Paper SAS2204-2018

Make SAS® Enterprise Miner™ Play Nicely with SAS® Viya®

Beth Ebersole and Wendy Czika, SAS Institute Inc.

ABSTRACT

With a few simple tricks, your SAS® Enterprise Miner on SAS® 9 will run seamlessly with your new SAS® Viya® installation. You can enjoy the perks of your old, familiar interface and your existing SAS Enterprise Miner projects, PLUS the advantage of faster, distributed, multi-threaded processing on SAS Viya. We will walk you through all the steps needed to make this work effortlessly, and we provide the code that will help you. Why choose, when you can have the advantages of both SAS Enterprise Miner AND SAS Viya?

INTRODUCTION

SAS Enterprise Miner was introduced in 1998. This product brought a user-friendly drag-and-drop graphical user interface to model development. SAS Enterprise Miner also introduced the SEMMA model development framework. SEMMA organized the tasks in the data mining process in this form.

S: Sampling

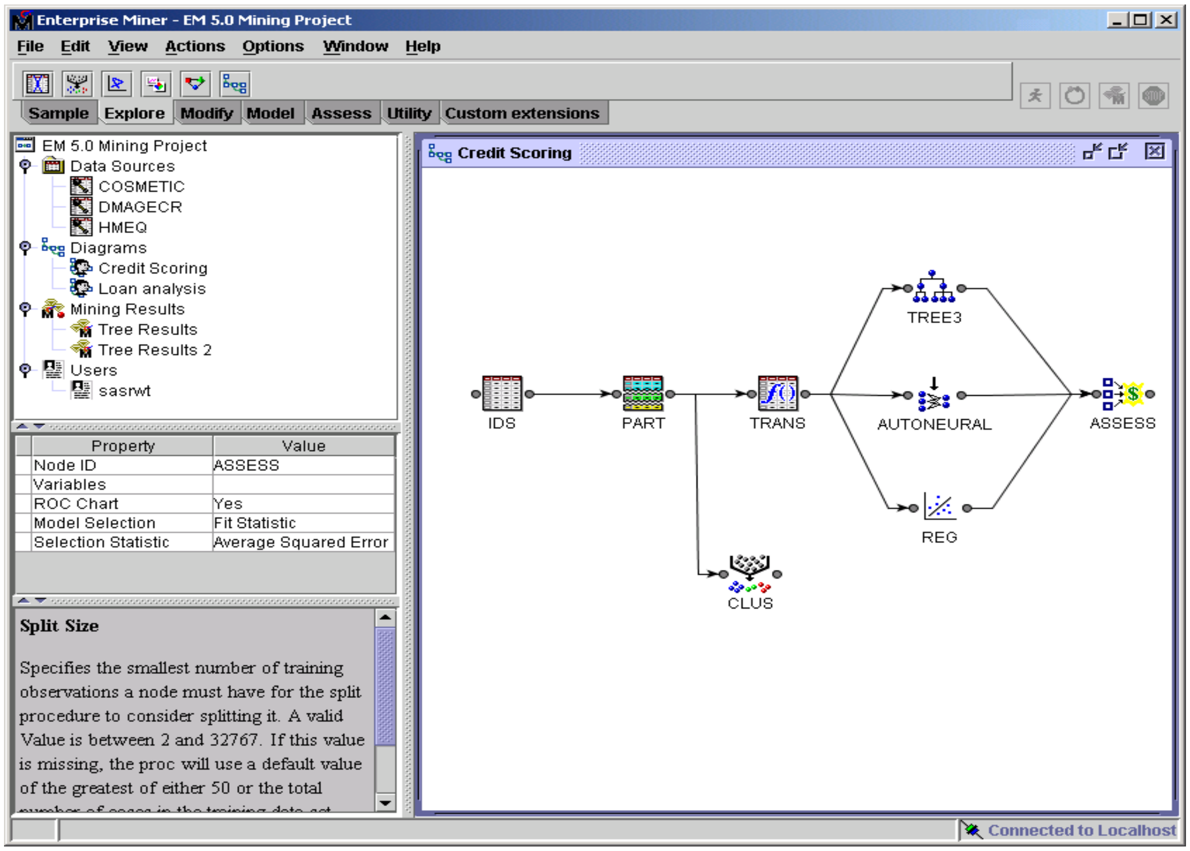
E: Exploring

M: Modifying

M: Modeling

A: Assessing

SAS Enterprise Miner facilitated creating models and comparing the results of your models by using a variety of assessment criteria.

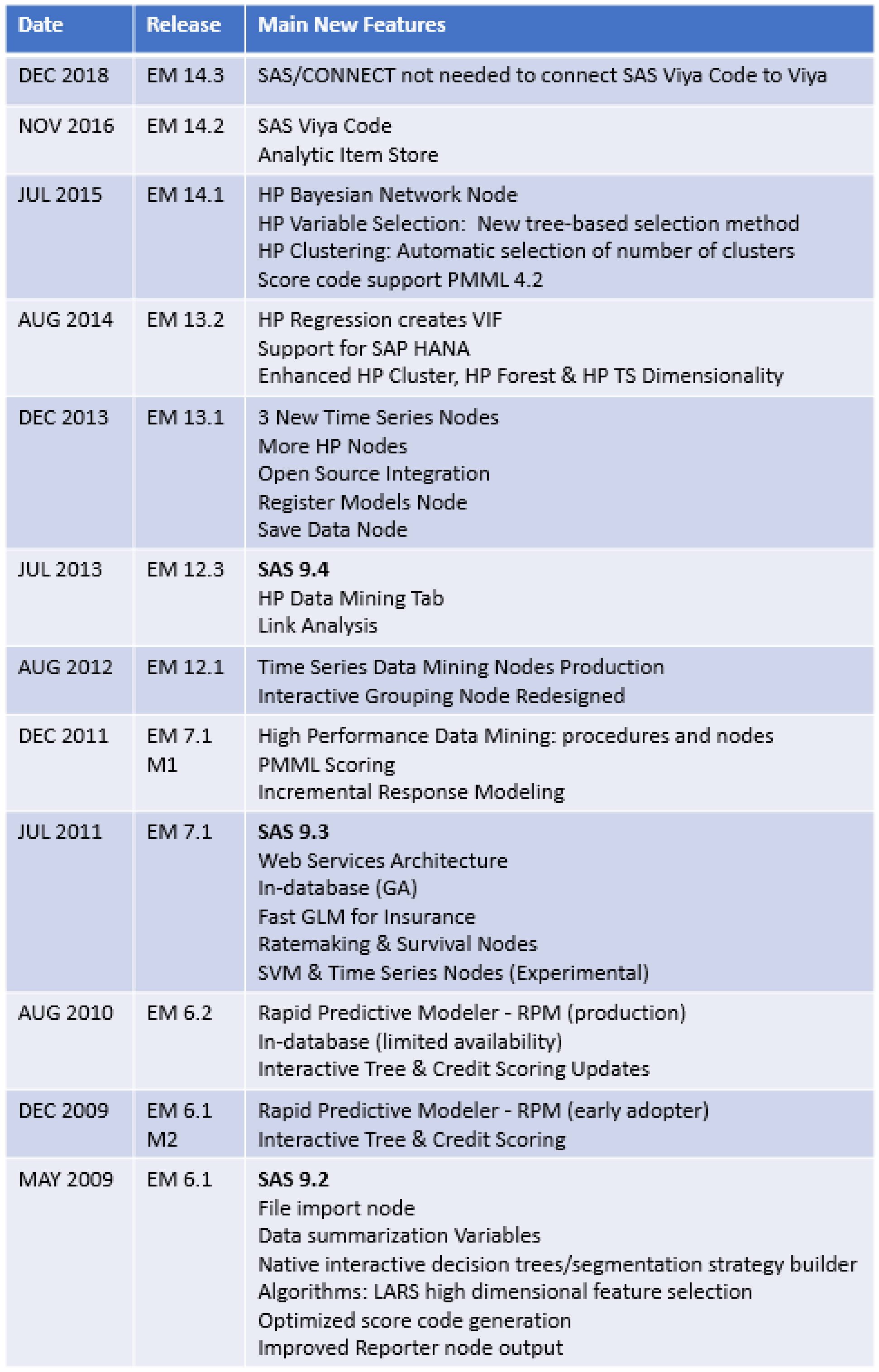


Property	Value
Node ID	ASSESS
Variables	
ROC Chart	Yes
Model Selection	Fit Statistic
Selection Statistic	Average Squared Error

Split Size

Specifies the smallest number of training observations a node must have for the split procedure to consider splitting it. A valid Value is between 2 and 32767. If this value is missing, the proc will use a default value of the greatest of either 50 or the total

SAS Enterprise Miner has evolved over the years, as shown in the table below (courtesy of Wayne Thompson and Melodie Rush).



Date	Release	Main New Features
DEC 2018	EM 14.3	SAS/CONNECT not needed to connect SAS Viya Code to Viya
NOV 2016	EM 14.2	SAS Viya Code Analytic Item Store
JUL 2015	EM 14.1	HP Bayesian Network Node HP Variable Selection: New tree-based selection method HP Clustering: Automatic selection of number of clusters Score code support PMML 4.2
AUG 2014	EM 13.2	HP Regression creates VIF Support for SAP HANA Enhanced HP Cluster, HP Forest & HP TS Dimensionality
DEC 2013	EM 13.1	3 New Time Series Nodes More HP Nodes Open Source Integration Register Models Node Save Data Node
JUL 2013	EM 12.3	**SAS 9.4** HP Data Mining Tab Link Analysis
AUG 2012	EM 12.1	Time Series Data Mining Nodes Production Interactive Grouping Node Redesigned
DEC 2011	EM 7.1 M1	High Performance Data Mining: procedures and nodes PMML Scoring Incremental Response Modeling
JUL 2011	EM 7.1	**SAS 9.3** Web Services Architecture In-database (GA) Fast GLM for Insurance Ratemaking & Survival Nodes SVM & Time Series Nodes (Experimental)
AUG 2010	EM 6.2	Rapid Predictive Modeler - RPM (production) In-database (limited availability) Interactive Tree & Credit Scoring Updates
DEC 2009	EM 6.1 M2	Rapid Predictive Modeler - RPM (early adopter) Interactive Tree & Credit Scoring
MAY 2009	EM 6.1	**SAS 9.2** File import node Data summarization Variables Native interactive decision trees/segmentation strategy builder Algorithms: LARS high dimensional feature selection Optimized score code generation Improved Reporter node output

Date	Release	Main New Features
DEC 2007	EM 5.3	**SAS 9.1.3** Updated PFD Group Processing Algorithms: Gradient Boosting, Variable Clustering, & Hierarchical Associations Native interactive binning for credit scoring (easy admin and improved usability) Graphical data exploration and persistence EM 4.3 Project Conversion Utility Interactive SAS Code node
NOV 2005	EM 5.2	Decision, Replacement & SOM/Kohonen nodes Transform Formula Builder Interactive selections for outliers , market baskets, principal components SAS Code node training and scoring editors with drag and drop macros Exported data node property New path plots EM SMC plug in Grid Processing EM Desktop release
2003	EM 5.1	**SAS 9.1** Rich Java Client/SAS Server Parallel and Batch processing XML diagram exchange Model packages Graph explorer Credit Scoring nodes
2002	TM 3.1	Text Miner Add-on (SAS is first DM vendor to integrate text with structured data)
2001	EM 4.1	Link Analysis Memory-Based Reasoning (MBR) Time Series
2000	EM 4.0	**SAS 8.2** C & Java Scoring Code Tree Desktop Viewer DMTOOL application
1998	EM 2.01	**EM Inaugural Release on SAS 6.12** Client Server Data Mining Workbench Windows and Unix Servers Process flow diagram drag-and-drop GUI based on SEMMA model development process Integrated model comparison SAS scoring logic including preliminary transformations
1996	Pre-EM	SAS develop Neural Network Application (NNA)
1994	Pre-EM	%TNN feed forward and %TREEDISC macros by Warren Sarle

Currently, SAS Enterprise Miner boasts many vehemently loyal users worldwide. SAS Enterprise Miner 14.3 is the latest iteration of the application. This version enables you to easily run models in SAS Viya, without requiring SAS/CONNECT (which was required to run SAS Enterprise Miner 14.2 with SAS Viya). SAS Viya's SAS Cloud Analytic Services (CAS) is a cloud-enabled, in-memory analytics engine. It lets you run advanced statistical and machine learning algorithms such as neural networks, support vector machines, logistic regression, and so on, **in parallel**. Not only can you run separate algorithms at the same time, but each algorithm can be broken up and run in parallel within itself. This means you can process large data sets very quickly, even when working with complex models. This means that data scientists get results more quickly, enabling them to get more accurate results.

