Paper: 129-2008

From Soup to Nuts: Practices in Data Management for Analytical Performance

David Duling, SAS Institute Inc., Cary, NC
Howard Plemmons, SAS Institute Inc., Cary, NC
Nancy Rausch, SAS Institute Inc., Cary, NC

ABSTRACT

Many customers face the issue of working with SAS®9 advanced analytics tools using data that comes from many different operational systems. The challenge presented to the customer is to extract, transform, and load the data into the analytical data structures with efficiency and scalability. This paper presents a case study describing practices to address the entire lifecycle of data access, aggregation, analysis, and deployment. A foundation discussion of how to interact with the Teradata database using SAS will be reviewed, followed by design and integration techniques available in SAS® Data Integration Studio to deliver data structures appropriate for analytical purposes. Finally, a presentation of how to leverage the data using SAS Enterprise Miner will be presented, including a discussion of techniques for managing algorithms that require large memory storage or multiple passes of the data.

INTRODUCTION

Data Miners typically explore many data sources in the enterprise data universe in search of features that will be used in building predictive and descriptive models. In this paper we will explore a use case that describes a best practices methodology for exploiting data for analytical processing. The end-result scenario is to provide an answer to the question: which customers are most likely to respond to next summer’s promotion? In this usage scenario, the data miner inspects the available data sources looking for information relating to prior promotions; sale amounts, items, and seasonality; past and current customer state attributes, and any other potentially relevant information. A model target must be identified that encodes the behavior we want to model, such as response to previous promotion. Model inputs must be identified from the additional data sources. Many of these data sources will be stored as transactions or daily aggregates and must be rolled-up into a single row of data containing terms that correlate to the model target. Various customer keys must be resolved and used to join data source extracts to form a single record for each customer. Only then may a statistical model be created that will be used to select customers for next year’s promotion.

All of this activity can be termed *data building*, and it is typically blamed for 80% or more of a data mining project’s scope. Many customers are seeking best practices for managing this activity to produce repeatable processes that they can use for both model development and model-production scoring. Often, they are working with more than one data warehousing system and need to make their data access as efficient as possible to minimize impact on production processes. We will build our best practice on this use case.

USE CASE

Starting with a typical question, which customers are most likely to respond to next summer's promotion, we can build a process that uses several data sources, potentially stored in several data systems. To simplify this paper, we will work from the point that the following data has already been aggregated in the source systems:

- promotion dates
- customers who received the promotion
- the time period for responses
- customers who purchased in the response time
- customers who did not respond

We will start with a table of customers who did and did not respond, and we will make other simplifications in extracting web data, customer demographics, and prior sales data. However, we still have the challenge of efficiently extracting these tables, identifying predictors, and joining records for use in model building. We will start with these data sources.

- **SUMMER PROMOTION RESPONSES:** This data contains a set of customer ID values that responded to the promotion and a set that did not respond. These records form our model target values. The records from all of the following data sources will form our model input values.
- **CUSTOMER DEMOGRAPHICS:** This data contains customer attributes, such as age, income, number of kids, number of cars. This data is often purchased from an outside vendor. These attributes are often important predictors in purchase models. This data source will contain records for a much greater number of customers than represented in the purchase response data.
- **STORE PURCHASES:** This data contains store purchase records that include a customer ID, the item, and the date and time stamp. Many records contain default or missing ID values for customers who made cash purchases in a store and did not present a loyalty card. Loyalty cards, credit cards, and web logins can be used to match purchase ID values with customer ID values.
- **WEB SESSIONS:** This data contains the web-log records from users of the company website. This data indicates current interest level and specific browser patterns, which can be used as model inputs. Some records contain cookie ID values.
- **SESSION-COOKIE:** This data identifies those web users that have registered or made purchases and who have known customer ID values. Not all web users have known cookie values or customer ID values.

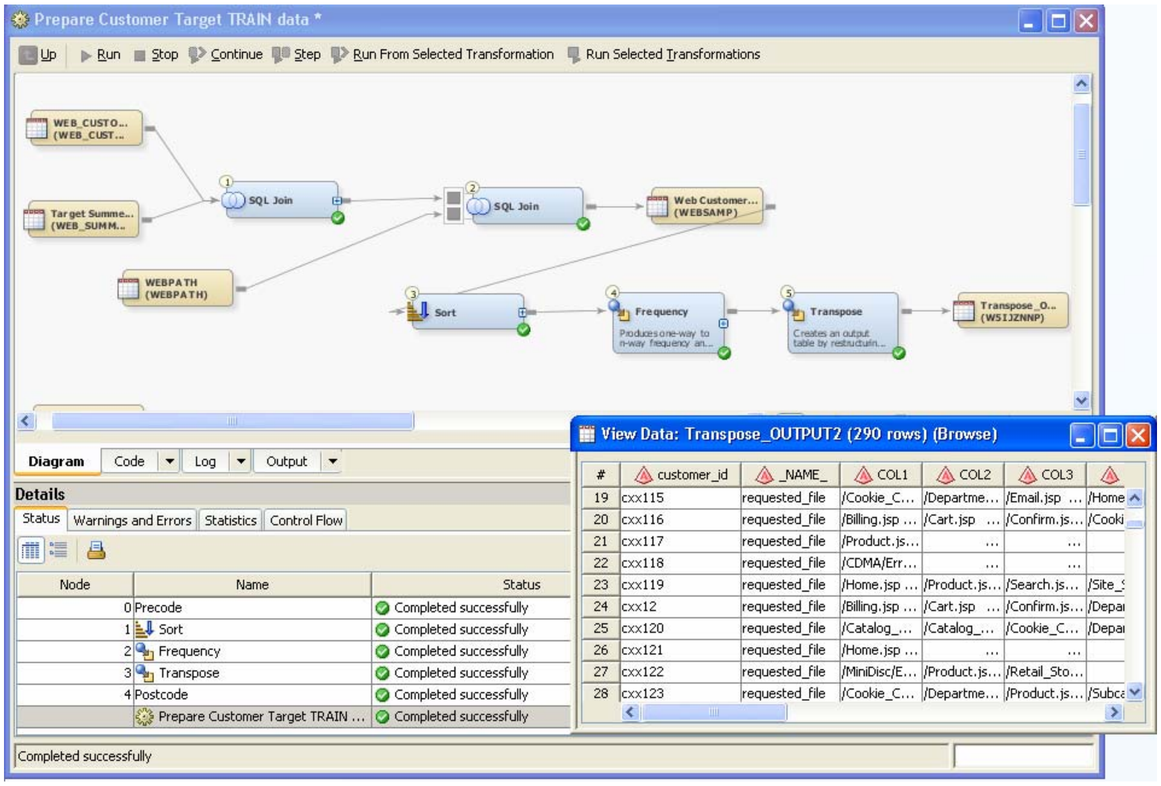
At this point, we need to consider the dimension of time. When the model is deployed, we will need a finite period between the time interval in which the prediction is made and the time interval when the event (purchase) will be made. For example, we might need to make predictions in May 2008 to identify customers that should receive a promotion offer in June that is good for the month of July. To model that process closely, we should identify a promotion from June 2007 (or earlier), extract model target data from the month of July 2007, and model input data from the period of time ending in the month of May 2007 (or earlier). These date time filters should be used when extracting data from the original sources.

USING SAS DATA INTEGRATION STUDIO

Once the data miner has identified the data sources that are needed for this project, the data sources must be efficiently combined to produce a final table set that will serve as inputs to SAS Enterprise Miner. SAS Data Integration Studio provides a visual design environment for building SAS jobs with many complex packaged transformations for transforming incoming data. SAS Data Integration Studio also has transformation logic tailored for use with Enterprise Miner. Its features include a scoring transform for integrating SAS Enterprise Miner models into a repeatable process; and several analytical transformations for handling frequency, transpose, and other complex data manipulation functions. These features assist in providing efficient transformation logic for creating the target data needed for modeling.

One source of data that is used in the sample scenario comes from web logs. Data gleaned from company websites such as web logs and web session data, when available, can provide a rich source of information that you can use in predictive modeling. Transactional web data is typically retrieved from web log and other content files, and summarized to data sets or to an RDBMS for further processing. In the example use case, the web-log records and session cookie information have been extracted to data sets, which could be SAS data sets or data stored in an RDBMS ready for processing. The web data needs to be extracted and joined with customer information. Once enriched with actual customer ids, the data can provide specific details about customers' interests and browser patterns.

In the example process below, data is brought in from the web logs, joined with web-session cookie information, and joined with the customer data set to produce an intermediate data set that provides detailed information about each customer's web-browser use.


