valid values should be greater than 0, but less than or equal to 1.
- **Stay Significance Level** — When the Selection Default property is set to **No**, the Stay Significance Level property specifies the significance level to remove variables during backward or stepwise regression. The default value for the Stay Significance Level is 0.05. Valid values should be greater than 0, but less than or equal to 1.
- **Start Variable Number** — When the Selection Default property is set to **No**, the Start Variable Number specifies the number of variables to use in the first model. The forward and stepwise selection methods use default Start values of 0. The backward selection method uses all variables in the first model by default.
- **Stop Variable Number** — For the forward selection method, the Stop Variable Number specifies the maximum number of variables to appear in the final model. The default for the forward method is the total number of input variables. For the backward selection method, the Stop Variable Number specifies the minimum number of variables to appear in the final model. The default value for the backward method is 0. Other criteria can be applied and the node can terminate the variable selection before the Stop Variable Number criterion is satisfied. The Stop Variable Number is not applicable when the stepwise method is used. The Use Selection Defaults property must be set to **No** in order to enable the Stop Variable Number property.
- **Force Candidate Effects** — When you use the Model Ordering property to manually order variables for entry into the model, you can use the Force Candidate Effects property to specify a number of variables (also known as candidate effects) that are forced to be included in the model. For example, if you specify the number 2, the first two variables listed in the Order Models window of the Model Ordering property are forced to be included in the model.

- **Maximum Number of Steps** — If the Selection Default property is set to **No**, the Maximum Number of Steps property specifies the maximum number of steps allowed during the STEPWISE model effect selection process. The default setting for Maximum Number of Steps is 0.

Scorecard Node Status Properties

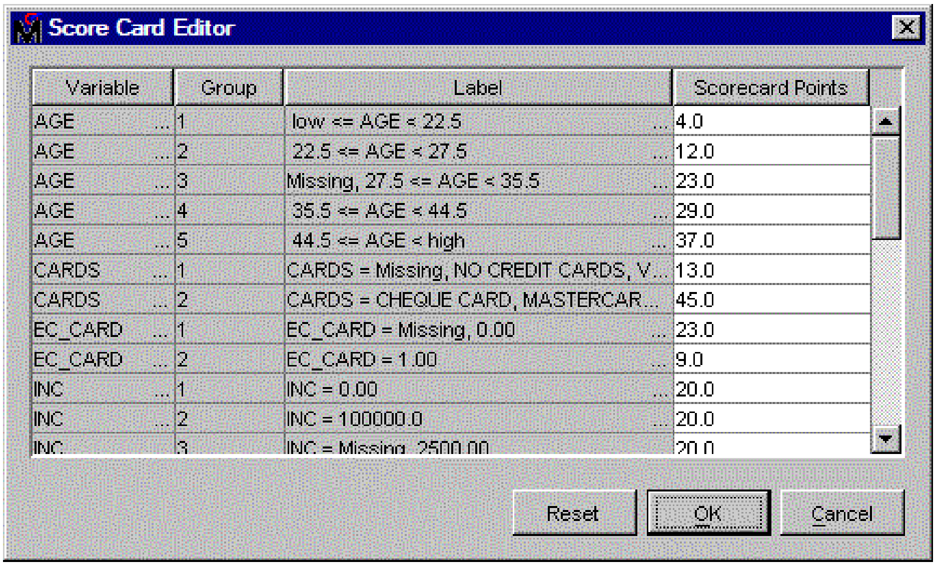
The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Scorecard Editor Window

The Scorecard Editor window enables you to manually overwrite the scorecard points that the Scorecard node generates and assign values of your own. To open the Scorecard Editor window, ensure that the Scorecard node is selected in the Diagram Workspace.

Then, click the ... button to the right of the Scorecard Points property in the node Properties panel to open the Scorecard Editor window.