START TO FINISH DEMO OF SAS ENTERPRISE MINER WITH SAS VIYA

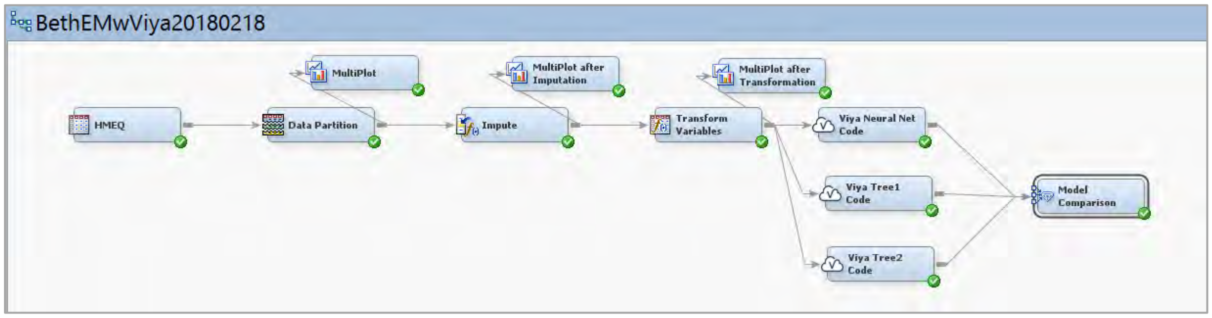
GET STARTED

The first step is to ensure that you have a working environment with SAS Enterprise Miner 14.3 properly configured to work with SAS Viya 3.3. Be aware that just accepting the default settings for CAS in a SAS Viya 3.3 deployment does not automatically configure access to SAS Viya from SAS Enterprise Miner. Be sure to properly address matters related to security, certificates, encryption, credentials, and _authinfo files. For details and instructions on proper configuration see Mark Thomas's blog *Configuring Enterprise Miner to Chat with CAS*, which is listed in the References section at the end of this paper.

Objective: We will now walk through the steps to build an Enterprise Miner project that performs these functions.

- Brings in, explores, partitions, imputes, and transforms our data.
- Uses the **SAS Viya Code** node to run a neural network and two decision tree models in CAS and return the results to SAS Enterprise Miner.
- Uses SAS Enterprise Miner to compare the results from both SAS Enterprise Miner and CAS!

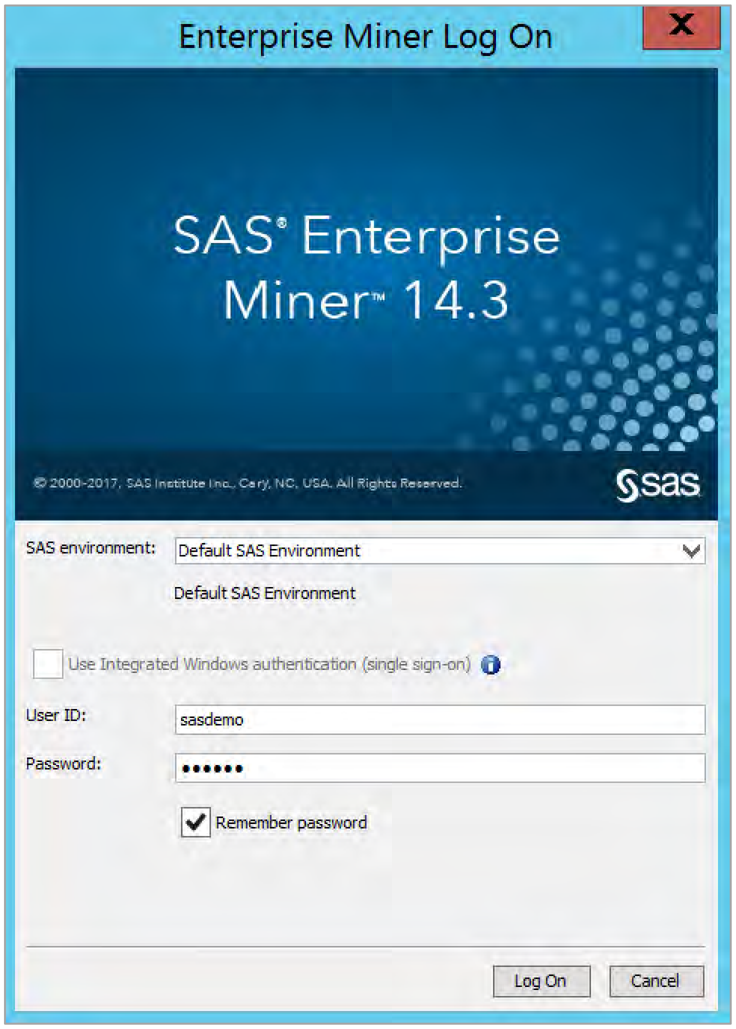
This is what our final SAS Enterprise Miner diagram will look like.



OPEN SAS ENTERPRISE MINER 14.3

SAS Enterprise Miner 14.3 is a java-based interactive graphical user interface (GUI) for data scientists that streamlines the data mining process. It is easy to use, and this paper provides step-by-step instructions for a start-to-finish demonstration.

To open SAS Enterprise Miner, select the SAS Enterprise Miner Client 14.3 shortcut from the Windows **Start** menu, or search for SAS Enterprise Miner. Enter your SAS Enterprise Miner user name and password.


Enterprise Miner Log On
SAS® Enterprise Miner™ 14.3
© 2000-2017, SAS Institute Inc., Cary, NC, USA. All Rights Reserved.
sas
Log On
Cancel

SAS environment: Default SAS Environment

Default SAS Environment

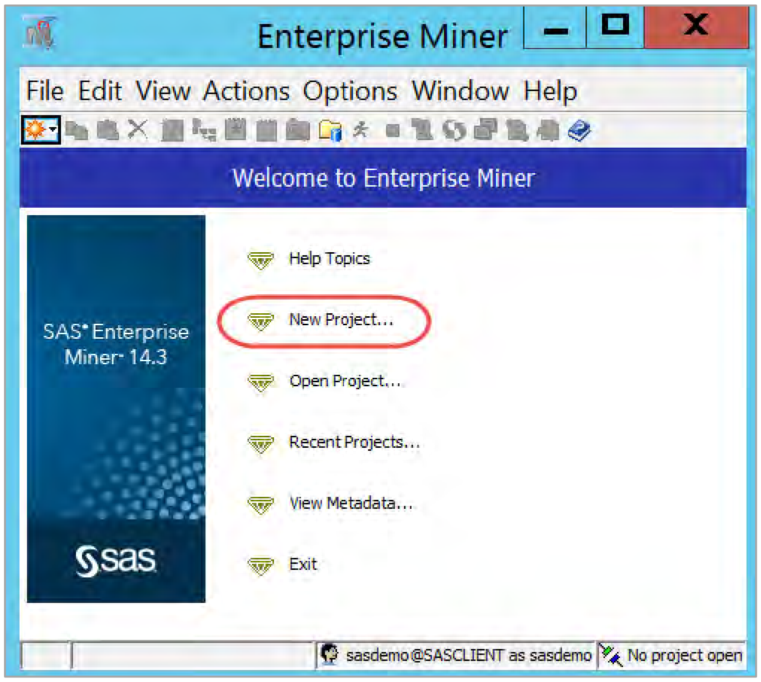
Use Integrated Windows authentication (single sign-on)

User ID: sasdemo

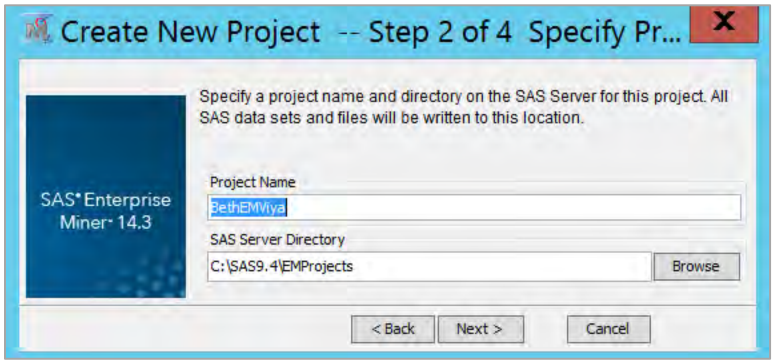
Password: ••••••

✓ Remember password

Click **New Project**.


Enterprise Miner
File Edit View Actions Options Window Help
Welcome to Enterprise Miner
SAS® Enterprise Miner™ 14.3
Help Topics
New Project...
Open Project...
Recent Projects...
View Metadata...
Exit
sasdemo@SASCLIENT as sasdemo
No project open

If you haven't done so already, you need to create a folder to store your SAS Enterprise Miner projects. In our case, we are storing projects in C:\SAS9.4\EMProjects.

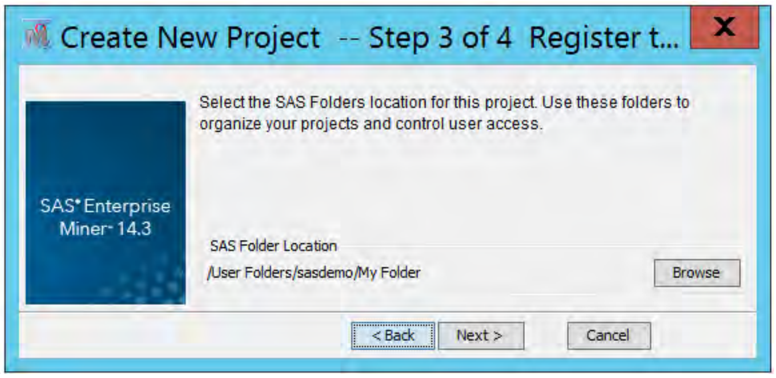

Create New Project -- Step 2 of 4 Specify Pr...
SAS® Enterprise Miner™ 14.3
< Back
Next >
Cancel
Browse

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
BethEMViya

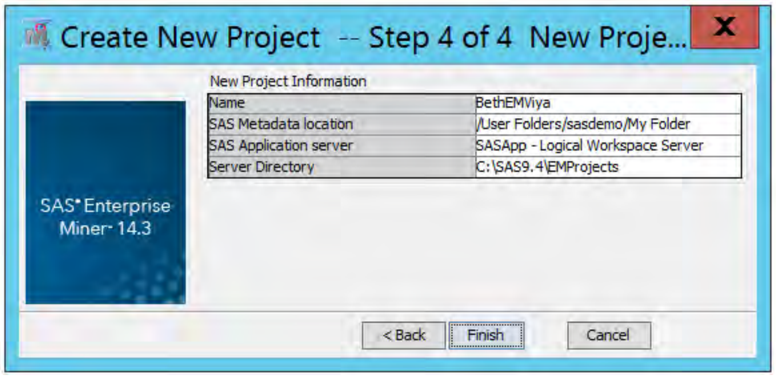
SAS Server Directory
C:\SAS9.4\EMProjects

We use the default value for **SAS Folder Location**. Accept the default values in the remaining steps in the wizard and click Finish to complete the process.


Create New Project -- Step 3 of 4 Register t...
SAS® Enterprise Miner™ 14.3
< Back
Next >
Cancel
Browse

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

SAS Folder Location
/User Folders/sasdemo/My Folder



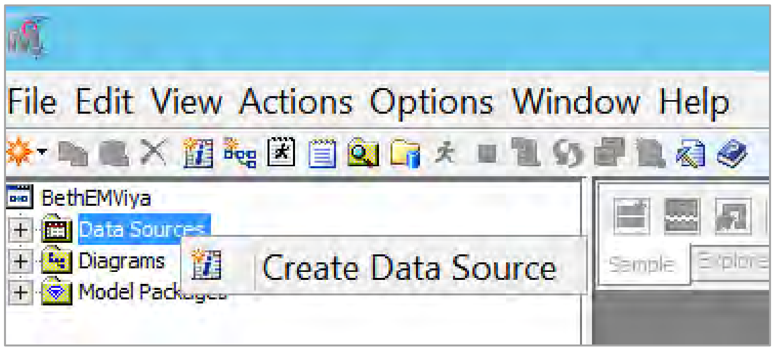
Create New Project -- Step 4 of 4 New Proje...

New Project Information

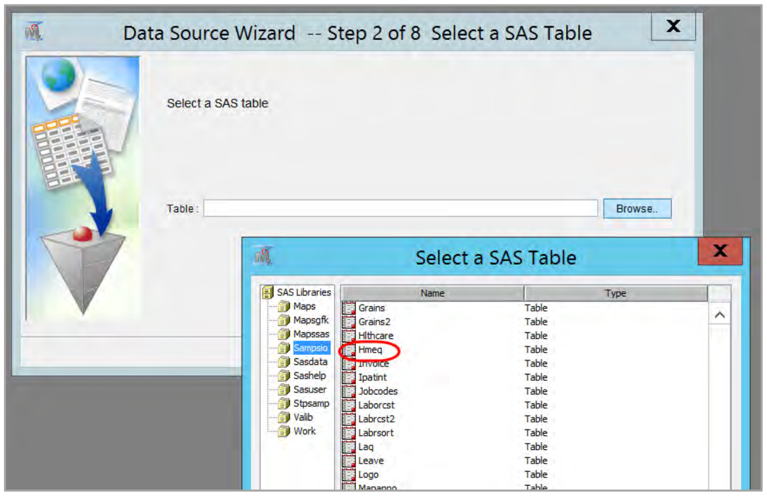
Name	BethEMViya
SAS Metadata location	/User Folders/sasdemo/My Folder
SAS Application server	SASApp - Logical Workspace Server
Server Directory	C:\SAS9.4\EMProjects

BRING IN A SAMPLE DATA SET

Right-click **Data Source** and select **Create Data Source** to start the **Data Source Wizard**.