Prepare Customer Target TRAIN data *
Up
Run
Stop
Continue
Step
Run From Selected Transformation
Run Selected Transformations
WEB_CUSTO... (WEB_CUST...
Target Summe... (WEB_SUMM...
WEBPATH (WEBPATH)
SQL Join
SQL Join
Web Customer... (WEBSAMP)
Sort
Frequency
Produces one-way to n-way frequency an...
Transpose
Creates an output table by restructurin...
Transpose_O... (W5I3ZNNP)
Diagram
Code
Log
Output
Details
Status
Warnings and Errors
Statistics
Control Flow
Completed successfully

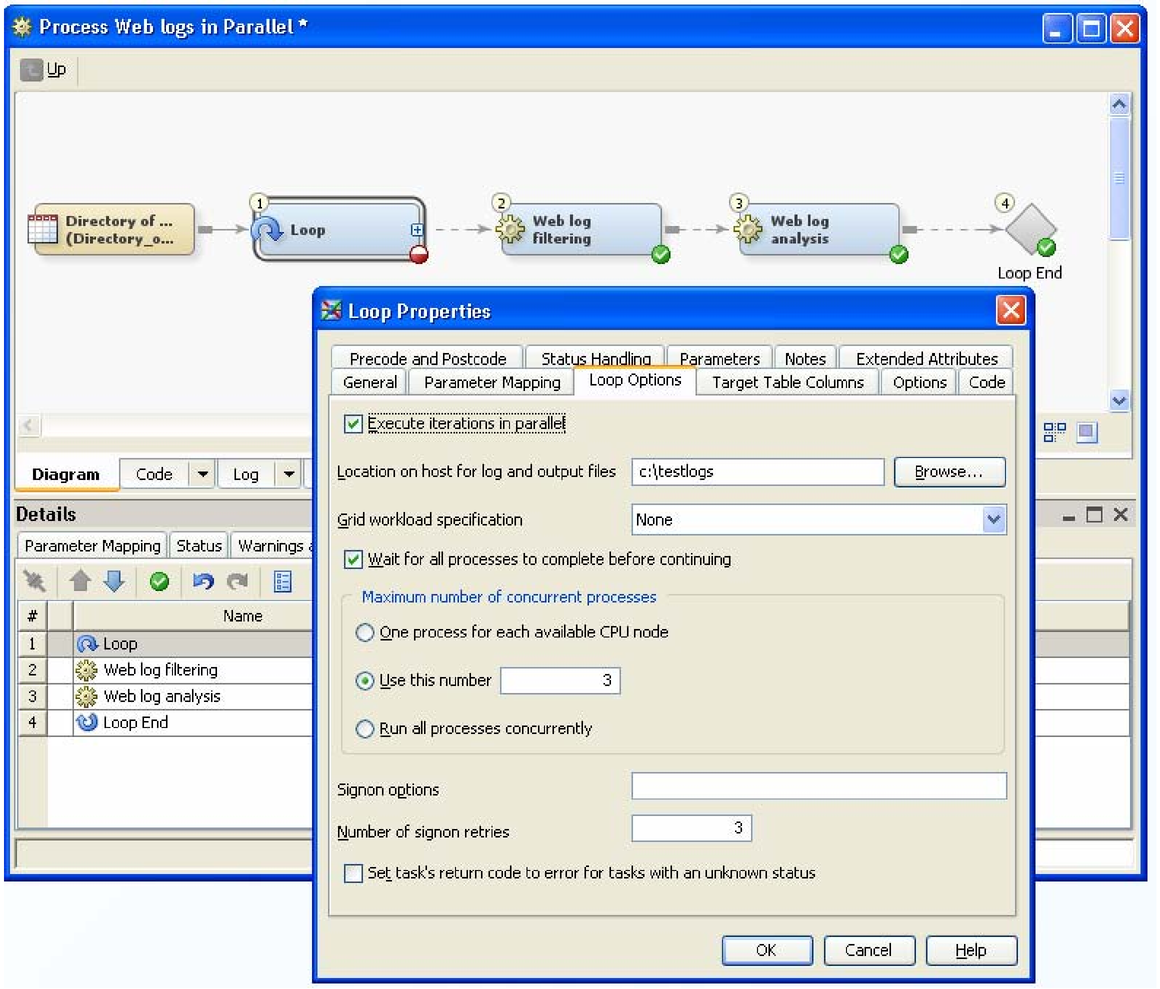
Node	Name	Status
0	Precode	Completed successfully
1	Sort	Completed successfully
2	Frequency	Completed successfully
3	Transpose	Completed successfully
4	Postcode	Completed successfully
	Prepare Customer Target TRAIN ...	Completed successfully

View Data: Transpose_OUTPUT2 (290 rows) (Browse)

#	customer_id	_NAME_	COL1	COL2	COL3	
19	cxx115	requested_file	/Cookie_C...	/Departme...	/Email.jsp ...	/Home
20	cxx116	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Cooki
21	cxx117	requested_file	/Product.js...	...	...	
22	cxx118	requested_file	/CDMA/Err...	...	...	
23	cxx119	requested_file	/Home.jsp ...	/Product.js...	/Search.js...	/Site_
24	cxx12	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Depai
25	cxx120	requested_file	/Catalog_...	/Catalog_...	/Cookie_C...	/Depai
26	cxx121	requested_file	/Home.jsp ...	...	...	
27	cxx122	requested_file	/MiniDisc/E...	/Product.js...	/Retail_Sto...	
28	cxx123	requested_file	/Cookie_C...	/Departme...	/Product.js...	/Subca

Figure 1: A SAS Data Integration Studio process for joining incoming web data with customer information.

There are several best practices to consider in the above process, particularly if you are working with large data volumes. Web logs can provide a significant challenge when you are scaling an ETL process because of the volume of information from the web logs, and because information from web servers usually comes from several sources. When working with a process that uses web information, consider cleansing and aggregating the information from the web sources early to reduce the number of records and to increase the relevance of the web-long information. Web data also lends itself to reading the log information in parallel using an iterative function such as the SAS Data Integration Studio Loop transform, which can process data in parallel from multiple sources. The parallel process can be run on a machine that has multiple CPUs available and it can be scaled to support multiple servers, such as in a Grid-enabled system.



#	Name
1	Loop
2	Web log filtering
3	Web log analysis
4	Loop End

Loop Properties

[✓] Execute iterations in parallel

Location on host for log and output files: c:\testlogs

Grid workload specification: None

[✓] Wait for all processes to complete before continuing

Maximum number of concurrent processes
- ○ One process for each available CPU node
- ● Use this number: 3
- ○ Run all processes concurrently

Signon options:

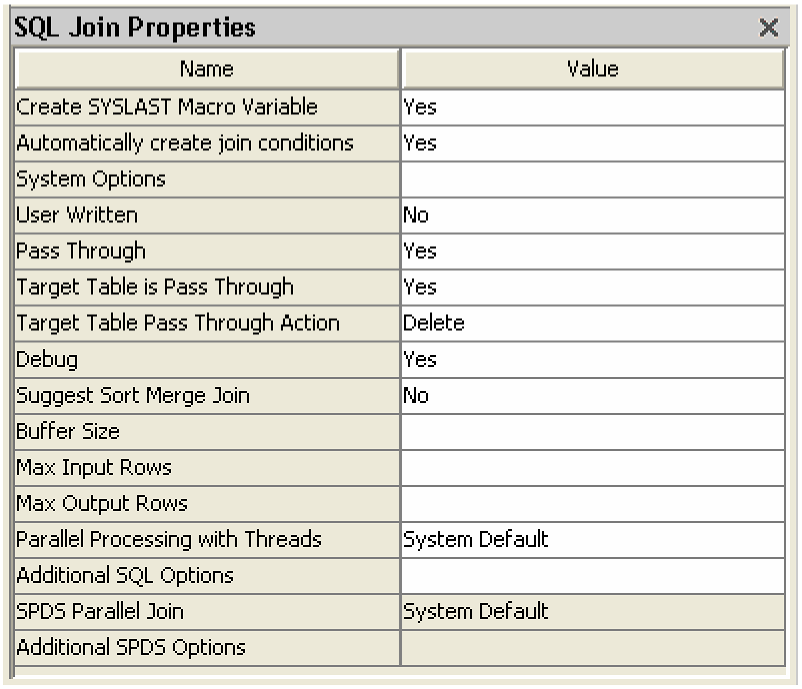
Number of signon retries: 3

[ ] Set task's return code to error for tasks with an unknown status

Figure 2: An example process using the SAS Data Integration Studio Loop transform to process web data in parallel.

In the above example, the SAS Data Integration Studio Loop transform is used to process the incoming web logs in parallel and to produce a result set of tables, one per iteration. The Loop transform uses SAS MP/Connect technology to read and process the incoming data in parallel. Input structures are parameterized so that substitution values for each run are supplied, enabling the same process to apply to N iterations of the incoming data. To produce the final target data set that is required for analysis with SAS Enterprise Miner, you can recombine the resulting tables using the Load, Append, or (if using Scalable Performance Data Server) the SPD Server Cluster transforms. All of these are available in SAS Data Integration Studio.

If the data you want to combine comes from an RDBMS system, such as Teradata, another best practice is to perform the join in the RDBMS system where the data resides, and then move the results. SAS Data Integration Studio enables you to develop optimized joins when you work with RDBMS systems through bulk-load options that support upload to the RDBMS of source tables, and explicit pass-through code generation.