Variable	Group	Label	Scorecard Points
AGE	1	low <= AGE < 22.5	4.0
AGE	2	22.5 <= AGE < 27.5	12.0
AGE	3	Missing, 27.5 <= AGE < 35.5	23.0
AGE	4	35.5 <= AGE < 44.5	29.0
AGE	5	44.5 <= AGE < high	37.0
CARDS	1	CARDS = Missing, NO CREDIT CARDS, V...	13.0
CARDS	2	CARDS = CHEQUE CARD, MASTERCAR...	45.0
EC_CARD	1	EC_CARD = Missing, 0.00	23.0
EC_CARD	2	EC_CARD = 1.00	9.0
INC	1	INC = 0.00	20.0
INC	2	INC = 100000.0	20.0
INC	3	INC = Missing, 2500.00	20.0

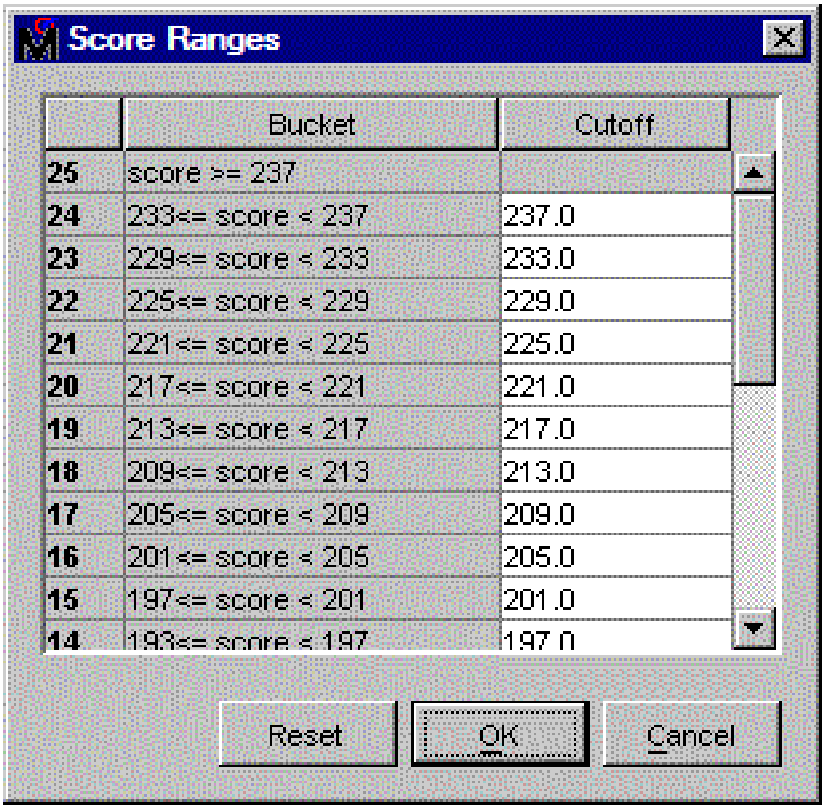
The Scorecard Editor window has the following columns:

- **Variable** — displays the name of the variable (characteristic).
- **Group** — displays the group number of the attribute.
- **Label** — displays the label of the attribute.

- **Scorecard Points** — displays the scorecard points that are generated when you run the Scorecard node. You can change the values in this column by entering numbers in the column. You should not enter missing values (.) for all the groups of a variable.

Score Ranges Window

The Score Ranges window enables you to modify the upper limit of the score range for each score bucket. Score buckets are used in the plots and tables that appear in the Scorecard node's results. To open the Score Ranges window, ensure that the Scorecard node is selected in the Diagram Workspace. Then, click the ... button to the right of the Score Ranges property in the node Properties panel to open the Score Ranges window.



	Bucket	Cutoff
25	score >= 237	
24	233<= score < 237	237.0
23	229<= score < 233	233.0
22	225<= score < 229	229.0
21	221<= score < 225	225.0
20	217<= score < 221	221.0
19	213<= score < 217	217.0
18	209<= score < 213	213.0
17	205<= score < 209	209.0
16	201<= score < 205	205.0
15	197<= score < 201	201.0
14	193<= score < 197	197.0

Scorecard Node Results

You can open the Results window of the Scorecard node by right-clicking the node and selecting **Results**. For general information about the Results window, see “Using the Results Window” on page 247 in the SAS Enterprise Miner Help.

Select **View** from the main menu to view the following results in the Scorecard Results window:

- **Properties**
  - **Settings** — displays a window with a read-only table of the Scorecard node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.