In step 2 of the wizard, open **SAS Libraries**, double-click **Sampsio,** and select the **Hmeq** data set. Click **Next**.



Data Source Wizard -- Step 2 of 8 Select a SAS Table

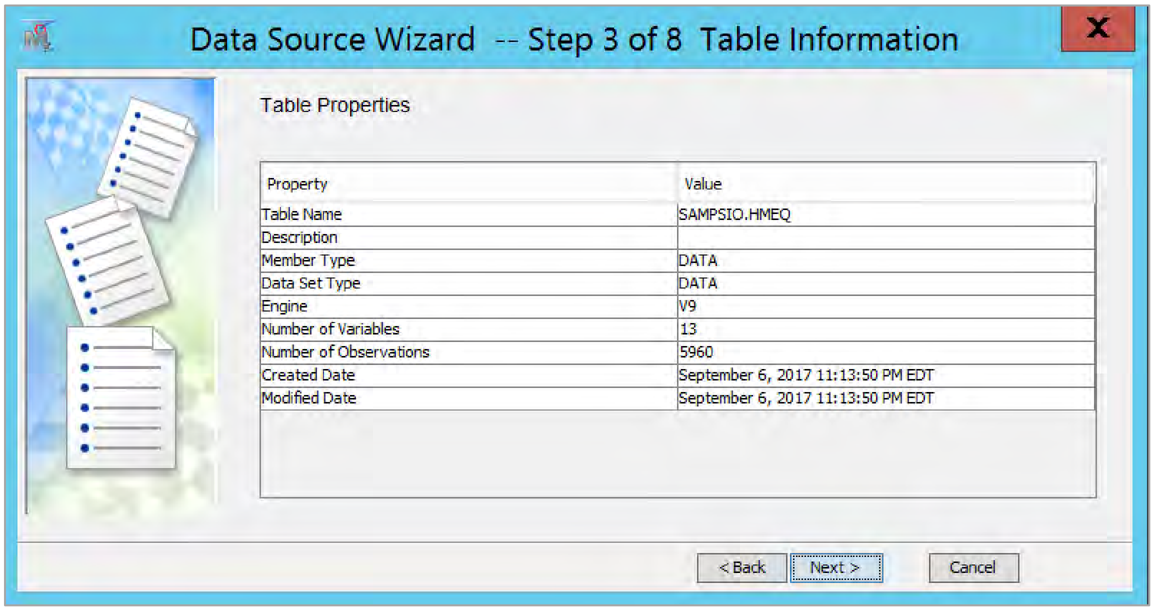
Select a SAS table

Table :

Select a SAS Table

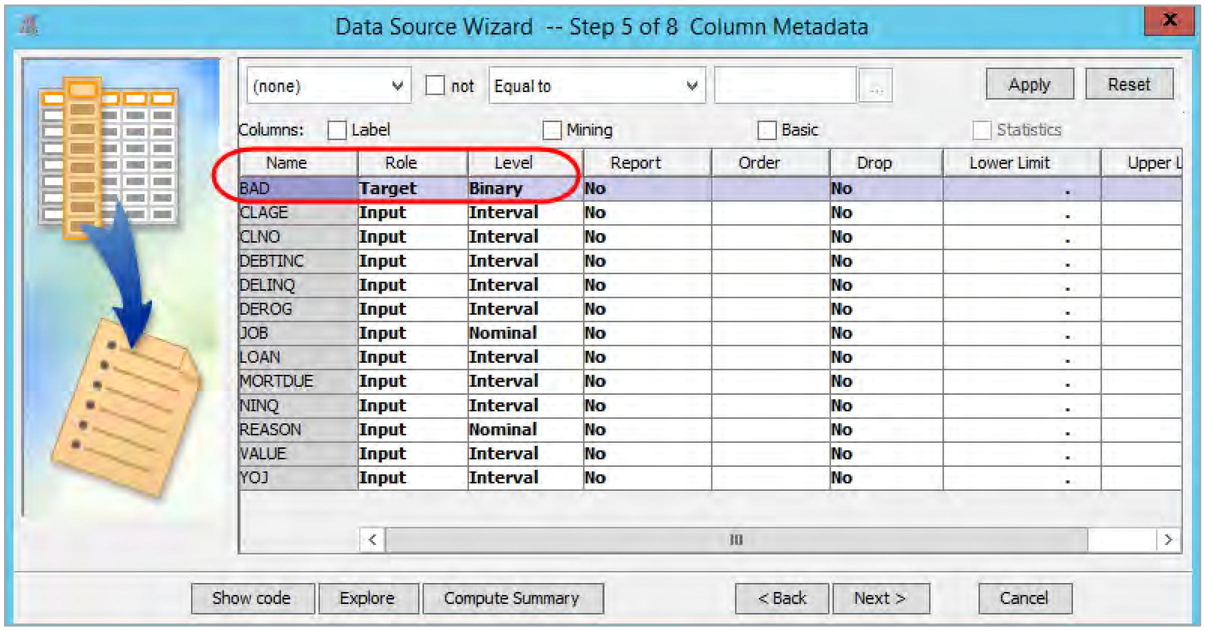
Name	Type
Grains	Table
Grains2	Table
Hlthcare	Table
Hmeq	Table
Invoice	Table
Ipatint	Table
Jobcodes	Table
Laborcst	Table
Labrcst2	Table
Labrsort	Table
Laq	Table
Leave	Table
Logo	Table

In step 3 of the wizard, we see that the HMEQ data set has 13 variables (columns) and 5960 observations (rows).



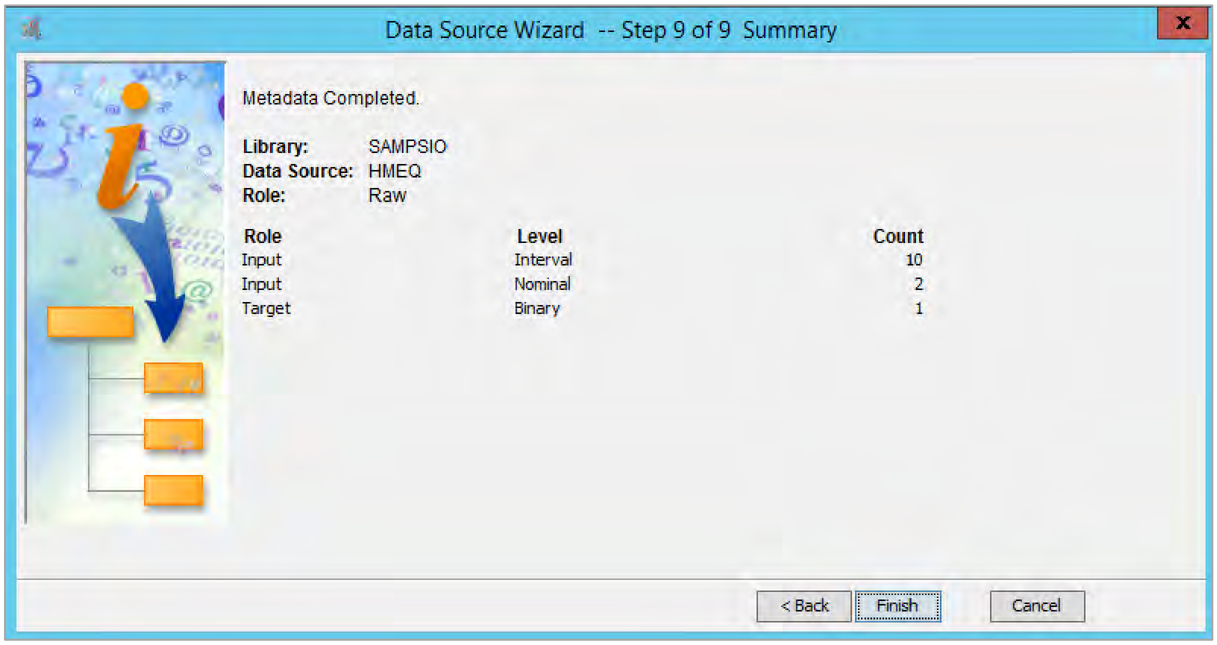
Property	Value
Table Name	SAMPSIO.HMEQ
Description	
Member Type	DATA
Data Set Type	DATA
Engine	V9
Number of Variables	13
Number of Observations	5960
Created Date	September 6, 2017 11:13:50 PM EDT
Modified Date	September 6, 2017 11:13:50 PM EDT

The HMEQ data set is a historic data set with information about people who defaulted on their home equity loan (BAD=1) or did not default (BAD=0). Input variables include information about debt to income ratio (DEBTINC), delinquent payments (DELINQ), and so on. At Step 4 of the Data Source Wizard, use the default selection of **Basic**. At Step 5, set the variable **BAD** as a binary target variable.



Name	Role	Level	Report	Order	Drop	Lower Limit	Upper L
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	.	

Continue through the **Data Source Wizard**, accepting the default values. We bring the data in as a **Raw** data set. Notice at step 9 of the wizard that we have ten interval inputs, two nominal inputs, and one binary target variable. Click **Finish**.



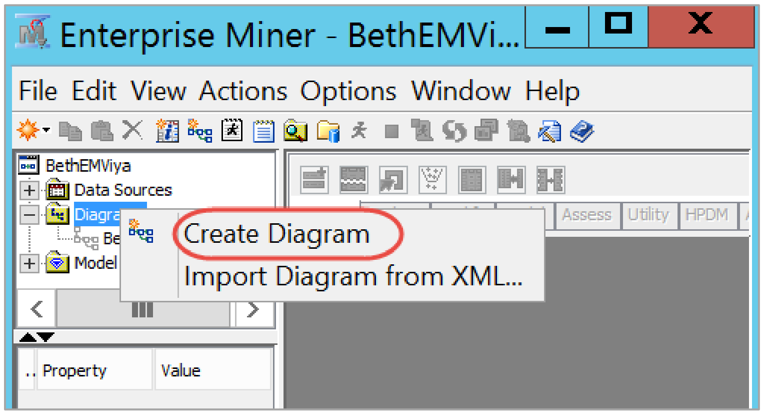
Metadata Completed.

Library: SAMPSIO
Data Source: HMEQ
Role: Raw

Role	Level	Count
Input	Interval	10
Input	Nominal	2
Target	Binary	1

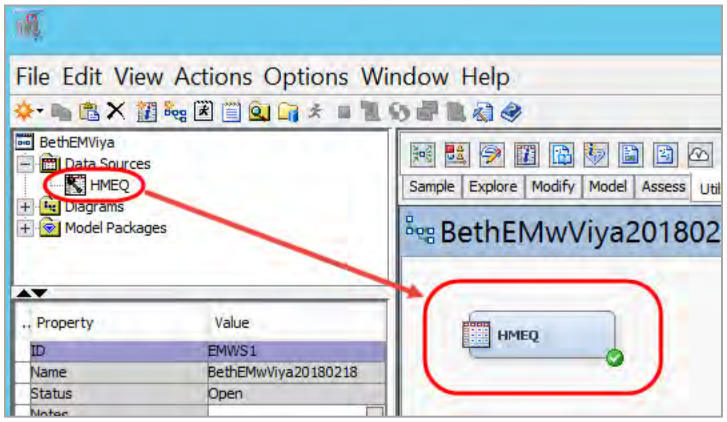
CREATE DIAGRAM

In SAS Enterprise Miner, right-click **Diagrams** and select **Create Diagram.** Next we name our diagram (this example is named **BethEMViyaDate**) and click **OK**.


Enterprise Miner - BethEMVi...
File Edit View Actions Options Window Help
BethEMViya
Data Sources
Diagra
Be
Model
Assess
Utility
HPDM
Create Diagram
Import Diagram from XML...
Property
Value

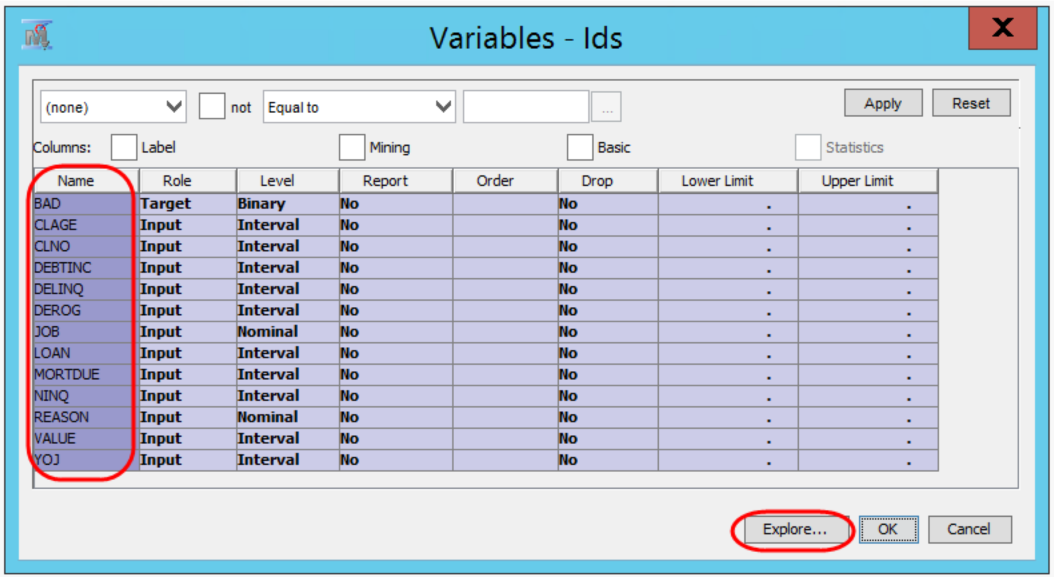
EXPLORE DATA

Drag the **HMEQ** data set into the diagram.