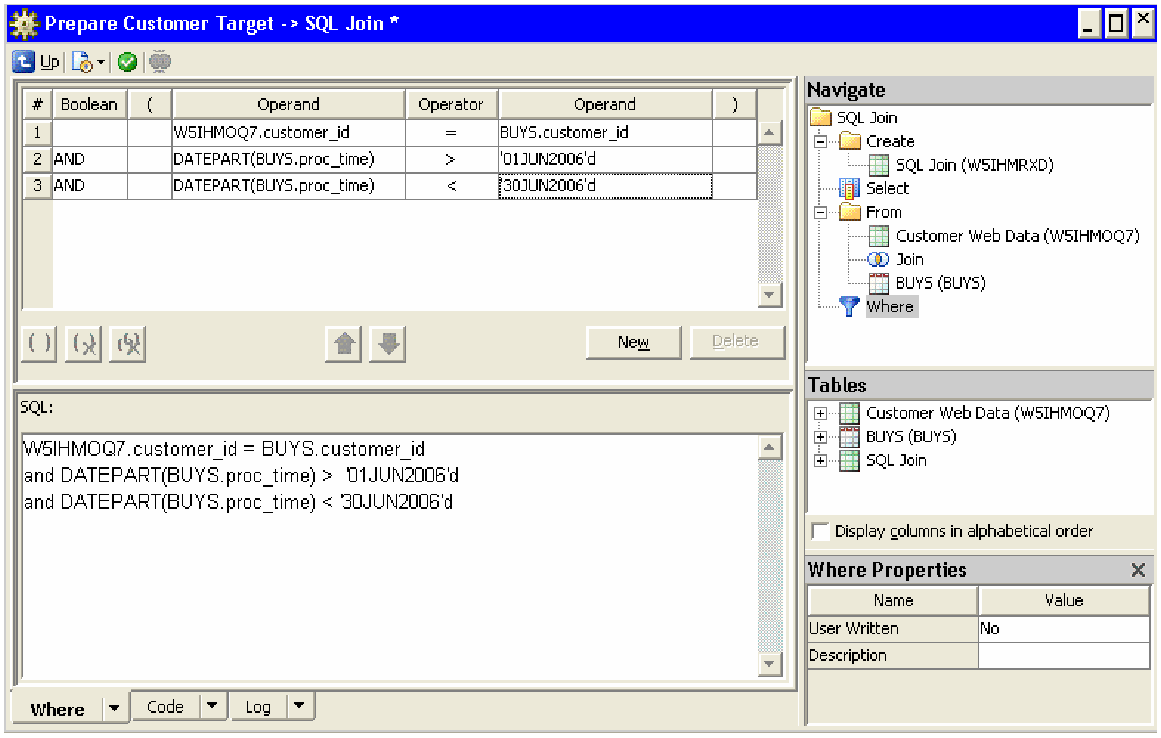


SQL Join Properties

Name	Value
Create SYSLAST Macro Variable	Yes
Automatically create join conditions	Yes
System Options	
User Written	No
Pass Through	Yes
Target Table is Pass Through	Yes
Target Table Pass Through Action	Delete
Debug	Yes
Suggest Sort Merge Join	No
Buffer Size	
Max Input Rows	
Max Output Rows	
Parallel Processing with Threads	System Default
Additional SQL Options	
SPDS Parallel Join	System Default
Additional SPDS Options	

Figure 3: Options available for optimizing the performance of your joins using the SAS Data Integration Studio Join transform.

Once the customer data has been enriched with information about web patterns, you need to add joins that subset the data by the time period of interest to the data modeler. This would normally be done via subsetting the information in the WHERE clause of the join. In the example below, we limit the data to the time period in which the modeler is interested by extracting data from a similar month in the past where a promotion was run.



#	Boolean	(	Operand	Operator	Operand	)
1			W5IHMOQ7.customer_id	=	BUYS.customer_id	
2	AND		DATEPART(BUYS.proc_time)	>	'01JUN2006'd	
3	AND		DATEPART(BUYS.proc_time)	<	'30JUN2006'd	

```
W5IHMOQ7.customer_id = BUYS.customer_id
and DATEPART(BUYS.proc_time) >  '01JUN2006'd
and DATEPART(BUYS.proc_time) < '30JUN2006'd
```

Name	Value
User Written	No
Description	

Figure 4: Developing a WHERE clause for handling date/time subsetting using the Join transform

Here, too, there are some best practices that will ensure the best possible performance of your job. When working with time-period data, your first instinct might be to use SAS date functions. These functions are very powerful, but there are some issues you need to be aware of when you use a SAS date function (or any function that exists only in SAS):

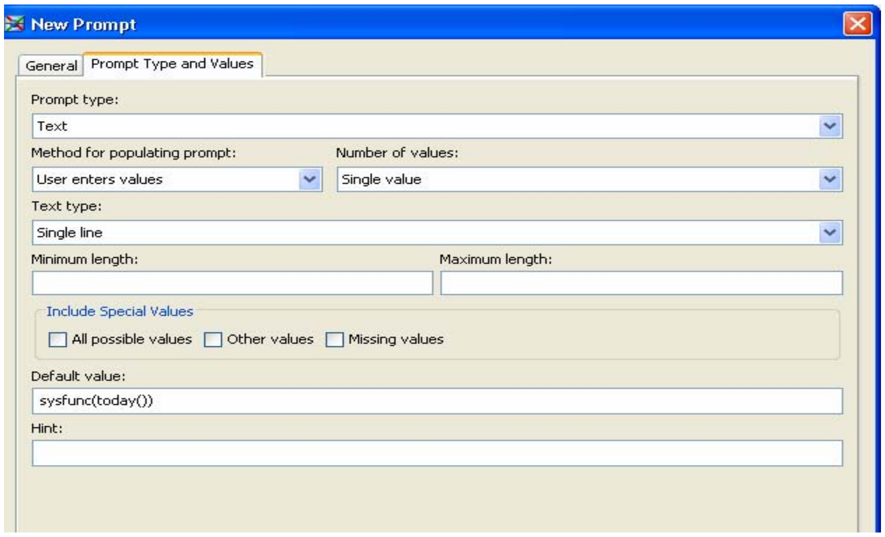
- Function calls incur overhead for every row that is processed in the step. This overhead should be avoided for WHERE filtering or other iterative processes when the function's desired return value is, in effect, a constant value across all rows. The cost is negligible for small tables, but it is substantial if there are millions of rows.
- During a long-running process, SAS date functions can yield unintended or undesirable results. For example, the value that is returned by the TODAY( ) function will change if the job runs past midnight into the following day.
- SAS might not support the passing down of these functions for evaluation on an RDBMS. The result is that each record will be read into SAS so that the function can be processed.
- SAS might not support passing some WHERE clause expressions that contain date and time variables when working with an RDBMS.

One alternative to using a SAS function directly in a filter or iterative process is to pre-evaluate the function expression in a SAS macro variable definition. Then, you can use the SAS macro variable in a direct comparison in a SAS WHERE clause or in assignments in a SAS DATA step.   For example, the following would be a better way to handle calling the TODAY() function:

```
/* macro variable for subsequent variable assignment */
%let target_date=%sysfunc(today());

data _null_;
format date_var date9.;
do n=1 to 1000000;
date_var = &target_date;
end;
put date_var=;
run;
```

In SAS Data Integration Studio you can accomplish this by specifying a parameter for the function, and then using that parameter in the mapping expression.



Prompt type: Text
Method for populating prompt: User enters values
Number of values: Single value
Text type: Single line
Minimum length:
Maximum length:
Include Special Values
☐ All possible values ☐ Other values ☐ Missing values
Default value: sysfunc(today())
Hint:

Figure 5: Defining macro variables that can be defined outside of an expression

Continuing the process of building the target table, additional transformations that perform a Frequency Analysis, and then Transpose the data can be built in the SAS Data Integration process. The final target table has all of the columns required as input to SAS Enterprise Miner.