  - **Run Status** — indicates the status of the Scorecard node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
  - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
  - **Train Code** — the code that SAS Enterprise Miner used to train the node.
  - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
  - **Log** — the SAS log of the Scorecard node run.
  - **Output** — the SAS output of the Scorecard node run. The Scorecard SAS output includes the following:
    - variables summary
    - model events summary
    - a list of predicted and decision variables
    - PROC DMREG model information
    - target profile information
    - PROC DMREG parameter estimates
    - optimization results
    - analysis of maximum likelihood estimates
    - odds ratio estimates
    - scorecards
    - fit statistics
    - classification tables
    - event classification tables
    - assessment score rankings
    - an assessment score distribution
  - **Flow Code** — the SAS code used to produce the output that the Scorecard node passes on to the next node in the process flow diagram.
- **Scoring**
  - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the SAS Enterprise Miner environment in custom user applications.
  - **PMML Code** — the PMML Code on page 55 that was generated by the node. The PMML Code menu item is dimmed and unavailable unless PMML is enabled. on page 55
- **Assessment**
  - **Fit Statistics** — The fit statistics table displays the following statistics for the training, validation, and test data sets (if available):
    - _AIC_ Akaike's Information Criterion

  - _ASE_ Average Squared Error
  - _AVERR_ Average Error Function
  - _DFE_ Degrees of Freedom for Error
  - _DFM_ Model Degrees of Freedom
  - _DFT_ Total Degrees of Freedom
  - _DIV_ Divisor for ASE
  - _ERR_ Error Function
  - _FPE_ Final Prediction Error
  - _MAX_ Maximum Absolute Error
  - _MSE_ Mean Square Errors
  - _NOBS_ Sum of Frequencies
  - _NW_ Number of Estimate Weights
  - _RASE_ Root Average Sum of Squares
  - _RFPE_ Root Final Prediction Error
  - _RMSE_ Root Mean Squared Error
  - _SBC_ Schwarz's Bayesian Criterion
  - _SSE_ Sum of Squared Error
  - _SUMW_ Sum of Case Weights Times Freq
  - _MISC_ Misclassification Rate
  - _KS_ Kolmogorov-Smirnov Statistic
  - _AUR_ Area under ROC
  - _Gini_ Gini Coefficient
  - _ARATIO_ Accuracy Ratio
- **Classification Chart** — a bar chart that shows the correct classification of the values of the target variable.
- **Score Rankings Overlay** — In a score rankings overlay chart, several statistics for each decile (group) of observations are plotted on the vertical axis. The statistics computed for the train and validate data sets are overlaid on the same graph. All observations in the scored data set are sorted by the posterior probabilities of the event level in descending order.

  By default, the horizontal axis of a score rankings chart displays the deciles (groups) of the observations.

  The vertical axis displays the following values:

  - Cumulative Lift
  - Lift
  - Gain
  - % Response
  - Cumulative % Response
  - % Captured Response

        - Cumulative % Captured Response
    - **Score Rankings Matrix** — In a score rankings matrix chart, several statistics for each decile (group) of observations are plotted on the vertical axis. Separate graphs are produced for the train and validate data sets. All observations in the scored data set are sorted by the posterior probabilities of the event level in descending order.
        - Cumulative Lift
        - Lift
        - Gain
        - % Response
        - Cumulative % Response
        - % Captured Response
        - Cumulative % Captured Response
    - **Score Distribution** — The Score Distribution chart plots the proportions of events (by default), non-events, and other values on the vertical axis. The values on the horizontal axis represent the model score of a bin. The model score, in this context, refers to the predicted probability that the target equals 1. The model score depends on the prediction of the target and the number of buckets used.

      The Score Distribution chart of a useful model shows a higher percentage of events for higher model scores and a higher percentage of non-events for lower model scores. The default chart is Percentage of Events.

      Other Score Distribution chart options include the following:
        - Percentage of Events
        - Number of Events
        - Cumulative Percentage of Events
- **Model**
    - **Scorepoints Code** — The Scorepoints Code displays the SAS code and formulas that are used to generate scorecard points for each of the scored variables.
    - **Score Distribution** — The following score distribution charts are available in the Scorecard node Model Results:
        - Event Odds
        - Log Event Odds
        - Score Points

      These score distribution charts show the distributions of the values of various statistics including the event odds, logarithm of event odds, and score points, respectively. The horizontal axis represents the values of the statistics. The vertical axis represents the frequency counts of applicants whose scores fall into the bins that are displayed on the horizontal axis. The range of the statistics is divided into ten bins. The values that are displayed along the horizontal axis are the lower limits of the bins.
    - **Trade-Off Plots** — The Trade-Off plots provide line plots of the following statistics in the Gains Table plotted against the lower limit of the score interval.
        - cumulative event rate and cumulative approval rate
        - average marginal profit and cumulative approval rate

  - average total profit and cumulative approval rate

- **Empirical Odds Plot** — The Empirical Odds plot displays a plot of the empirical odds plotted against the average values of the scorecard points for train and validate data sets.