File Edit View Actions Options Window Help
BethEMViya
Data Sources
HMEQ
Diagrams
Model Packages
Sample Explore Modify Model Assess Util
BethEMwViya201802
HMEQ

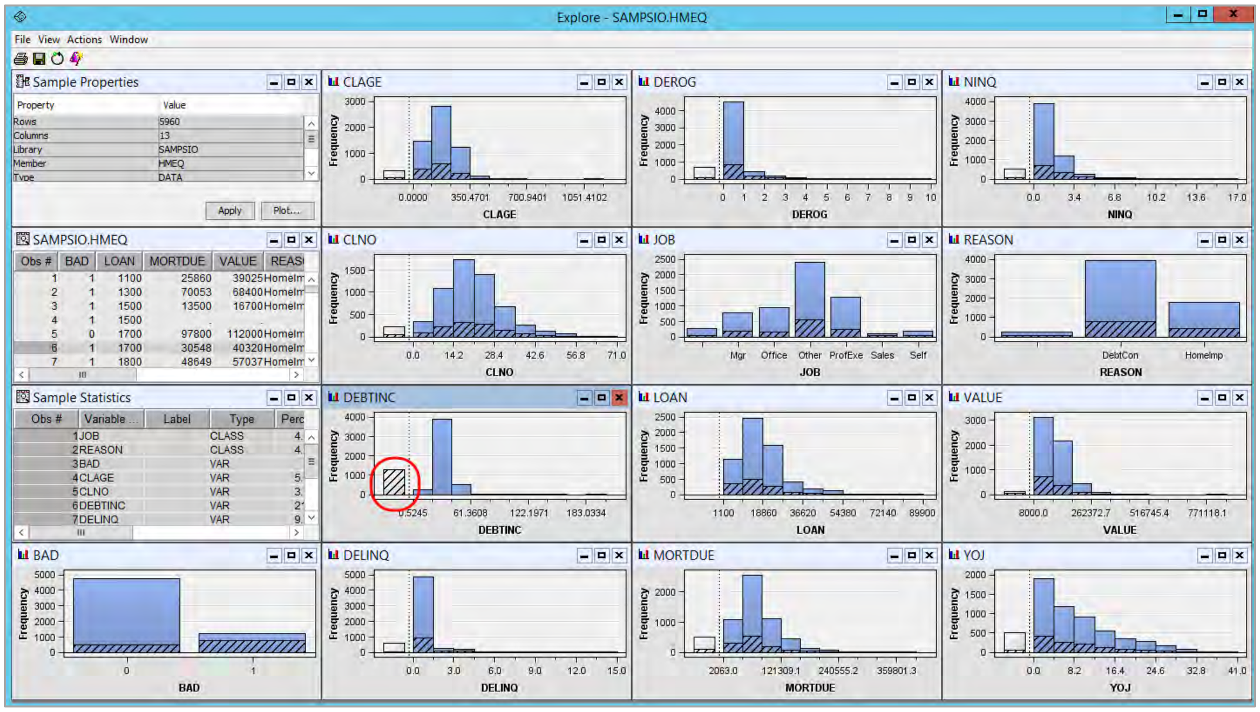
Property	Value
ID	EMWS1
Name	BethEMwViya20180218
Status	Open
Notes	

Right-click the **HMEQ** data node and click **Edit Variables**. Select all variables (to do this, select the first variable **BAD**, hold down the Shift key, and select the last variable **YOJ**). Click the **Explore…** button as illustrated below.



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	.	.

We see a histogram for each variable. Notice the high number of missing values for the **DEBTINC** variable, as indicated by the first white bar on the left side of the histogram. Click this white bar to see how the observations that are missing from the DEBTINC variable are distributed in the other histograms (shown by cross hatching), as in the screenshot below.



Sample Properties

Property	Value
Rows	5960
Columns	13
Library	SAMPSIO
Member	HMEQ
Type	DATA

SAMPSIO.HMEQ

Obs #	BAD	LOAN	MORTDUE	VALUE	REAS
1	1	1100	25860	39025	HomeIm
2	1	1300	70053	68400	HomeIm
3	1	1500	13500	16700	HomeIm
4	1	1500	.	.	
5	0	1700	97800	112000	HomeIm
6	1	1700	30548	40320	HomeIm
7	1	1800	48649	57037	HomeIm

Sample Statistics

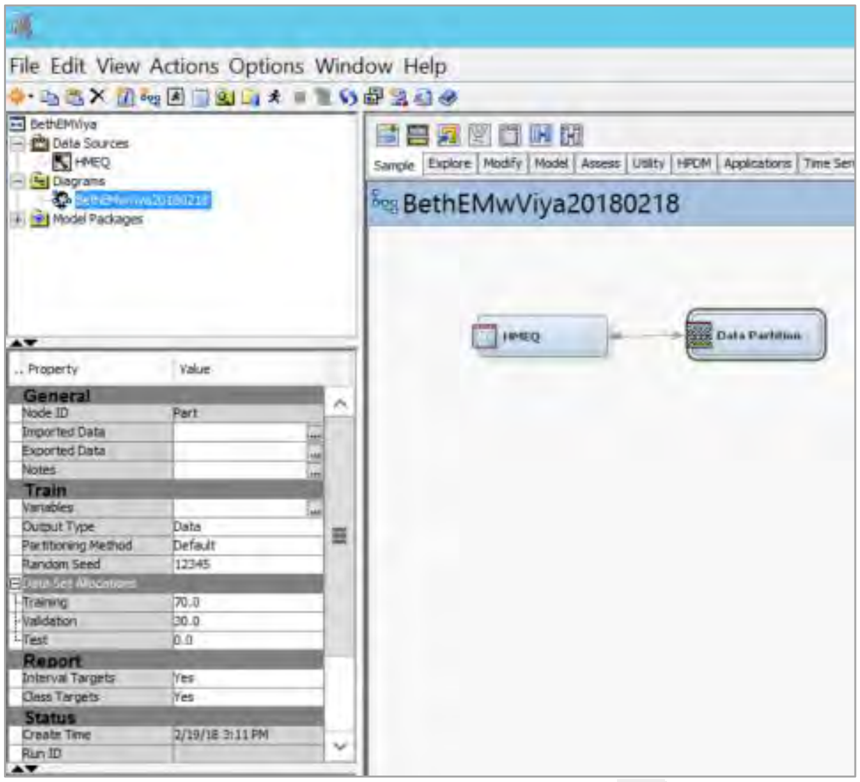
Obs #	Variable ...	Label	Type	Perc
1	JOB		CLASS	4.
2	REASON		CLASS	4.
3	BAD		VAR	
4	CLAGE		VAR	5.
5	CLNO		VAR	3.
6	DEBTINC		VAR	2
7	DELINQ		VAR	9.

Close the **Explore** pane and the **Variables-Ids** pane.

PARTITION DATA

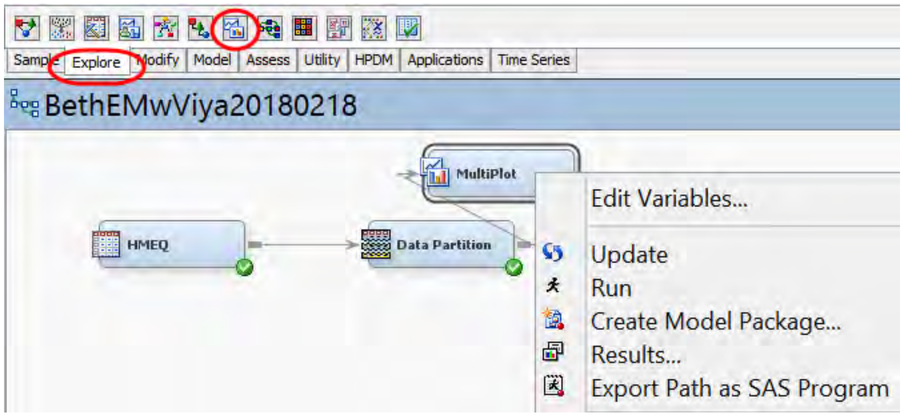
The next step is to partition the data. Although it is best practice to partition the data into training, validation, and test data sets, for simplicity's sake in this example we will use only the training and validation data sets.

Select the **Sample** tab and drag the **Data Partition** icon onto the diagram. Connect the **HMEQ** data node to the **Data Partition** node. Select the **Data Partition** node and change the values of **Data Set Allocations** to **Training: 70.0**, **Validation: 30.0**, and **Test: 0.0**.

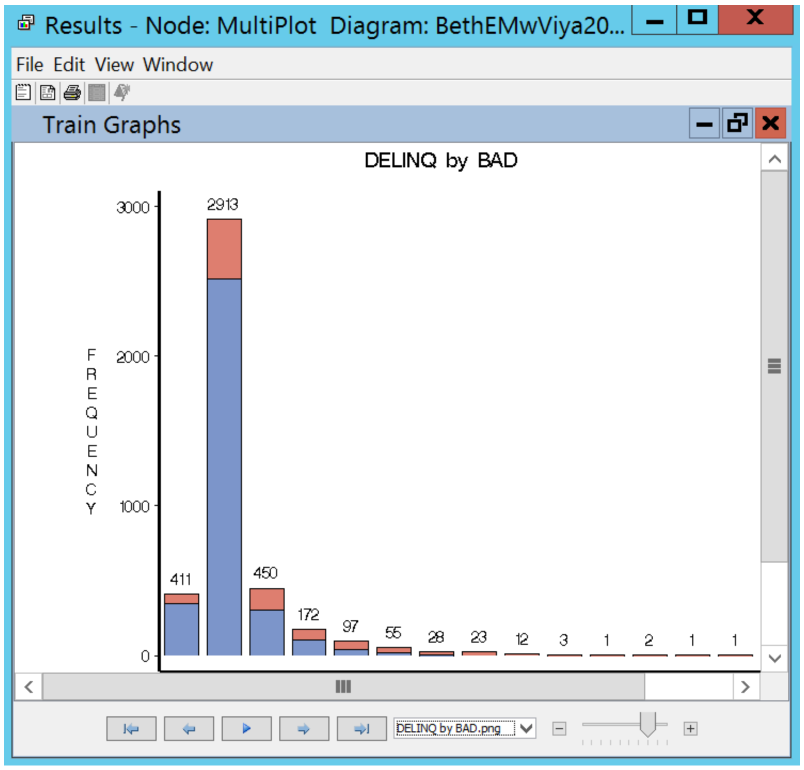


Property	Value
General	
Node ID	Part
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Output Type	Data
Partitioning Method	Default
Random Seed	12345
Data Set Allocations	
Training	70.0
Validation	30.0
Test	0.0
Report	
Interval Targets	Yes
Class Targets	Yes
Status	
Create Time	2/19/18 3:11 PM
Run ID	

Open the **Explore** tab, and drag the **MultiPlot** icon onto the diagram. Connect the **HMEQ** data node to the **MultiPlot** node. Right-click **MultiPlot**, and select **Edit Variables** to see the variables coming into the **MultiPlot** node. Close the **Variables – Plot** pane. Right-click the **MultiPlot** node again, and click **Run** to run the path.

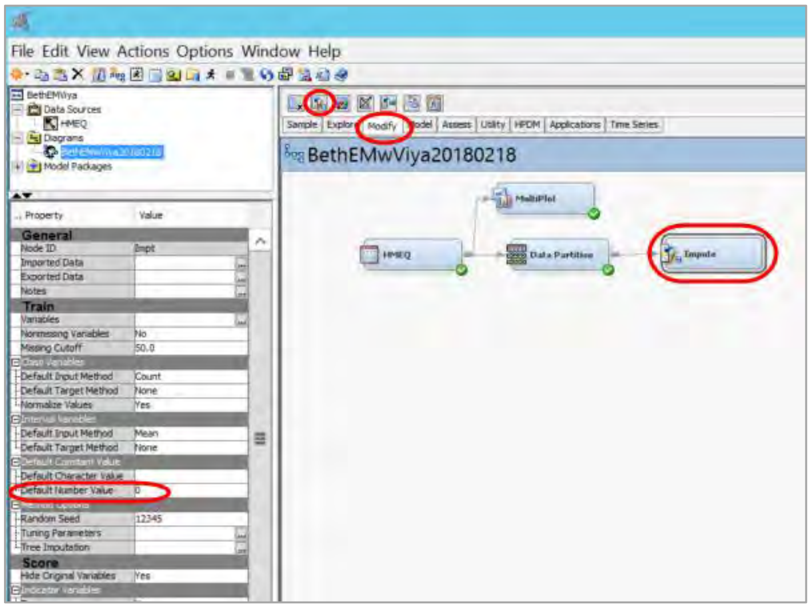


When the pop-up menu appears, click **Results**. Look at the graphs of interest, such as **DELINQ by BAD**.