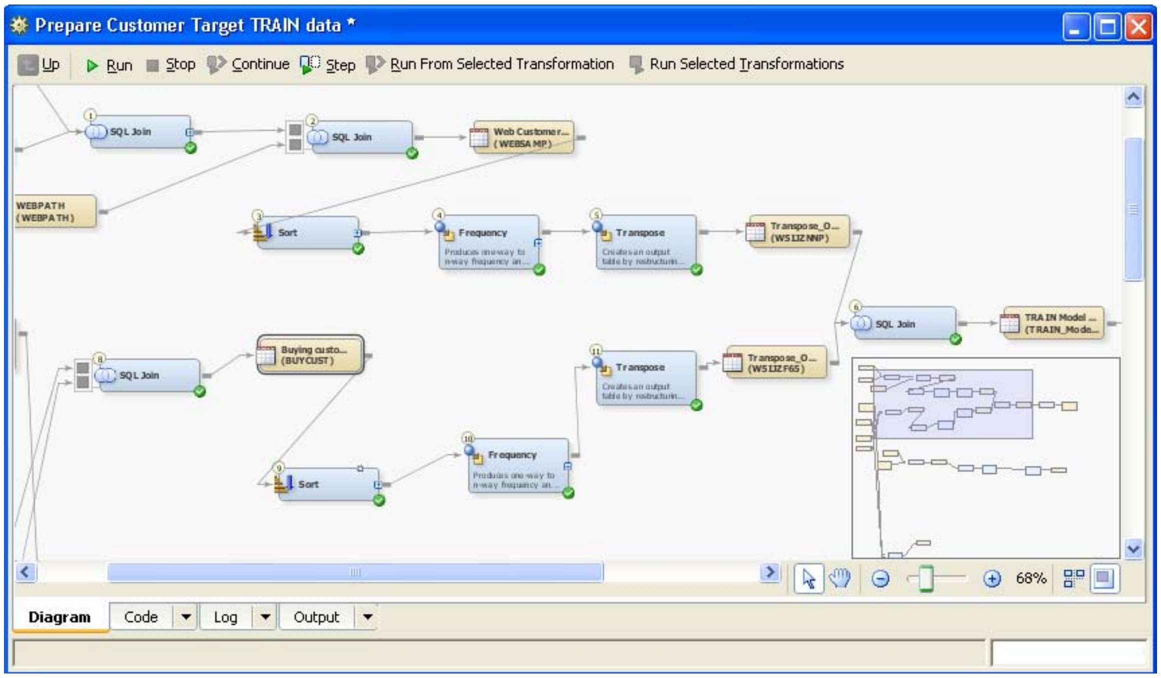
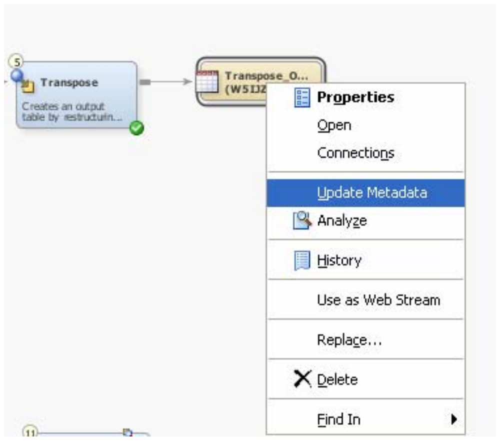
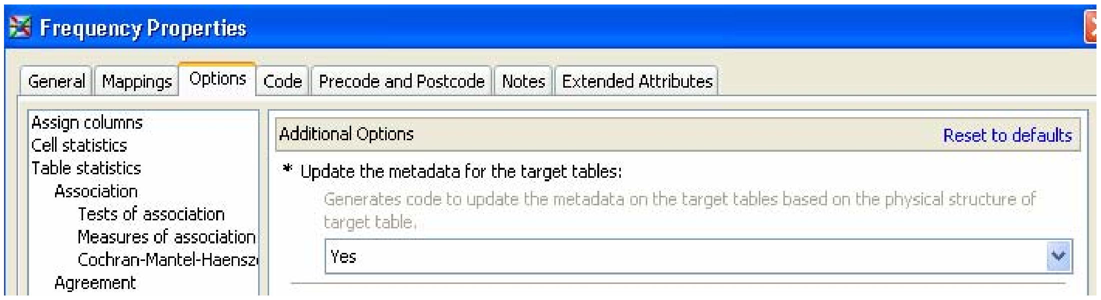

Prepare Customer Target TRAIN data *
Up
Run
Stop
Continue
Step
Run From Selected Transformation
Run Selected Transformations
SQL Join
Web Customer... (WEBSAMP)
WEBPATH (WEBPATH)
Sort
Frequency
Transpose
Transpose_O... (W51IZNNP)
TRAIN Model ... (TRAIN_Mode...
Buying custo... (BUYCUST)
Transpose_O... (W51IZF65)
68%
Diagram
Code
Log
Output

Figure 6: Final steps in the process of building a modeling ready source data set

In the above process, it is not necessary to have any knowledge of the data structures involved. SAS Data Integration Studio provides the ability to update data structures such as temporary and permanent tables through the use of dynamic registration. This makes it easy to build your processes without any a-priori knowledge of your intermediate or final data structures. SAS Data Integration Studio also allows you to create metadata programmatically that can be included in data-building processes. When you need to schedule data preparation in batch jobs, these techniques can be a great productivity tool.

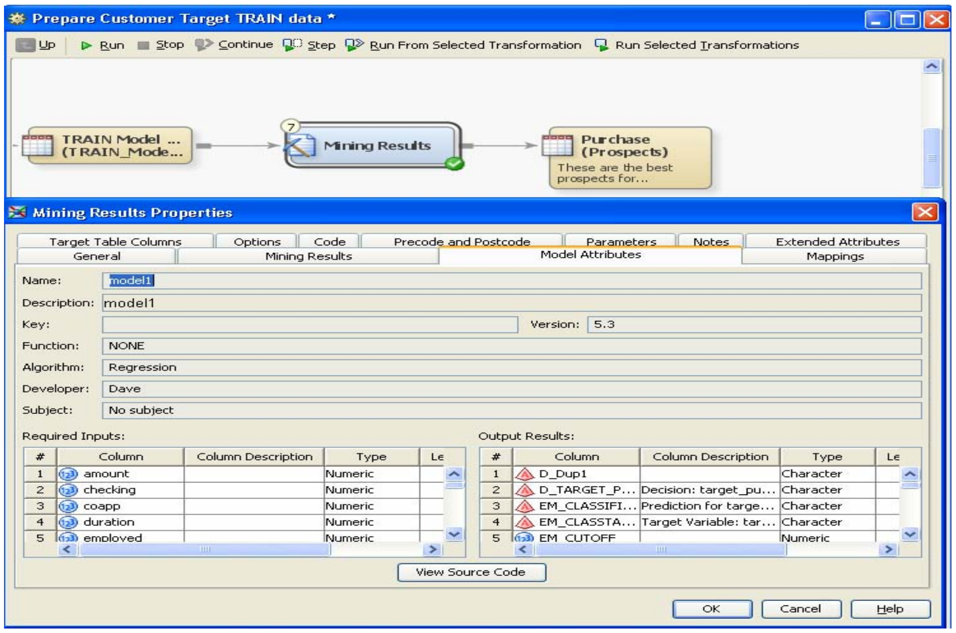




* Update the metadata for the target tables:
Generates code to update the metadata on the target tables based on the physical structure of target table.
Yes

Figure 7: Dynamic registration of metadata about intermediate and final target data structures

The Mining Results transformation is an additional feature available as an integration point between the two products. Once a model has been developed in SAS Enterprise Miner, it can be shared between SAS Data Integration Studio and SAS Enterprise Miner. You can use the Mining Results transformation in Data Integration Studio to apply the model in a job that can be deployed and scheduled to run periodically, for example as part of a nightly production process.



Name:	model1
Description:	model1
Key:	
Version:	5.3
Function:	NONE
Algorithm:	Regression
Developer:	Dave
Subject:	No subject

Required Inputs:

#	Column	Column Description	Type	Le
1	amount		Numeric	
2	checking		Numeric	
3	coapp		Numeric	
4	duration		Numeric	
5	employed		Numeric	

Output Results:

#	Column	Column Description	Type	Le
1	D_Dup1		Character	
2	D_TARGET_P...	Decision: target_pu...	Character	
3	EM_CLASSIFI...	Prediction for targe...	Character	
4	EM_CLASSTA...	Target Variable: tar...	Character	
5	EM_CUTOFF		Numeric	

Figure 8: Mining results transformation for integrating Enterprise Miner models into your production jobs

The Mining Results transformation reads the metadata that is registered at the server about the model inputs and outputs. It provides detailed information about the model source and targets so that this data can be appropriately mapped to produce the correct results provided by the model.

Models can also be exported and imported between development and production systems to ensure that production processes are updated as needed.

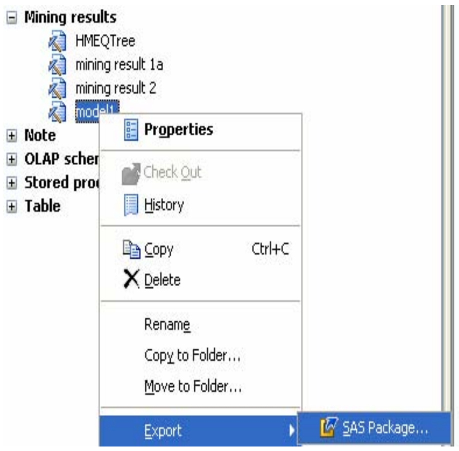
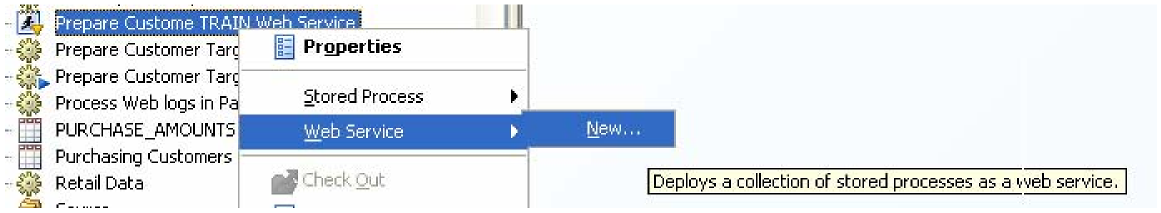
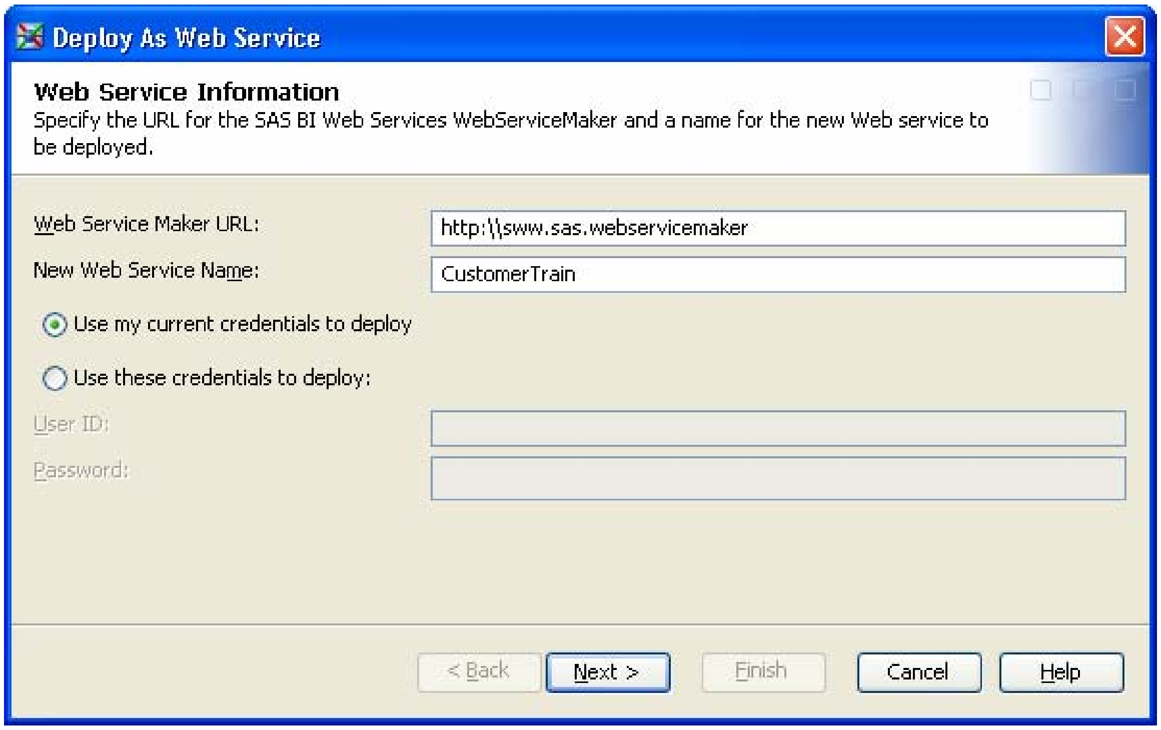