- **Event Frequency Charts** — In the Event Frequency chart, event counts and rates are plotted against cutoff scores.

  The Event Frequency chart provides the following choices:

  - **Event Count** — (default setting) a plot of event counts plotted against cutoff score points. Event Count is the number of events in a score bucket.

  - **Cumulative Event Count** — a graph of cumulative event counts plotted against cutoff score points.

  - **Marginal Event Rates** — a graph of marginal event rates plotted against cutoff score points. The Marginal Event Rate is calculated as 100*(Event Count/Total Count).

  - Cumulative Event Rates — a graph of cumulative event rates plotted against cutoff score points. The Cumulative Event Rate is calculated as 100*(Event $\text{Count}_{\text{cumulative}}$ / Total $\text{Count}_{\text{cumulative}}$). Event $\text{Count}_{\text{cumulative}}$ is the cumulative number of events for all applicants with a score less than or equal to the cutoff score. Total $\text{Count}_{\text{cumulative}}$ is the cumulative number of applicants with a score less than or equal to the cutoff score.

- **Average Predicted Probability** — The Average Predicted Probability chart plots the average posterior probability of an event for all applicants versus a range of cutoff scores. Separate graphs are produced for the train and validate data sets.

- **Statistics Table** — The Statistics Table displays the following information for each of the original variables that met the selection criterion of the Interactive Grouping node that precedes the Scorecard node in the process flow diagram.

  - **Gini Statistic** — The Gini statistic measures the predictive power of the scorecard. A scorecard that provides better prediction has a bigger value of the Gini statistic. The equation for calculating the Gini statistic is as follows:

    $$\sum_{\lambda} \sum_{j} (1 - \hat{p}_{\lambda j}^{2})$$

    In the equation, $\hat{p}_{\lambda j}$ is the posterior probability of target value j for observations in the group λ.

  - **Information Value** — The Information Value coefficient measures the variable's contribution to the model.

  - **Variable Label** — the label used to identify a variable.

  - **Information Value Ordering** — the variable's rank according to the Information Value coefficient.

- **Gains Table** — The Gains Table displays a summary of various statistics for each score bucket. You set the number of bins to divide the score into (that is, the number of rows in the Gains Table) by using the Number of Buckets property. Each row in the Gains Table corresponds to a score bucket. The rows in the Gains Table are displayed in descending order of the score range. Here is a list of the Gains Table statistics.

- **Bucket** — the bucket ID.
- **Score Bucket** — the range of scorecard points for applicants whose scores are in the bucket.
- **Data Role** — the SAS data role of the variable.
- **Count** — the number of applicants whose scores fall into the score bucket.
- **Cumulative Count** — the number of applicants who have a score higher than the lower limit of the score bucket, regardless of their target value.
- **Event Count** — the number of applicants whose scores are in the bucket and who have the event target value.
- **Cumulative Event Count** — the number of applicants who have a score higher than the lower limit of the score bucket and who have the event target value.
- **Non-Event Count** — the number of applicants whose scores are in the bucket and who have the non-event target value.
- **Cumulative Non-Event Count** — the number of applicants who have a score higher than the lower limit of the score bucket and who have the non-event target value.
- **Marginal Event Rate** — the percentage of applicants whose scores are in the bucket and who have the even target value.
- **Marginal Non-Event Rate** — the percentage of applicants whose scores are in the bucket and who have the non-event target value.
- **Cumulative Event Rate** — Cumulative Event Count / Cumulative Count. In other words, the cumulative event rate is the proportion of applicants who have the event target value and a score that is higher than the lower limit of the score bucket among all those that have a score higher than the lower limit of the score bucket.
- **Cumulative Non-Event Rate** — the percentage of applicants who have a score higher than the lower limit of the score bucket and who have the non-event target value, among all those that have a score higher than the lower limit of the score bucket.
- **Average Predicted Probability** — the average of the posterior probabilities of applicants whose scores are in the bucket.
- **Low Predicted Probability Threshold** — the minimum value of posterior probabilities of applicants whose scores are in the bucket.
- **High Predicted Probability Threshold** — the maximum value of posterior probabilities of applicants whose scores are in the bucket.
- **Cumulative Approval Rate** — the cumulative count divided by the number of applicants in the data set, expressed in percent.
- **Average Marginal Profit** — the average expected profit per applicant if all applicants who have scores higher than the lower limit of the score bucket are accepted. Average marginal profit is calculated on the basis of the cumulative event count, cumulative non-event count, and the Revenue Accepted Good and Cost Accepted Bad properties.
- **Average Total Profit** — the expected total profit if all applicants who have a score higher than the lower limit of the score bucket are accepted. This number is calculated by multiplying the average marginal profit by the cumulative approval rate divided by 100.