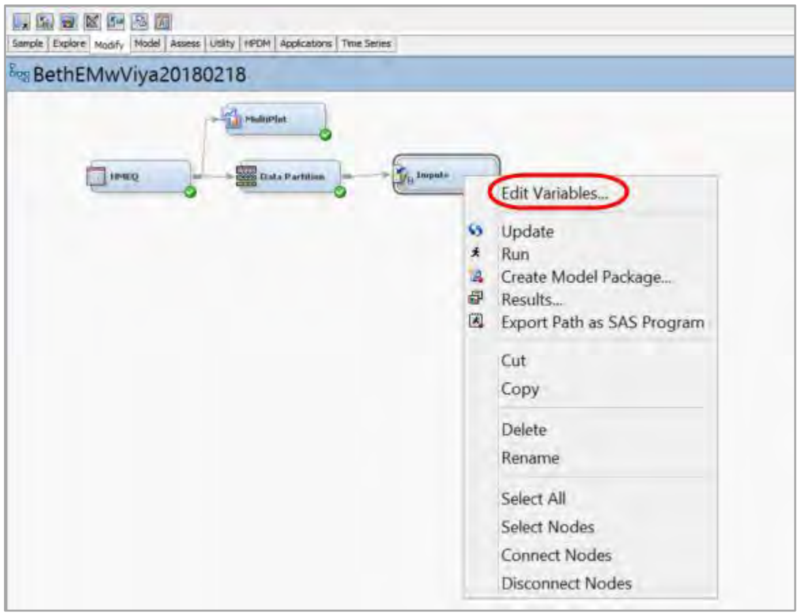
IMPUTING MISSING VALUES

Select the **Modify** tab, drag the **Impute** icon onto the diagram, and change the **Default Number Value** to 0 (under the **Default Constant Value** heading).

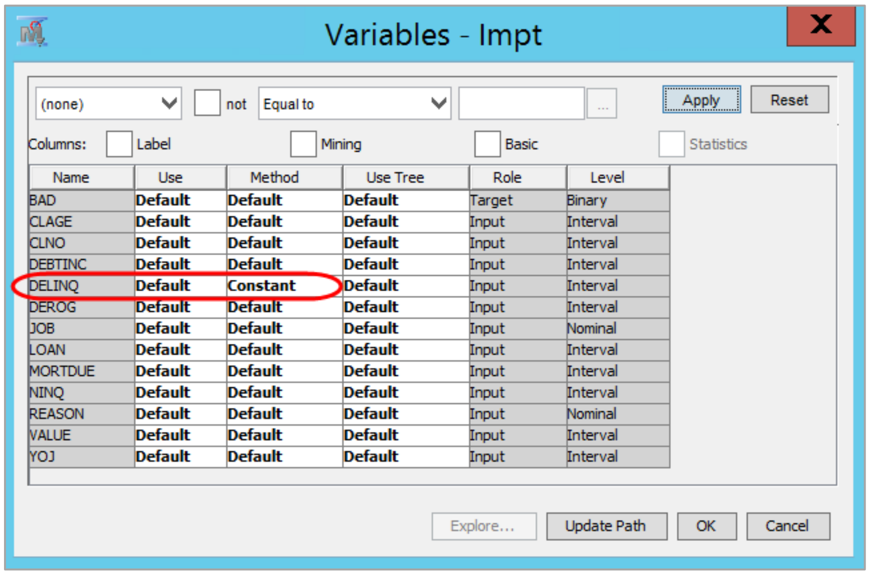


Property	Value
General	
Node ID	Impt
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Nonmissing Variables	No
Missing Cutoff	50.0
Class Variables	
Default Input Method	Count
Default Target Method	None
Normalize Values	Yes
Interval Variables	
Default Input Method	Mean
Default Target Method	None
Default Constant Value	
Default Character Value	
Default Number Value	0
Method Options	
Random Seed	12345
Tuning Parameters	
Tree Imputation	
Score	
Hide Original Variables	Yes
Indicator Variables	

Right-click the **Impute** node and select **Edit Variables**.

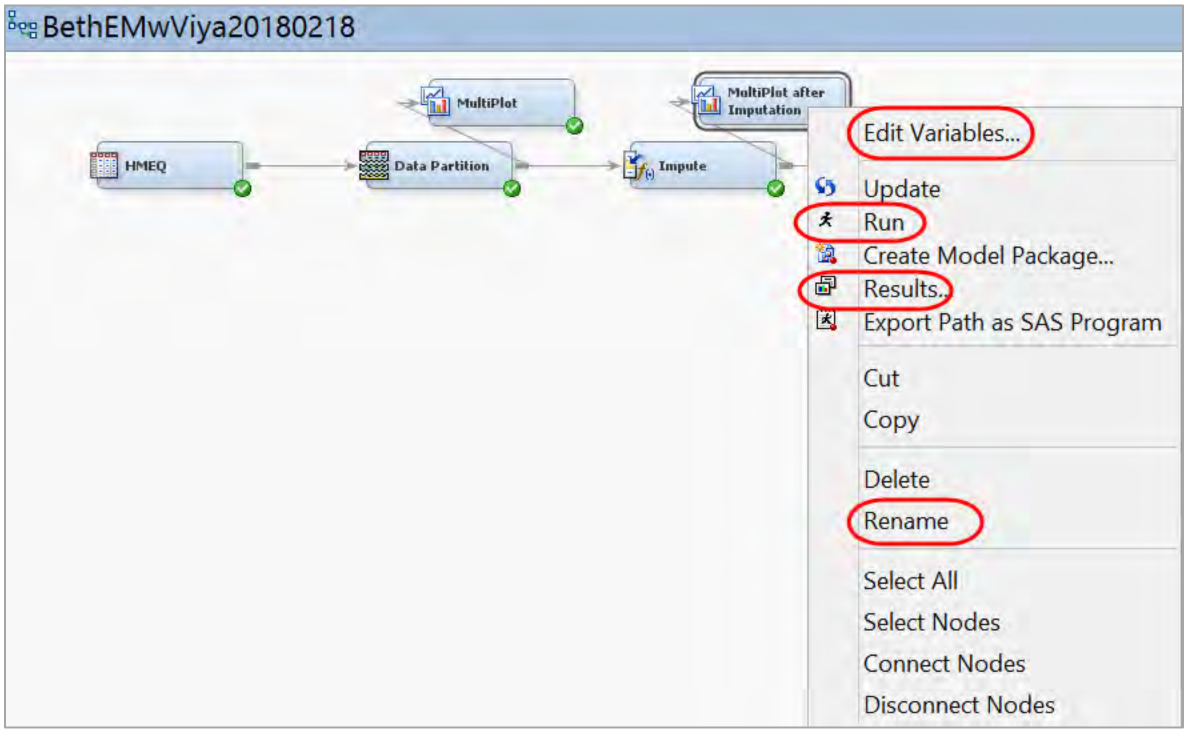


Let's assume that we have some business knowledge about the DELINQ values. We know that in any case where the DELINQ value is denoted as missing, it is actually equal to zero. Change the **Imputation Method** for DELINQ to **Constant**. This sets the missing values for DELINQ to 0, because we have set the **Default Number Value** to 0.

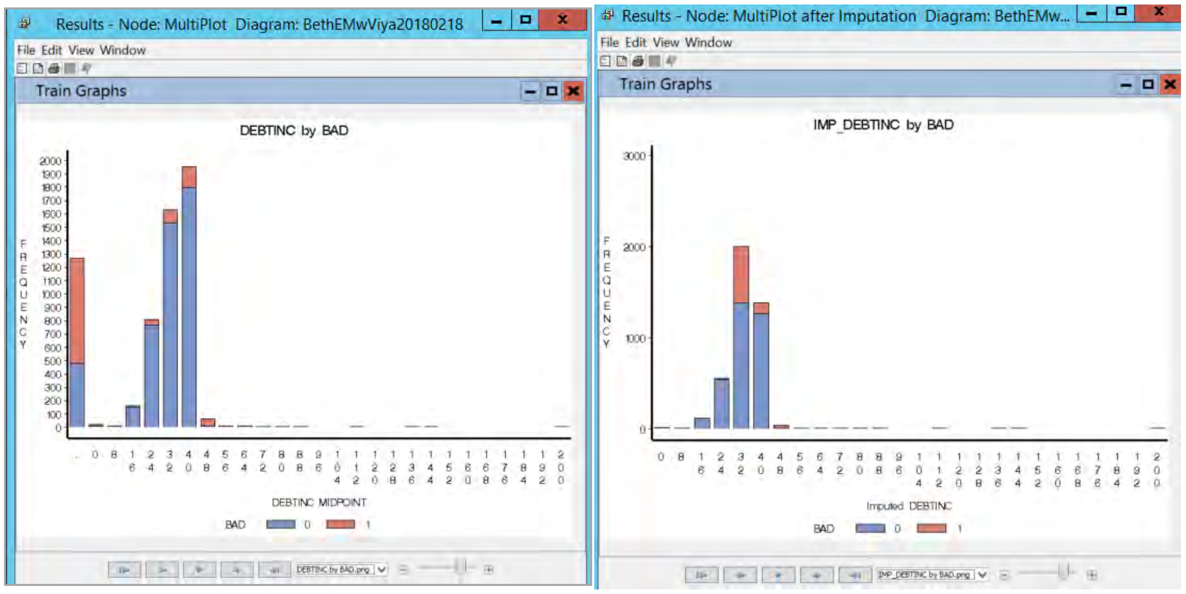


Name	Use	Method	Use Tree	Role	Level
BAD	Default	Default	Default	Target	Binary
CLAGE	Default	Default	Default	Input	Interval
CLNO	Default	Default	Default	Input	Interval
DEBTINC	Default	Default	Default	Input	Interval
DELINQ	Default	Constant	Default	Input	Interval
DEROG	Default	Default	Default	Input	Interval
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

Let's add another **MultiPlot** node (renamed as *Multiplot after Imputation*) from the **Explore** tab to the end of our flow. Run the new Multiplot node path, and look at the **Results**.

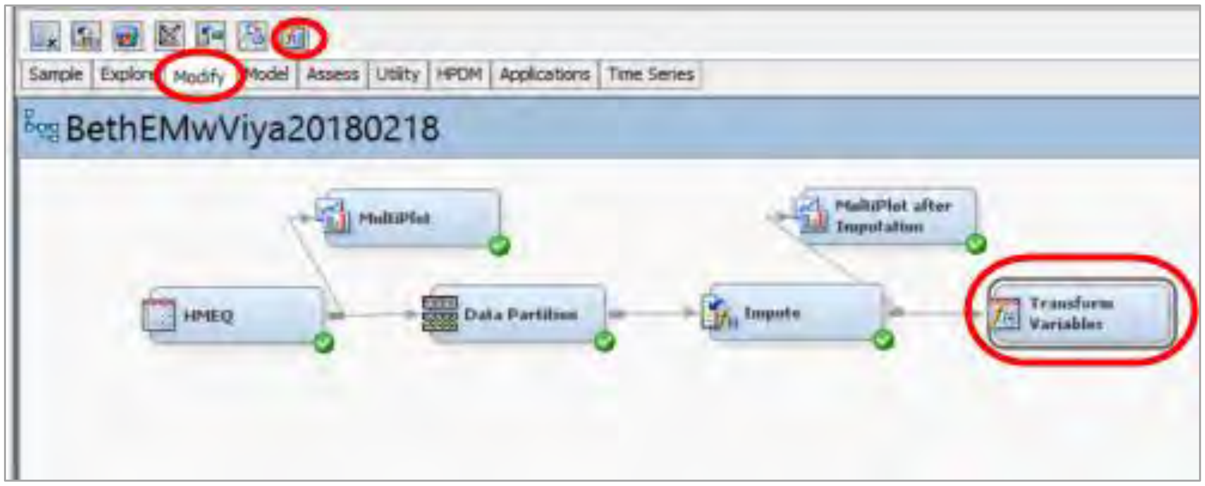


Compare the **Multiplot** histogram for DEBTINC (the first Multiplot node results) to IMP_DEBTINC by BAD to see how the distribution looks after we corrected some missing values.

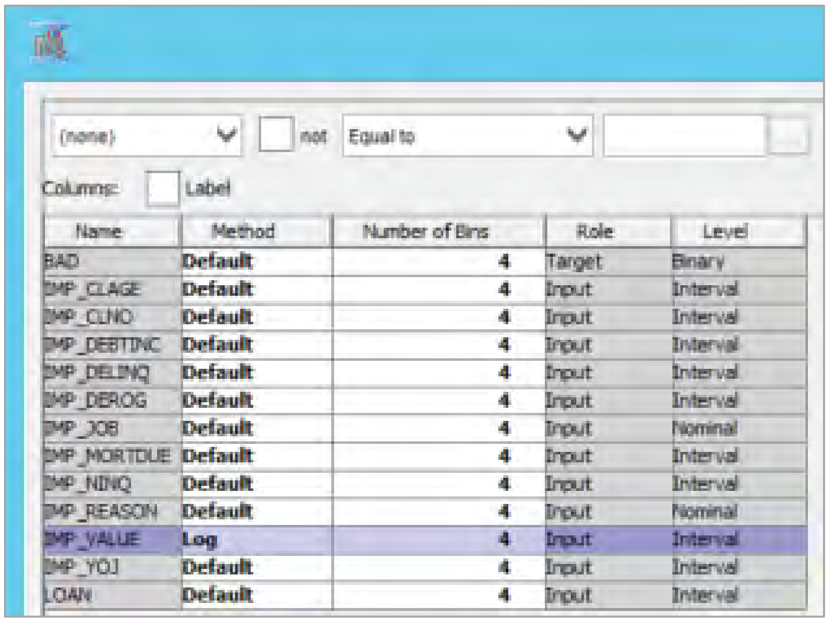


TRANSFORM DATA

Next we will transform input values. Select the **Modify** tab and drag the **Transform Variables** icon onto the diagram. Connect the **Transform Variables** node to the **Impute** node.