Figure 9: Models can be imported and exported between development and production environments

When you need to run a model on demand, SAS Data Integration Studio supports other methods for deploying jobs that leverage Enterprise Miner model scoring as well. Jobs can also be deployed as a Stored Process and as a Web Service to enable real-time data scoring in reports and other on-demand areas.





Web Service Information

Specify the URL for the SAS BI Web Services WebServiceMaker and a name for the new Web service to be deployed.

Web Service Maker URL: http:\\sww.sas.webservicemaker

New Web Service Name: CustomerTrain

[✓ Use my current credentials to deploy]

Use these credentials to deploy:

User ID:

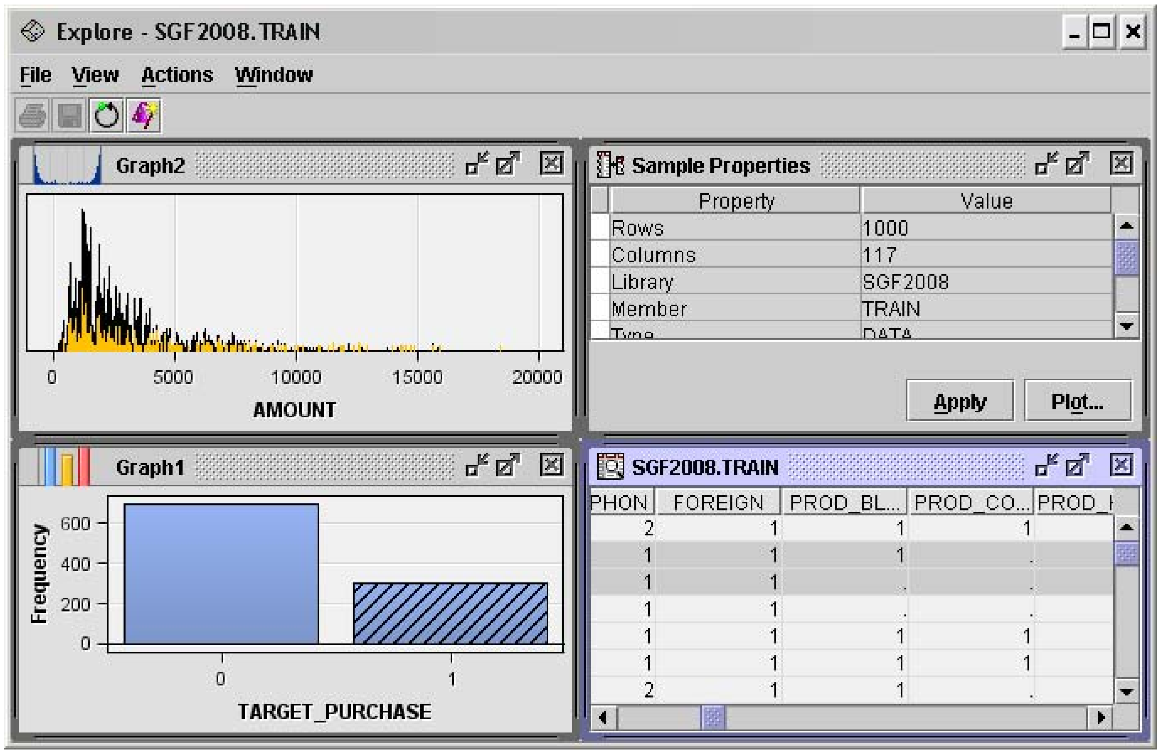
Password:

Figure 10: Jobs can be deployed for scheduling, as a Stored Process, and as a Web Service

USING SAS ENTERPRISE MINER

Because we have successfully offloaded the data preprocessing to Data Integration Studio, the job of the data miner is now much easier. We start our model development with a single table that has joined the customer demographics, web browsing, prior purchases, and response to promotion data sources into a single table ready for analysis. While our scenario has made use of four data sources, in many cases data miners are working with dozens of potential data sources.

Inside Enterprise Miner, our first task is often merely to explore the data to get to know the characteristics of the variables. The view below shows the distributions of two variables, the table properties, and the actual table. We can see that there are 1000 rows in this sample and 117 columns.



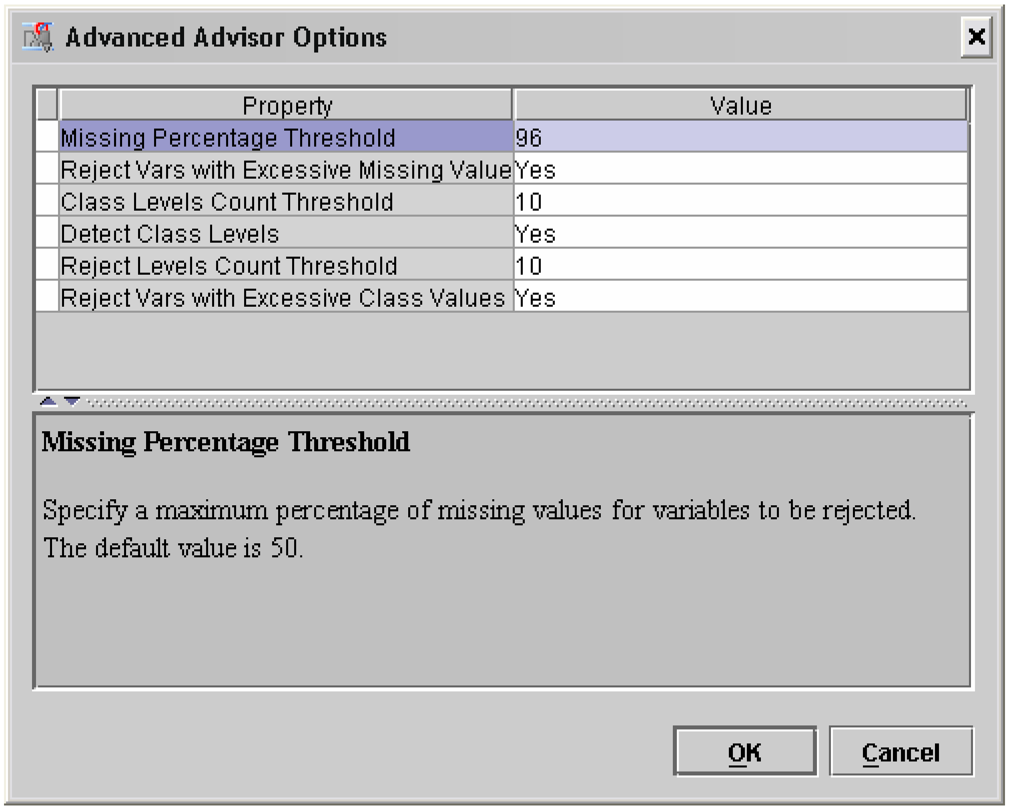
Property	Value
Rows	1000
Columns	117
Library	SGF2008
Member	TRAIN
Type	DATA

PHON	FOREIGN	PROD_BL...	PROD_CO...
2	1	1	1
1	1	1	.
1	1	.	.
1	1	.	.
1	1	1	1
1	1	1	1
2	1	1	.