- **Cutoff Score** — The upper bound of the scorecard points for applicants in a bucket.
- **Population Percentage** — the percentage of applicants who have a score higher than the lower limit of the score bucket and who have the event target value.
- **Average Scorecard Points** — the average of the scorecard points of the observations in a group.
- **Empirical Odds** — The empirical odds are calculated as follows: log(event_count / non_event_count), if both counts are non-missing and not equal to zero.
- **Predicted Odds** — The predicted odds are calculated as follows: log((average_predicted_value / (1-average_predicted_value)).

  If a prior vector has been declared, the Gains Table contains a column to display the adjusted values.

  If a Scorecard node is used in a process flow that specifies prior probabilities, then the Scorecard node Gains Table will contain the following adjusted variables:

  - adj_non_event_count
  - adj_cumulative_non_event_count
  - adj_count
  - adj_cumulative_count
  - adj_marginal_event_rate
  - adj_marginal_non_event_rate
  - adj_cumulative_event_rate
  - adj_cumulative_non_event_rate
  - adj_population_count
  - adj_approval_rate
  - adj_average_marginal_profit
  - adj_average_total_profit
  - adj_population_percentage

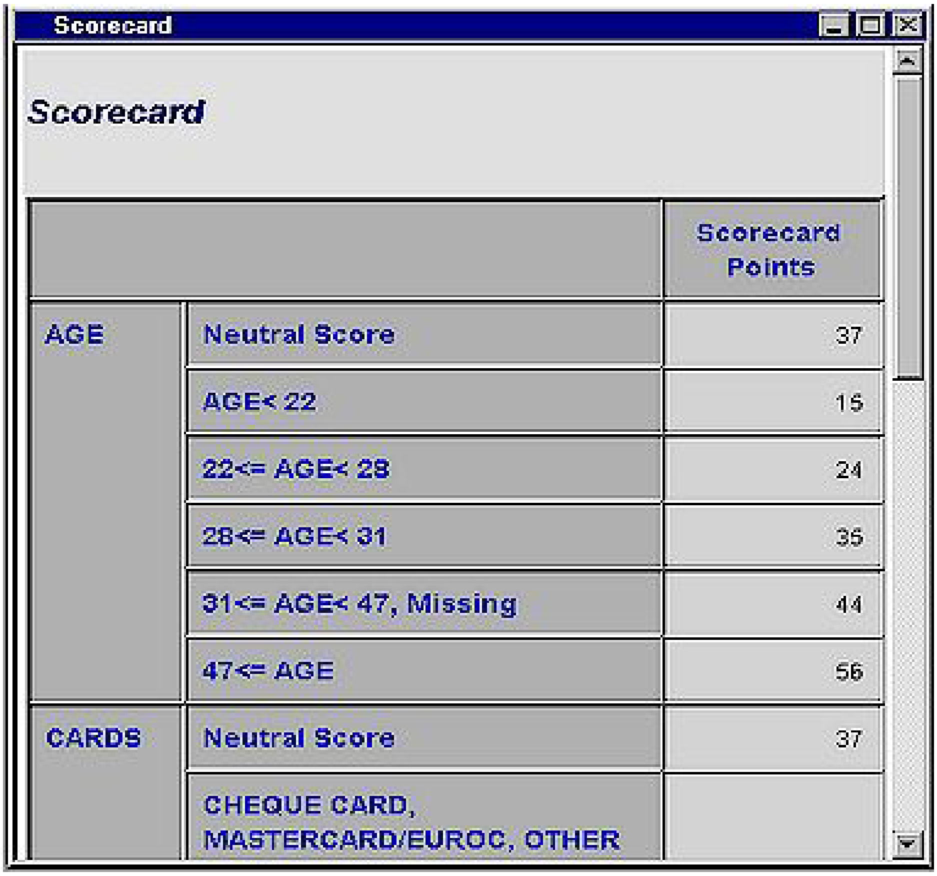
  The adjusted variables are calculated as follows:

  First, an offset is computed. TRAINPRIOR is the prior probability value that is proportional to the data. DECPRIOR is the adjusted probability value that you specified. E represents an event, and NE represents a non-event. The offset is defined by (E_TRAINPRIOR * NE_DECPRIOR) / (NE_TRAINPRIOR * E_DECPRIOR).