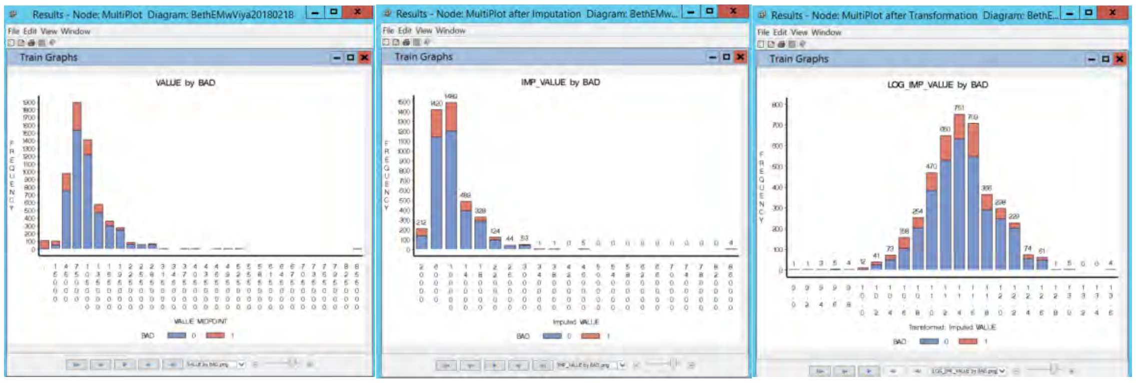


Right-click the **Transform Variables** node and select **Edit Variables**. Change the transformation method for **IMP_VALUE** from **Default** to **Log** to perform a natural log transformation of the imputed **VALUE** variable. Click **OK**.



Name	Method	Number of Bins	Role	Level
BAD	Default	4	Target	Binary
IMP_CLAGE	Default	4	Input	Interval
IMP_CLNO	Default	4	Input	Interval
IMP_DEBTINC	Default	4	Input	Interval
IMP_DELINQ	Default	4	Input	Interval
IMP_DEROG	Default	4	Input	Interval
IMP_JOB	Default	4	Input	Nominal
IMP_MORTDUE	Default	4	Input	Interval
IMP_NINQ	Default	4	Input	Interval
IMP_REASON	Default	4	Input	Nominal
IMP_VALUE	Log	4	Input	Interval
IMP_YOJ	Default	4	Input	Interval
LOAN	Default	4	Input	Interval

From the **Explore** tab, let's add another **Multiplot** node and rename it something like *Multiplot after Transformation*. We connect it to the **Transform Variables** node, run it, and examine the results. We can right-click **Results** on each **Multiplot** node to compare the results at different stages of our path. Below we examine histograms of the **VALUE** (home value) variable before imputation, after imputation, and after a natural log transformation of the imputed values.


Results - Node: MultiPlot Diagram: BethEMwViya20180218
Train Graphs
VALUE by BAD
VALUE MIDPOINT
BAD 0 1
Results - Node: MultiPlot after Imputation Diagram: BethEMw...
Train Graphs
IMP_VALUE by BAD
Imputed VALUE
BAD 0 1
Results - Node: MultiPlot after Transformation Diagram: BethE...
Train Graphs
LOG_IMP_VALUE by BAD
Transformed: Imputed VALUE
BAD 0 1
FREQUENCY

UNDERSTANDING THE SAS VIYA CODE NODE

Let's take a little side trip from our demonstration to get an understanding of the **SAS Viya Code** node. The **SAS Viya Code** node is a SAS Enterprise Miner node that includes a template that gets you started with the code needed to run procedures in the CAS engine from SAS Enterprise Miner. The node includes macros for performing these tasks.

- Start the connection and a CAS session
- Load data into CAS
- Assess models
- Produce graphs and output similar to what we are used to seeing from SAS Enterprise Miner

The **SAS Viya Code** node is similar to the **SAS Code** node. Like the **SAS Code** node, The SAS Viya Code node includes a code editor where you write custom SAS code. However, the code editor in the SAS Viya Code node gives you a leg up because it is pre-populated with a template. That template includes the utility macros for simplifying the integration of your SAS code (which can be SAS Viya code) into SAS Enterprise Miner.

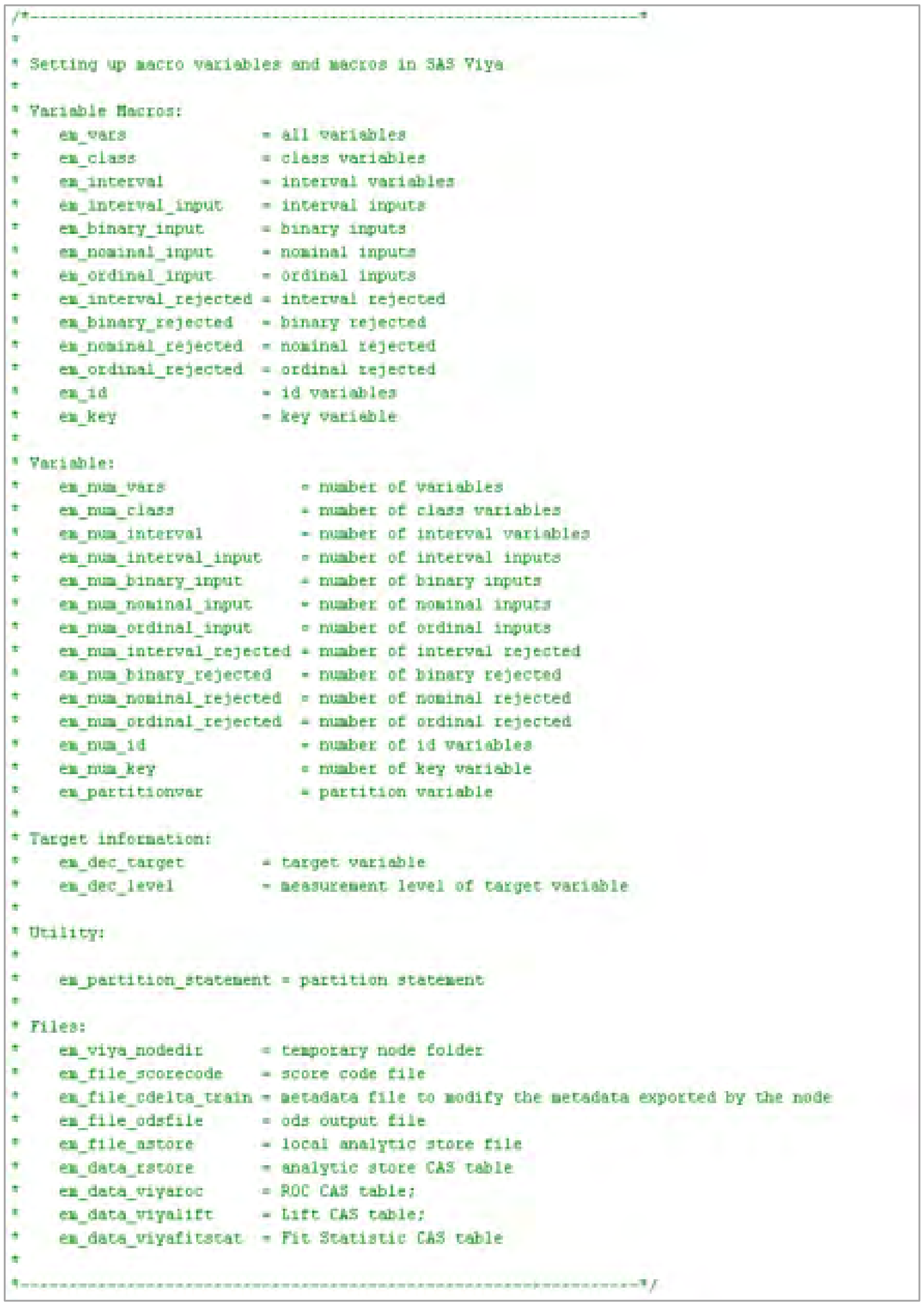
The first part of the SAS Viya Code node includes these items.

- Arguments for the **casHost** and **casPort** macro variables. Note that these variables do not need to be included if the CAS host and port have already been specified through the **em_casHost** and **em_casPort macro** variables in the **Project Start Code**.
- Arguments for the **caslib** and **casuser** macro variables to define the caslib where the data are loaded. For example, these variables could use these values.
  - CASUSER, your default personal caslib.
  - Your user ID (we are using SASTEST1)
- **em_register** macro calls for any tables or files for which you want to register a unique file key; that is, tables or files you want displayed or used for plots in the node results.

These three macros are also included in the template.

- **em_viya_signon**, used to create your CAS session and assign the required libraries.
- **em_viya_dataprep**, used to prepare the data (all partitions, if data has been partitioned) and upload it to the CAS in-memory execution engine.
- **em_viya_setmacrovars**, used to create the macros and macro variables that you can use to represent various attributes about your data in your SAS code. This includes creating the macro variables for tables or files registered with the **em_register** macro. A macro variable called **em_user_key** is created and can then be used to reference the table or file in your code, and this table or file is saved to the workspace directory.

This is a list of the other macros and macro variables that are included in the SAS Viya code template.



```
/*---------------------------------------------------------------*
*
* Setting up macro variables and macros in SAS Viya
*
* Variable Macros:
*     em_vars              = all variables
*     em_class             = class variables
*     em_interval          = interval variables
*     em_interval_input    = interval inputs
*     em_binary_input      = binary inputs
*     em_nominal_input     = nominal inputs
*     em_ordinal_input     = ordinal inputs
*     em_interval_rejected = interval rejected
*     em_binary_rejected   = binary rejected
*     em_nominal_rejected  = nominal rejected
*     em_ordinal_rejected  = ordinal rejected
*     em_id                = id variables
*     em_key               = key variable
*
* Variable:
*     em_num_vars              = number of variables
*     em_num_class             = number of class variables
*     em_num_interval          = number of interval variables
*     em_num_interval_input    = number of interval inputs
*     em_num_binary_input      = number of binary inputs
*     em_num_nominal_input     = number of nominal inputs
*     em_num_ordinal_input     = number of ordinal inputs
*     em_num_interval_rejected = number of interval rejected
*     em_num_binary_rejected   = number of binary rejected
*     em_num_nominal_rejected  = number of nominal rejected
*     em_num_ordinal_rejected  = number of ordinal rejected
*     em_num_id                = number of id variables
*     em_num_key               = number of key variable
*     em_partitionvar          = partition variable
*
* Target information:
*     em_dec_target        = target variable
*     em_dec_level         = measurement level of target variable
*
* Utility:
*
*     em_partition_statement = partition statement
*
* Files:
*     em_viya_nodedir      = temporary node folder
*     em_file_scorecode    = score code file
*     em_file_cdelta_train = metadata file to modify the metadata exported by the node
*     em_file_odsfile      = ods output file
*     em_file_astore       = local analytic store file
*     em_data_rstore       = analytic store CAS table
*     em_data_viyaroc      = ROC CAS table;
*     em_data_viyalift     = Lift CAS table;
*     em_data_viyafitstat  = Fit Statistic CAS table
*
*----------------------------------------------------------------*/
```

The **em_report** macro is used to specify information for any tables or plots you want to include in the results. It uses the key that you created with the **em_register** macro. The **em_viya_report** macro is included in the template to process all the assessment results and other reports.

The **em_viya_modelselection** macro can be used if you want to run multiple models in one **SAS Viya Code** node. It selects the champion model from that node.

The **em_viya_terminate** macro included in the template terminates your CAS session, which is a best practice.

CONNECT TO SAS VIYA

Remember that SAS Enterprise Miner 14.3 does not need SAS/CONNECT in order to work with SAS Viya 3.3.

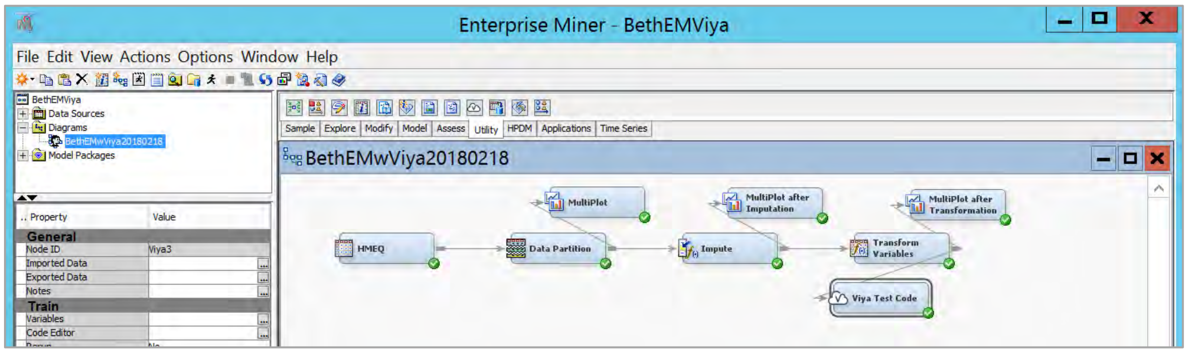
To test that our system is properly configured, let's create a minimal SAS Viya Code node for testing.