Figure 11: Exploring the data using SAS Enterprise Miner

We also notice that the product purchase count columns were prepared with missing values where we would expect zeros. The same applies to the web page count columns. We will take care of this once we begin our analysis.

The next task is to use the Data Source Wizard to build analytical metadata. This tool leads the user through the process of identifying variable roles (for example, input, target, ID, freq) and their measurement levels (interval, nominal, or ordinal). The wizard includes an advanced advisor routine that scans and summarizes the data to apply rules that create the measurement levels and some variable roles. Because we know that the missing values are really zero counts, we change the settings of the advisor for the threshold for rejecting variables with missing values from 50% to 96%. That is, any variable with more than 96% missing values will be rejected from analysis. Meanwhile, any numeric variable with less than 10 distinct values will be used as a nominal (class) variable with distinct values instead of an interval (continuous) variable.



Advanced Advisor Options

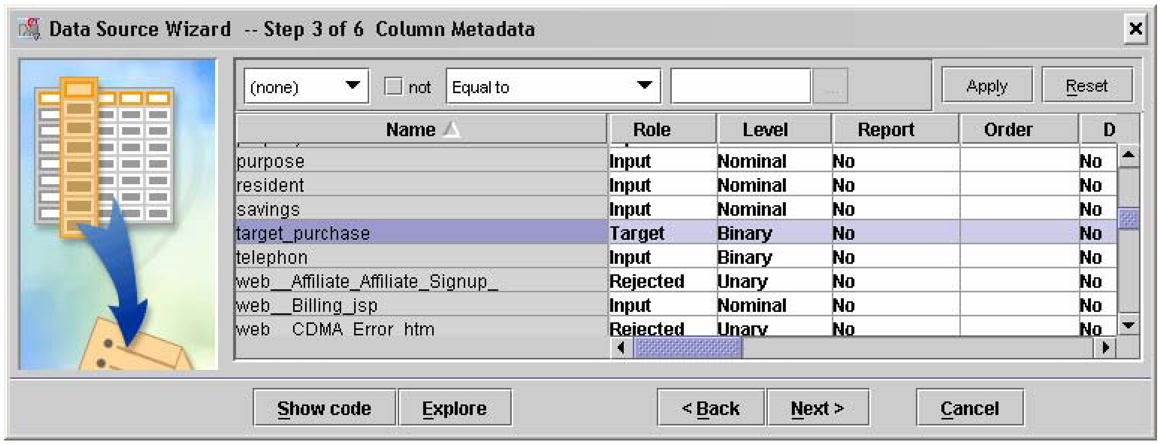
Property	Value
Missing Percentage Threshold	96
Reject Vars with Excessive Missing Value	Yes
Class Levels Count Threshold	10
Detect Class Levels	Yes
Reject Levels Count Threshold	10
Reject Vars with Excessive Class Values	Yes

Missing Percentage Threshold

Specify a maximum percentage of missing values for variables to be rejected. The default value is 50.

Figure 12: Using the Advisor routine of the Data Source Wizard

Next, the data is scanned, the rules are applied, and we can review the settings and make any necessary changes. Some of the rules apply to variable names; for example, the column Customer_Id has been made an ID variable, and the column Target_Purchases has been made a target column. This is an important point for our data builders: create logical column names when building data.



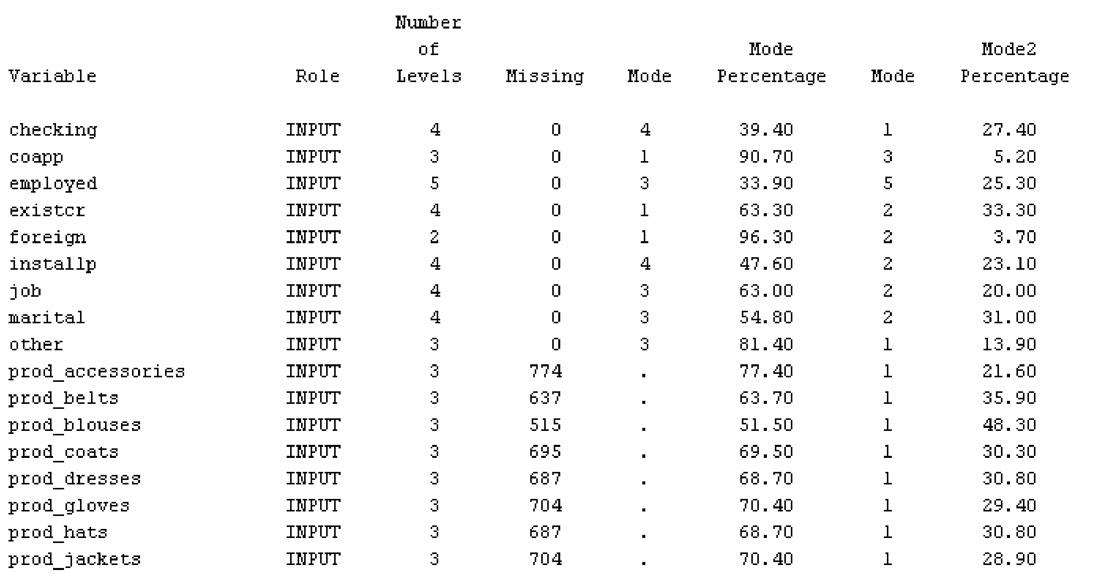
Data Source Wizard -- Step 3 of 6 Column Metadata

Name	Role	Level	Report	Order	D
purpose	Input	Nominal	No		No
resident	Input	Nominal	No		No
savings	Input	Nominal	No		No
target_purchase	Target	Binary	No		No
telephon	Input	Binary	No		No
web__Affiliate_Affiliate_Signup_	Rejected	Unary	No		No
web__Billing_jsp	Input	Nominal	No		No
web__CDMA_Error_htm	Rejected	Unary	No		No

Figure 13: Review the settings and update rules when needed

For automation purposes, Enterprise Miner also includes a data source macro that will create the analytical metadata programmatically and can be included in data building processes. When data preparation can be scheduled in batch jobs, this macro can be a great productivity tool.

Now that we have constructed the metadata, we can begin building models. The first task is often to explore the statistical distributions of the input and target columns. The StatExplore node produces both plots and easy-to-read output as shown below.

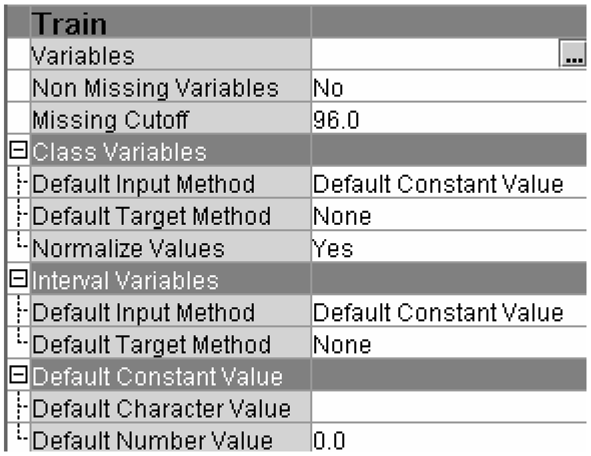


```
                                  Number
                                    of                                   Mode                    Mode2
Variable                 Role     Levels    Missing    Mode    Percentage    Mode    Percentage

checking                 INPUT      4          0         4        39.40        1        27.40
coapp                    INPUT      3          0         1        90.70        3         5.20
employed                 INPUT      5          0         3        33.90        5        25.30
existcr                  INPUT      4          0         1        63.30        2        33.30
foreign                  INPUT      2          0         1        96.30        2         3.70
installp                 INPUT      4          0         4        47.60        2        23.10
job                      INPUT      4          0         3        63.00        2        20.00
marital                  INPUT      4          0         3        54.80        2        31.00
other                    INPUT      3          0         3        81.40        1        13.90
prod_accessories         INPUT      3        774         .        77.40        1        21.60
prod_belts               INPUT      3        637         .        63.70        1        35.90
prod_blouses             INPUT      3        515         .        51.50        1        48.30
prod_coats               INPUT      3        695         .        69.50        1        30.30
prod_dresses             INPUT      3        687         .        68.70        1        30.80
prod_gloves              INPUT      3        704         .        70.40        1        29.40
prod_hats                INPUT      3        687         .        68.70        1        30.80
prod_jackets             INPUT      3        704         .        70.40        1        28.90
```

Figure 14: Use the StatExplore node to produce plots and easy-to-read output

This output confirms our earlier observation about the missing value distributions of the product and web page count variables. Therefore, our next task is to use the Impute node. All we need to do is change the properties for replacement rules for the default methods to “Default Constant Value” and enter a value of zero. All numeric missing values will now be replaced by zeros and the SAS code for missing value replacement will become part of the model scoring code. The property sheet for the Impute node is shown below.



Train	
Variables	...
Non Missing Variables	No
Missing Cutoff	96.0
Class Variables	
Default Input Method	Default Constant Value
Default Target Method	None
Normalize Values	Yes
Interval Variables	
Default Input Method	Default Constant Value
Default Target Method	None
Default Constant Value	
Default Character Value	
Default Number Value	0.0

Figure 15: Impute node property sheet showing the details of the TRAIN data

We now want to build models that predict the value of the variable target_purchase. The variable is distributed as 0, for *did not respond* and 1, for *did respond,* respectively, which makes it suitable for statistical classification techniques. One common task is to reduce model complexity by eliminating redundant and insignificant terms from the function. Enterprise Miner contains multiple techniques for variable selection. We will use two different techniques and compare the results. We will also use three different modeling techniques and select the best model. This portion of the process flow diagram is shown below.