  Then the offset is used to adjust the non-event count._adj_non_event_count = _non_event_count * offset

  Finally, the adjusted non-event count is used during any of the adjusted calculations that are displayed in the Gains Table. _adj_count = _event_count + _adj_non_event_count.

- **Scorecard** — A summary scorecard contains information such as the attribute and the scorecard points for each attribute. Here is an example:

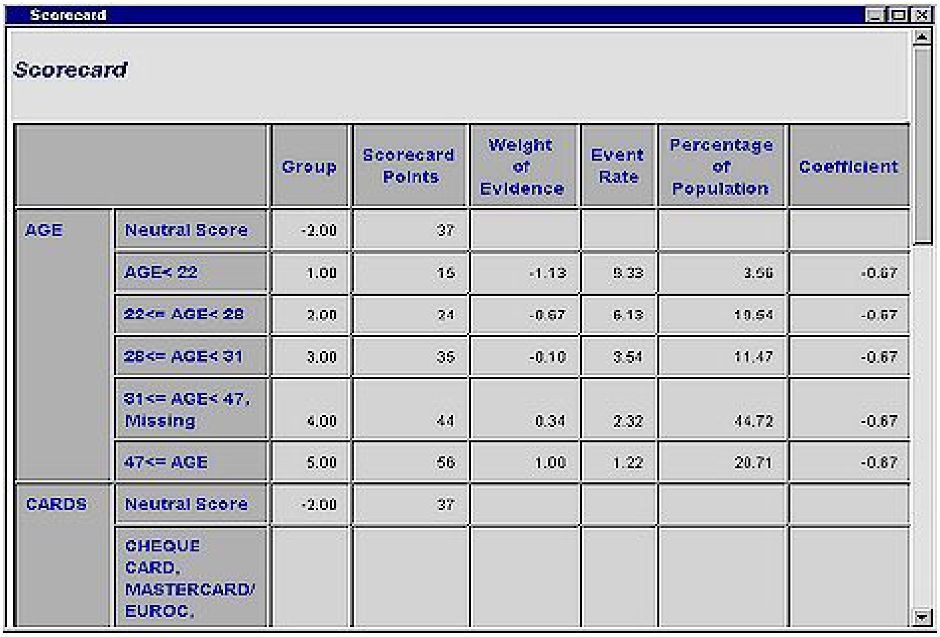


Scorecard

Scorecard

		Scorecard Points
AGE	Neutral Score	37
	AGE< 22	15
	22<= AGE< 28	24
	28<= AGE< 31	35
	31<= AGE< 47, Missing	44
	47<= AGE	56
CARDS	Neutral Score	37
	CHEQUE CARD, MASTERCARD/EUROC, OTHER	

A detailed scorecard displays additional information such as group number, weight of evidence, event rate, percentage of population, and regression coefficient for each attribute. Here is an example. A group number of -2 identifies the neutral score for each characteristic.