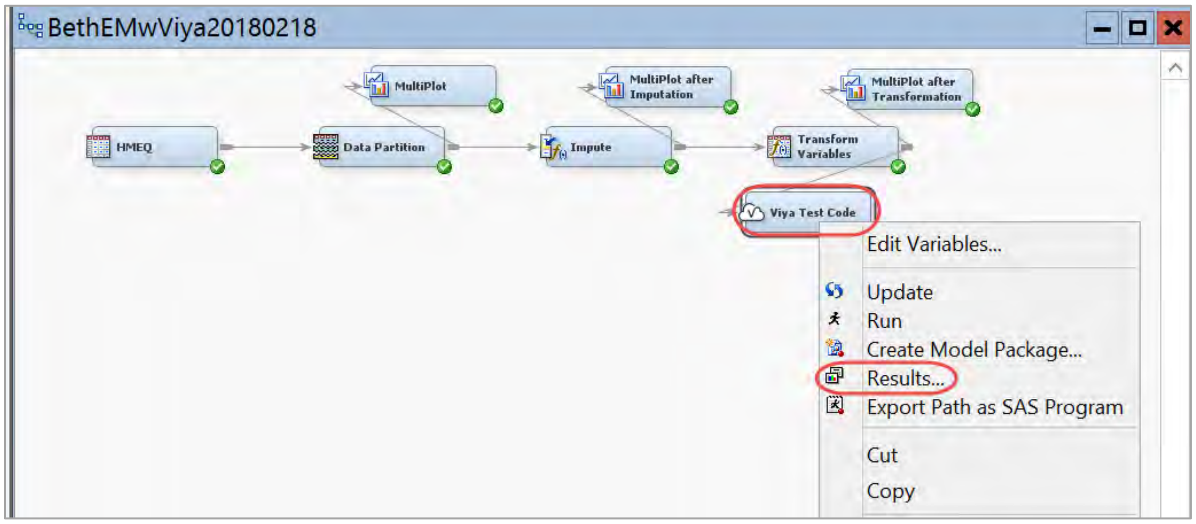
Drag **SAS Viya Code** node into our diagram and connect it to the end of our process flow. As a first step toward making sure that our system is properly configured, use this very basic code.

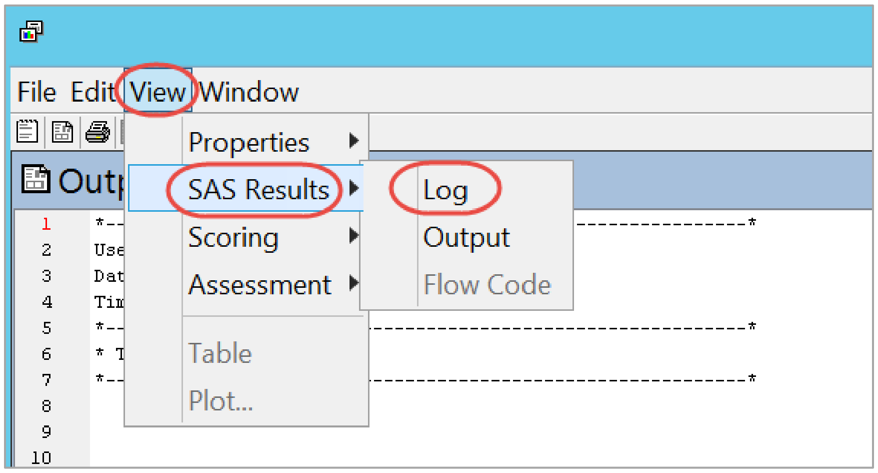
```
%let caslib  = casuser;
%let casUser = sastest1;

options macrogen symbolgen;

%em_viya_signon;
%em_viya_terminate;
```

You must provide the logon credentials (user name and password) in each **SAS Viya Code** node that you create. A best practice is to first encode your password, but we will not do that for this demonstration. Also, because this is a simple demonstration, we authenticate with the same casUser name and viyaUser name (both are sastest1). If you use a different user name and password, you must provide the **user** option with a corresponding authinfo file.

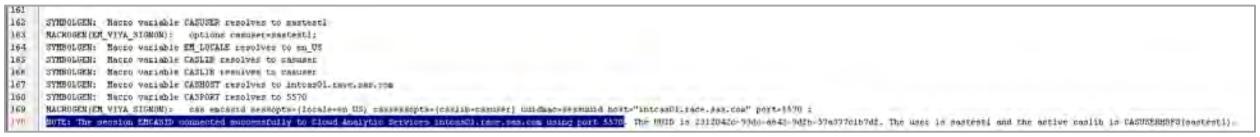
Let's right-click and run our **SAS Viya Test Code** node, and view the SAS log as shown in the screenshots below.







Search in our log for Cloud Analytic Services. If all is well, we can see that the session connected successfully to CAS using port 5570.



```
SYMBOLGEN:  Macro variable CASUSER resolves to sastest1
MACROGEN(EM_VIYA_SIGNON):   options casuser=sastest1;
SYMBOLGEN:  Macro variable EM_LOCALE resolves to en_US
SYMBOLGEN:  Macro variable CASLIB resolves to casuser
SYMBOLGEN:  Macro variable CASLIB resolves to casuser
SYMBOLGEN:  Macro variable CASHOST resolves to intcas01.race.sas.com
SYMBOLGEN:  Macro variable CASPORT resolves to 5570
MACROGEN(EM_VIYA_SIGNON):   cas emcasid sessopts=(locale=en_US) casssessopts=(caslib=casuser) uuidmac=&emuuid host="intcas01.race.sas.com" port=5570 ;
NOTE: The session EMCASID connected successfully to Cloud Analytic Services intcas01.race.sas.com using port 5570. The UUID is 2312042c-93dc-e642-9d2b-57a377c1b7d2. The user is sastest1 and the active caslib is CASUSERHDFS(sastest1).
```

If this node works, disconnect the node. We will not need it again.

If this node did not run correctly, look for clues in the log and read the additional resources provided at the end of this paper about how to properly install and configure SAS Enterprise Miner 14.3 and SAS Viya 3.3.

CREATE A SAS VIYA NEURAL NETWORK NODE

The template of the **SAS Viya Code** node includes a section for you to enter your custom SAS code, which can be a combination of SAS 9 and SAS Viya code. We will first create a **SAS Viya Neural Network** node using the SAS® Visual Data Mining and Machine Learning (VDMML) NNET procedure, but you can use any appropriate SAS® Visual Statistics or VDMML procedures.

If your code creates score code that produces the appropriate posterior probabilities (for a class target) or predicted variables (for an interval target), you can then use the **em_viya_assess** macro to obtain the assessment results in the same way as you do for SAS Enterprise Miner modeling nodes. The score code can either be DATA step code or an analytic store.

Right-click the node and rename it **Viya Neural Network**. Open the **Viya Neural Network** node code editor. As with the test node, we use %let to set the caslib and casUser values.

Next, we locate the **%em_viya_setMacroVars** line in the template code. Prior to calling the **em_viya_setmacrovars** macro, use the provided **EM_REGISTER** macro code to produce macro variables for the modelinfo, iterhist, scoreInfo, and nnetmodel data sets. Note that the new CAS argument indicates that the nnetmodel table created by the OUTMODEL= option of the NNET procedure will be on the CAS server. This will also ensure that these tables are moved from the CAS server to the SAS Enterprise Miner diagram folder.

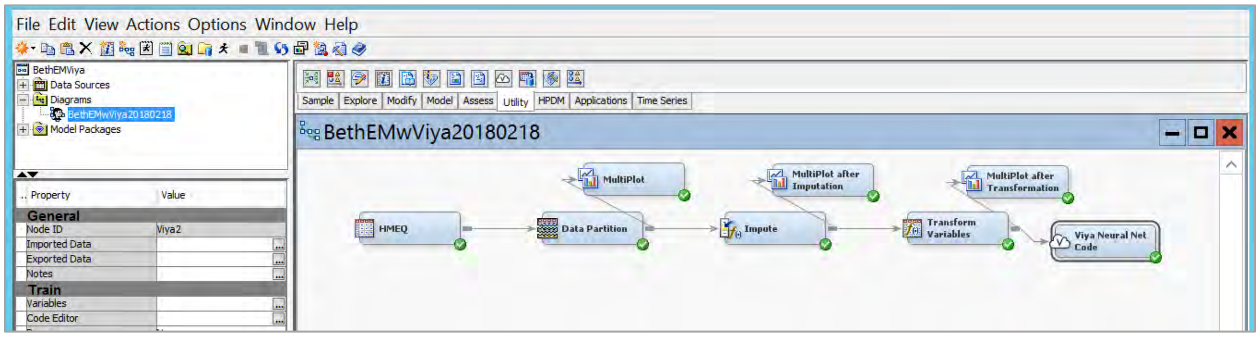
Our final code in the **Viya Neural Network** node should look something like this.

```
/* Viya Neural Network */

/*--------------------------------------------------------------*
* Setup the CAS information
* Note that the casHost and casPort have been specified in the EM project
* startup code using the em_casHost and em_casPort macro variables.
*-----------------------------------------------------------------*/
   %let caslib   = casuser;
   %let casuser  = sastest1;

   options macrogen symbolgen ; *to help with debugging;

/*----------------------------------------------------------------*
* Establish a connection to SASviya machine
* Create a CAS session and assign the CAS library.
*-----------------------------------------------------------------*/

 %em_viya_signon;

/*----------------------------------------------------------------*
* Prepare the training data and upload it to the SAS Viya machine
* The em_viya_dataprep macro  assumes that a connection has been
* established with the SAS Viya server and a CAS library has been
* assigned.
*-----------------------------------------------------------------*/

%em_viya_dataprep;

/*----------------------------------------------------------------*
* Set up macro variables and macros in SAS Viya
*-----------------------------------------------------------------*/

%em_register(key=modelinfo, type=DATA);
%em_register(key=iterhist,  type=DATA);
%em_register(key=scoreInfo, type=DATA);
%em_register(key=nnetmodel, type=data, cas=y);

%em_viya_setMacroVars;

/*----------------------------------------------------------------*
* User Code to Run Neural Network on CAS
*-----------------------------------------------------------------*/

proc nnet data=&em_casdata;
  target &em_dec_target /level=&em_dec_level;
  hidden 2;
  input %em_interval_input / level=interval;
  input %em_nominal_input  / level=nominal;
  train outmodel=&em_user_nnetmodel;
  architecture MLP;
  optimization alg=LBFGS;
       *limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm;
  code file="&em_file_scorecode";
  ods output ModelInfo=&em_user_modelinfo
             OptIterHistory=&em_user_iterhist
             ScoreInfo=&em_user_scoreInfo;
  run;

/*----------------------------------------------------------------*
* Call the em_viya_assess macro to create
* assessment results using the ASSESS procedure.
```

```
*----------------------------------------------------------------*/

   %em_viya_assess(name=Neural);

/*---------------------------------------------------------------*
* The em_viya_report macro processes assessment tables, score code, as
* well as table and files specified via the em_register macro and downloads
* them from the SAS Viya machine to the EM diagram and node folders.
*----------------------------------------------------------------*/

   %em_report(key=nnetmodel, viewtype=DATA, description=NNET Model,
Block=Custom SAS Viya Reports);
   %em_report(key=modelinfo, viewtype=DATA, description=Model Information,
Block=Custom SAS Viya Reports);
   %em_report(key=iterhist,  viewtype=DATA, description=Iteration History,
Block=Custom SAS Viya Reports);
   %em_report(key=scoreinfo, viewtype=DATA, description=Score Information,
Block=Custom SAS Viya Reports);
   %em_viya_report;

/*---------------------------------------------------------------*
* The em_viya_terminate macro terminates the CAS session as well as ending
* the connection to the SAS Viya machine
*----------------------------------------------------------------*/

   %em_viya_terminate;
```