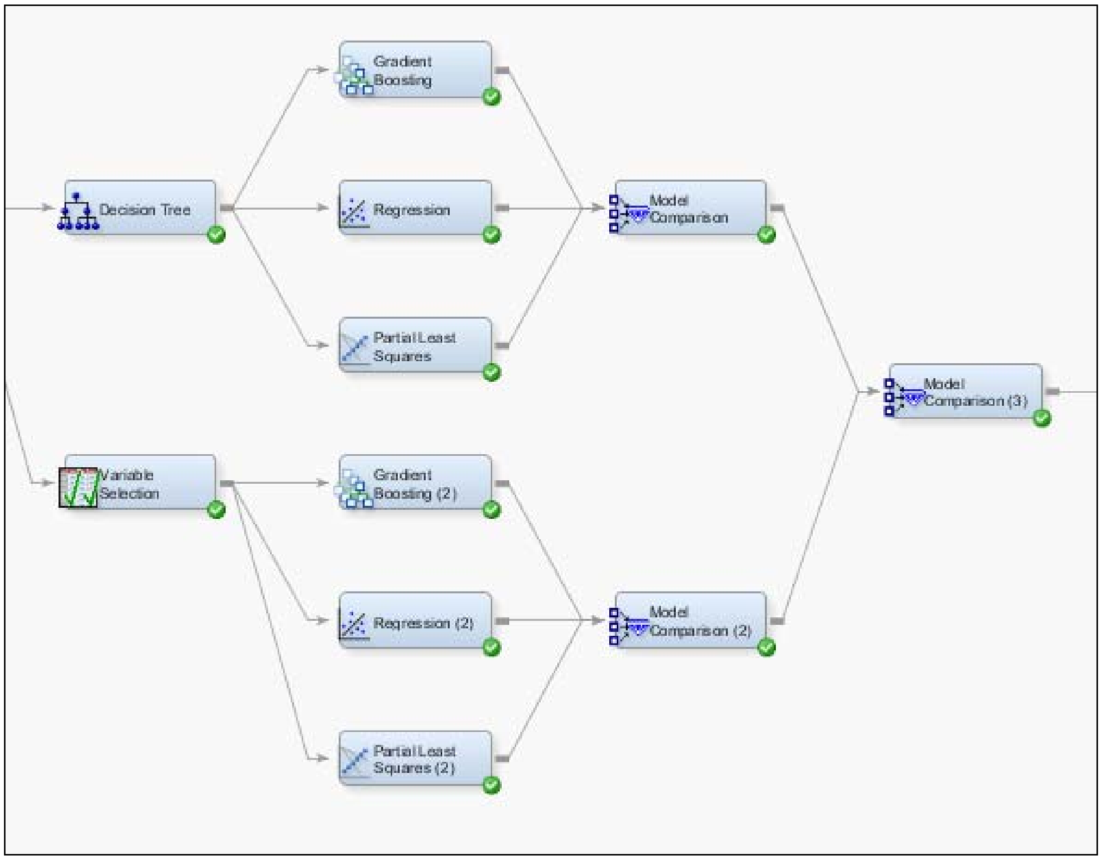
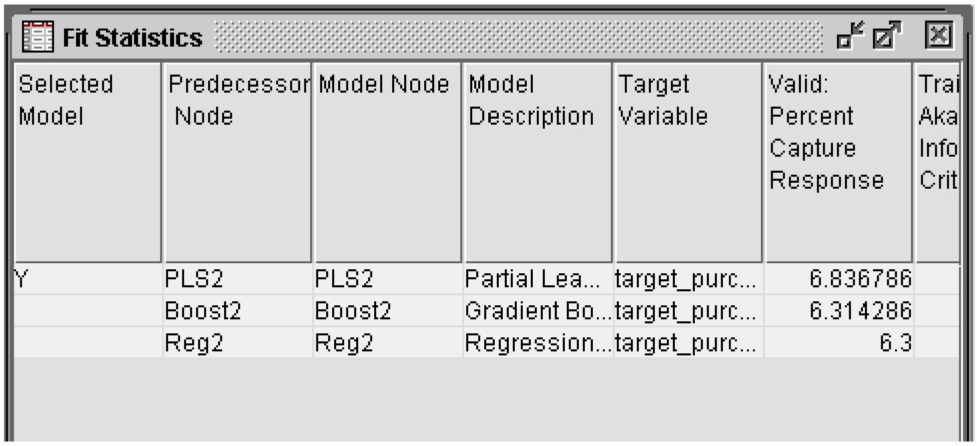


Figure 16: Enterprise Miner process using multiple modeling techniques to create the best model

The Decision Tree node implements a statistical technique that is known as recursive partitioning to identify values of variables that can be used to separate rows of the data based on affinity with the target variable. Because the algorithm seeks to find a minimal set of variables, this technique is often used for variable selection. To incorporate the tree structure into our candidate models, we use the leaf identifier as an input variable for successive models. For comparison, we also use the Variable Selection node, which implements a proprietary binning and regression technique. Our three modeling techniques in this exercise include the well known logistic regression as well as two new techniques in Enterprise Miner 5.3: Partial Least Squares and Gradient Boosting. These are relatively new techniques in the statistical and data mining communities. While we can follow some general rules, most often we do not know which modeling technique will produce the most accurate model until we have tried several of them. Model selection also depends on the statistical criterion, which is determined by the business need. In this case, because we want to identify the most likely responders, it is reasonable to use the value of Captured Response at

10 % of the sample. In other words, we choose the model that provides the most benefit when selecting 10% of our sample for receiving the promotional offer. At this criterion, the models that were produced using the proprietary variable selection technique performed best; and among those, the Partial Least Squares model (PLS) is selected as the champion.



Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Valid: Percent Capture Response	Trai Aka Info Crit
Y	PLS2	PLS2	Partial Lea...	target_purc...	6.836786	
	Boost2	Boost2	Gradient Bo...	target_purc...	6.314286	
	Reg2	Reg2	Regression...	target_purc...	6.3	

Figure 17: Best fit details retrieved from the analysis

Data Miners often need to prepare reports for management, archive models for auditors, and transfer the model logic back to the data builders for production model scoring. The new Reporter node in Enterprise Miner 5.3 produces a PDF file detailing the input data set, variables, model functions, parameters, model outputs, and exploratory and diagnostic plots. This report is a readable document that is appropriate for managers, colleagues, and auditors.

For production scoring jobs, Enterprise Miner registers the model to the SAS Metadata Server where it can be accessed by both SAS Model Manager and Data Integration Studio, which contains a model scoring transformation as was described earlier in this paper. The user can interactively build the production scoring data, select a named model, map data columns to model input variables, and design the target table containing the model output fields.

This functionality completes the data management and model development cycle.

OPTIMIZING SAS AND TERADATA WHEN WORKING WITH ANALYTICAL DATA

The general information and SAS descriptions in this section are common to all the major databases. The content of this section contains practices and recommendations for Teradata; however, you can extrapolate general best practices or usage scenarios to a database of your choice. SAS Data Integration Studio will generate SAS code to effectively move data, perform a join, and perform other manipulations on data. The rudimentary coding examples below are provided to illustrate processing conditions you might encounter and considerations that apply to generated code as well.

Unfortunately, for many data miners, their data access patterns were not considered in the logical/physical design or usage patterns of the database and database tables that they must analyze. They must rely on solutions and tools that allow them to mine and analyze data in a dynamic fashion. Knowing how to use this software and their database efficiently could be the difference between success and failure for their analysis. Data Miners are not usually DBAs and usually do not have the access or ability to adjust database design or functionality, especially around corporate data. Given this limitation we will focus on what they can do with SAS using Teradata to help them succeed.

Teradata exceeds at processing queries in parallel. If requests for results can be fanned out over all processing nodes, then results can be returned very quickly. Part of the success of the environment is partitioning the data equitably across all the processing nodes on storage devices that they control. To do this you need to choose a partitioning key for your data. In many Data Mining instances this has already occurred.