Scorecard

Scorecard

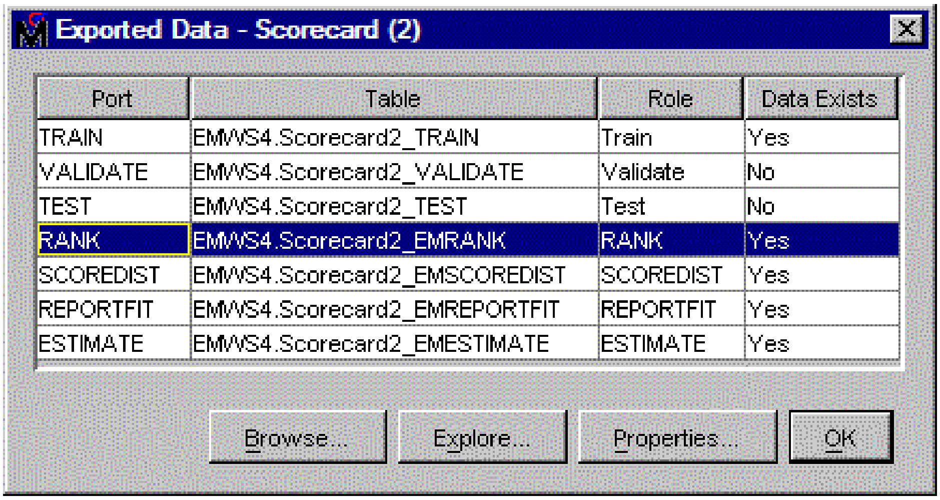
		Group	Scorecard Points	Weight of Evidence	Event Rate	Percentage of Population	Coefficient
AGE	Neutral Score	-2.00	37				
	AGE< 22	1.00	15	-1.13	9.33	3.56	-0.67
	22<= AGE< 28	2.00	24	-0.67	6.13	19.54	-0.67
	28<= AGE< 31	3.00	35	-0.10	3.54	11.47	-0.67
	31<= AGE< 47, Missing	4.00	44	0.34	2.32	44.72	-0.87
	47<= AGE	5.00	56	1.00	1.22	20.71	-0.87
CARDS	Neutral Score	-2.00	37				
	CHEQUE CARD, MASTERCARD/ EUROC,						

- **Effects Plot** — The Effects plot displays a bar chart of the absolute values of the regression coefficients for each independent variable in the regression model. Positive and negative values are differentiated by different colors of the bars.

- **Strength Statistics** — displays the following plots to assess the predictive power of the scorecard.
  - **Kolmogorov-Smirnov Plot** — The Kolmogorov-Smirnov chart shows the Kolmogorov-Smirnov statistics plotted against the score cutoff values. The Kolmogorov-Smirnov statistic measures the maximum vertical separation at a particular scorecard point between the cumulative distributions of applicants who have good scores and applicants who have bad scores . The Kolmogorov-Smirnov chart that is displayed in the Strength Statistics section of the Scorecard Results window displays the Kolmogorov-Smirnov statistic over the entire cutoff score range.
  - **ROC Plot** — The ROC (Receiver Operator Characteristic) chart is a graphical display that gives the measure of the predictive accuracy of a logistic regression model. It displays the sensitivity (the true positive rate) versus 1-specificity (the false positive rate) for a range of cutoff values.
  - **Captured Event Plot** — In a Captured Event plot, observations are first sorted in ascending order by the value of score points. The observations are then grouped into deciles. The Captured Event plot displays the cumulative proportion of the total event count on the vertical axis. The horizontal axis represents the cumulative proportion of the population.
  - **Accuracy Profile** — The Accuracy Profile is a graphical representation of the accuracy of the model. The **Saturated Model** graph represents a perfect model with complete accuracy, while the **Baseline** graph represents a completely random model with no predictive ability. The **Proportion of Total Event Count** graph represents the predictive model. Good predictive models will resemble the **Saturated Model** graph, while poor models will resemble the **Baseline** graph.
- **Tables** — displays the data table for a graph that you select.
- **Plot** — opens the Graph Wizard for the table that you select.

Output Data Sources of the Scorecard Node

To view the output Data Sources of the Scorecard node, select the node in the Diagram Workspace after the node has run. In the Scorecard node Properties panel, click the button to the right of the Exported Data property. The Exported Data - Scorecard window opens with a list of the output data sets that were produced by the Scorecard node:



Port	Table	Role	Data Exists
TRAIN	EMWS4.Scorecard2_TRAIN	Train	Yes
VALIDATE	EMWS4.Scorecard2_VALIDATE	Validate	No
TEST	EMWS4.Scorecard2_TEST	Test	No
RANK	EMWS4.Scorecard2_EMRANK	RANK	Yes
SCOREDIST	EMWS4.Scorecard2_EMSCOREDIST	SCOREDIST	Yes
REPORTFIT	EMWS4.Scorecard2_EMREPORTFIT	REPORTFIT	Yes
ESTIMATE	EMWS4.Scorecard2_EMESTIMATE	ESTIMATE	Yes

The Scorecard node can output as many as seven output data sets:

- Train data
- Validate data
- Test data
- Rank data
- Score Distribution data
- Fit Statistic Report Data
- Estimate Data