How can you use SAS to obtain information about the Teradata database tables? The SAS code below shows you how to gather information about the data you will analyze:

```
/*--- establish SAS connection to the Teradata database server---*/
libname tddata teradata user=<uid> password=<pid> server=<sid> <options>;

/*--- view all Teradata database tables that you can access from SAS---*/
proc datasets lib=tddata; run;

/*--- view attributes of the Teradata table as they will be represented to SAS---*/
/*--- datastep, applications or solutions – Note tabname is the table name        ---*/
/*--- gleaned form the proc datasets output above                                 ---*/
proc contents data=tddata.tabname; run;

/*--- gather specific metadata information about the individual tables of interest  ---*/
proc sql;
        connect to teradata(user=<uid> password=<pid> server=<sid>);

        quit;
```

You can use this information to validate queries that you write or ones that are generated for you and processed by SAS. The SAS options and code below allow you to track SQL processed on the database:

```
/* This section provides examples on how you can trace the interactions  */
/*--- between SAS and Teradata.  This option will show you the SQL that is      ---*/
/*--- being generated and passed to the DBMS                                     ---*/
```

```
/*--- on windows set the following option                              ---*/
options sastrace=',,d' sastraceloc=saslog;

/*--- on UNIX and the Mainframe set the following option              ---*/
options sastrace=',,d';
```

When you set these options you will generate trace logs of the SAS interaction with the Teradata server. There are other options in the SAS/ACCESS documentation that allow you to track processing time between steps. This information will help you track performance issues in the data access portion of your analysis. Once you have performed the necessary analysis you can turn off the log information using the following SAS option.

```
/*--- reset trace all platforms      ---*/
options sastrace=',,,,';
```

Identify data sources

When you begin your data preparations it is important to consider where your data sources are. If you have heterogeneous data sources, where some of the data exists in Teradata and some exists in SAS or other databases then you might encounter performance processing complications. For example, if you are joining these data sources together and SAS is doing the join, it would require all of the Teradata data to be extracted into SAS and then processed against SAS or other data sources. SAS resolves heterogeneous joins in this fashion, which can lead to long processing times if the database tables are large. How you control this process directly impacts performance. The SAS code below exhibits the problem and details a solution.

```
/*--- identify where the data is coming from: sasds – SAS, tdtab – Teradata        ---*/
libname sasds <dataset options>;
libname tdtab teradata user=<uid> password=<pid> server =<sid> <options>;

/*--- create a sas dataset call newtab that is a join of a sas dataset and a teradata ---*/
/*--- table based on some where criteria.                                           ---*/
proc sql;
        create table newtab as
select * from sasds.sasdata a, tdtab.dbdata b where a.x = b.x;
quit;

/*--- create a homogeneous data environment                                         ---*/
data tdtab.sasdata (bulkload=yes <other SAS/ACCESS loading options>);
        set sasds.sasdata;
        run;

/*--- perform the join on the Teradata server and bring back a smaller result set   ---*/
proc sql;
        create table newtab as
                select * from tdtab.sasdata a, tdtab.dbdata b where a.x = b.x;
        quit;
```

Web logs joined with web session cookie information

This analysis project requires multiple joins or transformations to the data to produce the result set for analysis. In a multi-step job it is important to understand and test performance considerations against a subset of the data before processing the entire data base. You can process a sample set by using Teradata views of the large data table. The Teradata views can be created by the DBA, or, if you have database authority using SAS as shown in the example below.

```
/*--- create Teradata views of the large database tables                      ---*/
/*---      to build Teradata views we use explicit SQL and limit the view scope  ---*/
/*---      with a where clause                                                 ---*/
proc sql;
connect to teradata( user=<uid> password=<pwd> server=<sid>);
execute (create view x1 as select * from large_td_table where
                    Key_val > 10 and key_val < 5500) by teradata;
   quit;
```

Once the view is created you can proceed with code development on a much smaller subset of data. Many Teradata DBAs use views extensively to subset, join, and organize data for consumption.

Passing functions and WHERE clause components to the DBMS using SAS

Potentially, one of the biggest performance gains is to push aggregation and where predicates to the DBMS. To enable this pushdown SAS offers two forms of PROC SQL. The first is explicit SQL, as shown in the following SAS code execution.

```
/*--- execute PROC SQL code in explicit mode – what you see is what you get   ---*/
proc sql;
        connect to teradata (user=<uid> password=<pwd> server=<sid>);
        exec( drop table fred ) by teradata;
        exec( commit work ) by teradata;
        exec( create table fred … ) by teradata;
        select * from connection to teradata
                ( select * from pete where td(x) > 10) by teradata;
        quit;
```

Essentially, in this example you are communicating directly with the Teradata databaseserver with Teradata SQL. We would like to focus on the last statement before the quit in the sample code above. Here td(x) > 10 could represent a Teradata user-defined function (UDF), or Teradata function we have not mapped to an equivalent SAS function, where the evaluation of x determines whether the row is selected and returned to SAS. In previous releases of SAS only explicit SQL could process this type of request.

With modifications to SAS/ACCESS engines and PROC SQL in SAS 9.2 that limitation has been lifted. The following example and subsequent discussion provide an overview of some of these key processing capabilities.

```
/*--- execute PROC SQL code using SAS SQL (implicit SQL) – SAS will parse      ---*/
/*---       and textualize this SQL into ANSI SQL that is pushed to the DBMS           ---*/
libname tdcon teradata user=<uid> password=<pwd> server=<sid> <sql_functions>;

proc sql;

           /*--- use a teradata UDFor other Teradata function                                   ---*/
           create table work.a as
select * from tdcon.tdtab where td(x) > 10;

           /*--- use SAS format statements - dresssize is a SAS format that buckets ---*/
           /*---       values of size as types of dresses, in this case values of x           ---*/
           /*---       4,5 and 6 are classified as small                                       ---*/
           create table work.b as
                      select * from tdcon.tdtab where put(x, dresssize) = 'small';
           quit;
```

The importance of this PROC SQL execution is potential for significant performance gains.

In SAS 9.1 the Teradata SAS/ACCESS engine, among others, supports the concept of an SQL Dictionary. This dictionary is used to validate functions that can be passed to the database and to map SAS functions to database functions. Current rules require that the function specified must exist in SAS.

In the first example the td(x) function does not exist in SAS. Therefore, with current releases of SAS this statement would be flagged as an error and you would have to revert to explicit SQL.

To support database-specific functions or UDFs a method was developed in SAS 9.2 that would allow you to update the internal SQL Dictionary within the SAS/ACCESS engine. The way that we process SQL was modified to permit pushing functions other than SAS functions directly to the DBMS, as in the example above. The SAS 9.2 documentation details the specifics on the dynamic SQL dictionary capabilities, information on the SQL_functions libname option and enhanced SQL processing. For this example the focus is on what you need to do to execute the td function in the database. Note this is a one-time process in that if failure occurs the SQL statement will be rejected. There is no SAS processing that can take place when a function other than SAS exists. The process is as follows:

- Add the td function to the SAS/ACCESS SQL Dictionary (see SAS/ACCESS documentation for processing details).

- Add the sql_functions option to the libname statement (see SAS documentation for details).
- Execute the implicit passthru statement, as shown above.

In the second example a SAS function is interjected into the SQL statement, specifically, the PUT function. With current releases of SAS the PUT function would be processed by SAS. This would require that the entire tdtab table be extracted into SAS. The PUT function would then be applied by SAS to the data, which would result in a segmented data set. With SAS 9.2, the PUT function can be decomposed into SQL components that can be passed directly to the DBMS. Therefore,in the PUT function above, given the definition of small in the user defined format, would be transposed into x=4 or x=5 or x=6. The SQL statement passed to the DBMS in this example would resemble the following statement:

```
select * from tdtab where x=4 or x=5 or x=6
```

This statement can be processed by Teradata with the potential of a much smaller result set returned to SAS. This example shows the significance of decomposing functionality that is specific to SAS into SQL predicates that can be pushed directly to the database for processing. There are many more formats in SAS 9.2 that can be processed into simple SQL. Please refer to the SAS 9.2 documentation for a comprehensive list.

CONCLUSION

We have now demonstrated some basic practices for managing data for analytical performance. The entire task of building data was delegated to SAS Data Integration Studio, which both reduces workload for the data miner and ensures the creation of a robust and repeatable data aggregation. This is most important when it is time for internal and external compliance reviews and for building production model scoring jobs. Enterprise Miner was used to build the remaining needed data transformations, select terms, and build models that predict the needed behavior. You can also use SAS Access and SAS PROC SQL to pass through data intensive aggregation operations to a target database, which will improve overall performance and minimize data transfer. Fortunately the SAS system provides all these functionalities, meaning that it’s a good time to be a data analyst.

REFERENCES

Friedman, Jerome H. 2002. “Stochastic Gradient Boosting.” *Computational Statistics & Data Analysis,* 38, 367-378.

Tobias, R. 1995. "An Introduction to Partial Least Squares Regression." *Proceedings of the Twentieth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc, Svolba, G. 2007. Data Preparation for Analytics Using SAS. SAS Institute, Inc. Cary, NC.

SAS Institute, What's New in SAS Enterprise Miner 5.3. 2007. SAS Institute, Inc. Cary, NC. Available at http://support.sas.com/documentation/whatsnew/91x/emgui53whatsnew.htm

SAS(R) Enterprise Miner™ 5.3 FACT SHEET. 2007, SAS Institute Inc., Cary, NC. Available at http://www.sas.com/technologies/analytics/datamining/miner/factsheet.pdf

Hunley, Eric, Gary Mehler, and Nancy Rausch, "A Whole New World: What's New in SAS(R) Data Integration Studio 4.2." *Proceedings of the SAS Global Forum 2008 Conference*. Cary NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008

SAS OnlineDoc 9.1.3, 2007. SAS Institute, Inc. Cary, NC. Available at http://support.sas.com/onlinedoc/913/docMainpage.jsp

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

David Duling,
SAS Institute, Inc.
Cary, NC 27513
David.Duling@sas.com

Nancy Rausch,
SAS Institute, Inc.
Cary, NC 27513
Nancy.Rausch@sas.com

Howard Plemmons
SAS Institute, Inc.
Cary, NC 27513
Howard.Plemmons@sas.com