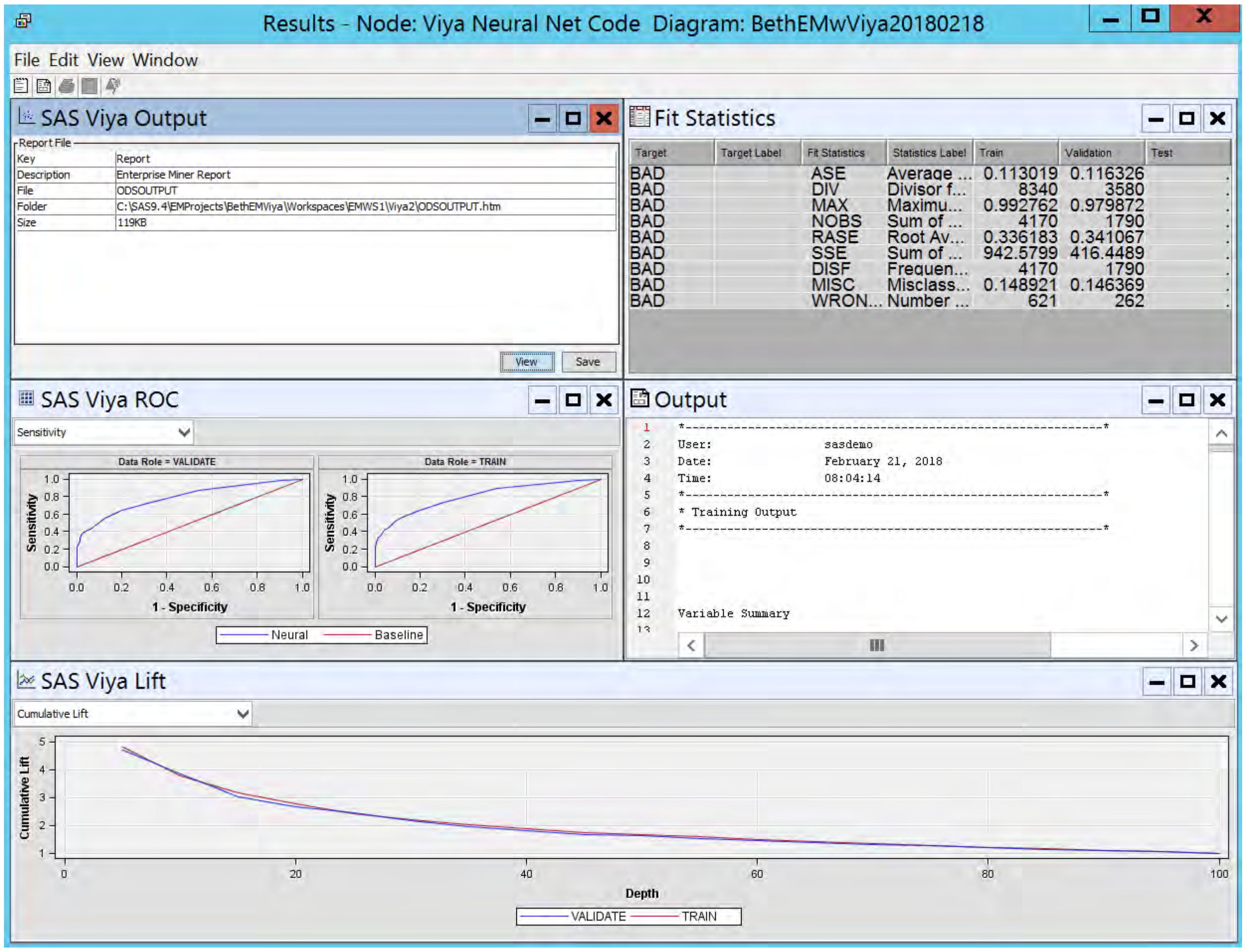
Notice that RSUBMIT and ENDRSUBMIT statements are not needed, because we no longer need to use SAS/CONNECT to access SAS Viya from SAS 9.4M5 as we did in SAS Enterprise Miner 14.2.

Let's connect our **Viya Neural Network** node to our **Transform Variables** node. Our diagram should look like the following.



Property	Value
General	
Node ID	Viya2
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Code Editor	...

Our results should look similar to the following.



Report File

Key	Report
Description	Enterprise Miner Report
File	ODSOUTPUT
Folder	C:\SAS9.4\EMProjects\BethEMViya\Workspaces\EMWS1\Viya2\ODSOUTPUT.htm
Size	119KB

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
BAD		ASE	Average ...	0.113019	0.116326	.
BAD		DIV	Divisor f...	8340	3580	.
BAD		MAX	Maximu...	0.992762	0.979872	.
BAD		NOBS	Sum of ...	4170	1790	.
BAD		RASE	Root Av...	0.336183	0.341067	.
BAD		SSE	Sum of ...	942.5799	416.4489	.
BAD		DISF	Frequen...	4170	1790	.
BAD		MISC	Misclass...	0.148921	0.146369	.
BAD		WRON...	Number ...	621	262	.

Output

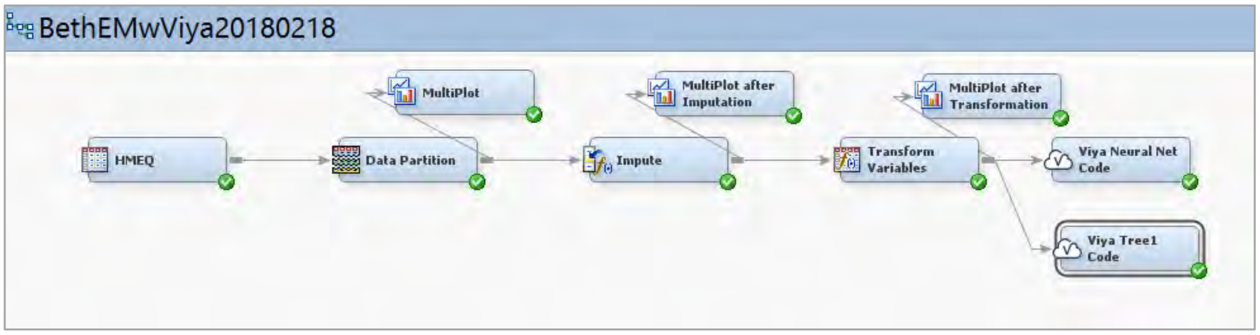
```
*------------------------------------------------------------*
User:                    sasdemo
Date:                    February 21, 2018
Time:                    08:04:14
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary
```

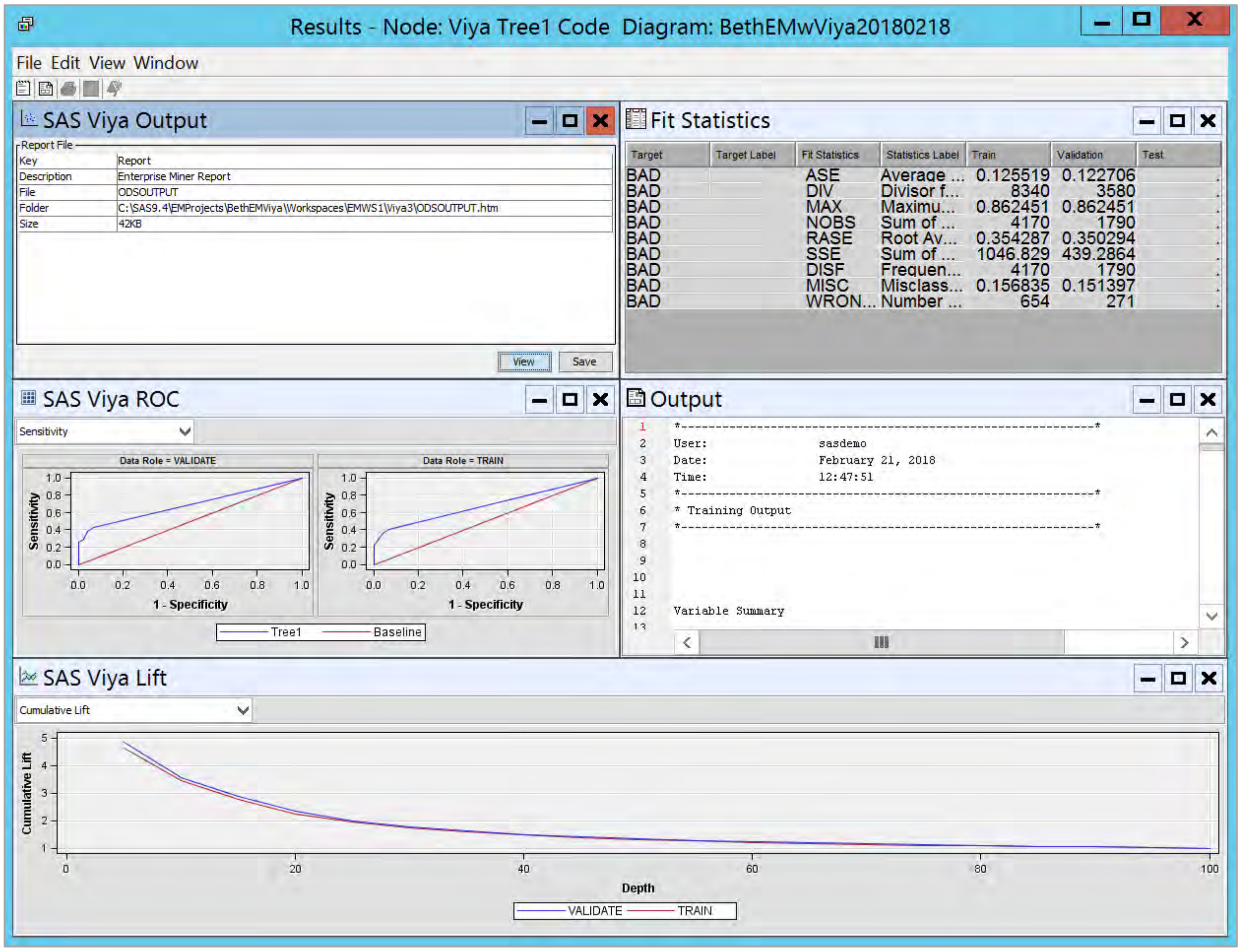
CREATE TWO SAS VIYA DECISION TREE NODES

Drag in the **SAS Viya Code** node into the diagram and rename it **Viya Tree1 Code**. Replace the code in our **Viya Decision Tree** node with the following code (for brevity I have removed the annotation).

```
    /* Viya Tree 1 */

    %let caslib  =casuser;
    %let casuser =sastest1;

    %em_viya_signon;

    %em_viya_dataprep;

    %em_register(key=varImportance1, type=DATA);
    %em_viya_setMacroVars;

proc treesplit data=&em_casdata;
  target &em_dec_target     / level=&em_dec_level;
  input %em_interval_input / level=interval;
  input %em_nominal_input  / level=nominal;
  prune reducederror;
  &em_partition_statement;
  code file="&em_file_scorecode";
  ods output variableImportance=&em_user_varimportance1;
run;

%em_viya_assess (name=Tree1);

%em_report(key=varImportance1, viewtype=DATA, description=Variable
Importance Tree1, Block=Custom SAS Viya Reports);
%em_viya_report;

%em_viya_terminate;
```

Connect our **Viya Tree1 Code** node and run it to get results similar to those below.





Report File

Key	Report
Description	Enterprise Miner Report
File	ODSOUTPUT
Folder	C:\SAS9.4\EMProjects\BethEMViya\Workspaces\EMWS1\Viya3\ODSOUTPUT.htm
Size	42KB

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
BAD		ASE	Average ...	0.125519	0.122706	.
BAD		DIV	Divisor f...	8340	3580	.
BAD		MAX	Maximu...	0.862451	0.862451	.
BAD		NOBS	Sum of ...	4170	1790	.
BAD		RASE	Root Av...	0.354287	0.350294	.
BAD		SSE	Sum of ...	1046.829	439.2864	.
BAD		DISF	Frequen...	4170	1790	.
BAD		MISC	Misclass...	0.156835	0.151397	.
BAD		WRON...	Number ...	654	271	.

Output

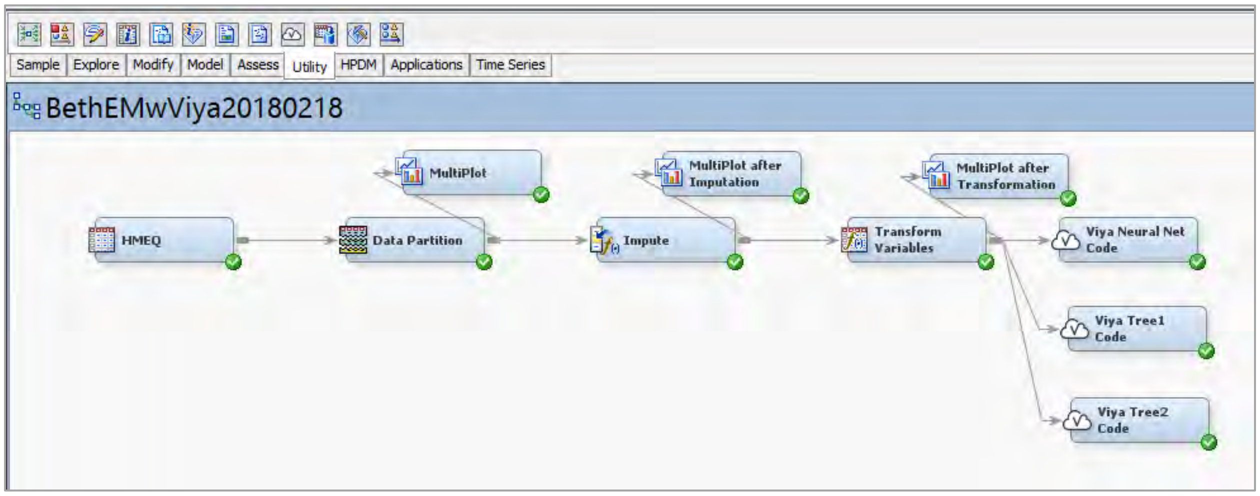
```
*------------------------------------------------------------*
User:                sasdemo
Date:                February 21, 2018
Time:                12:47:51
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary
```

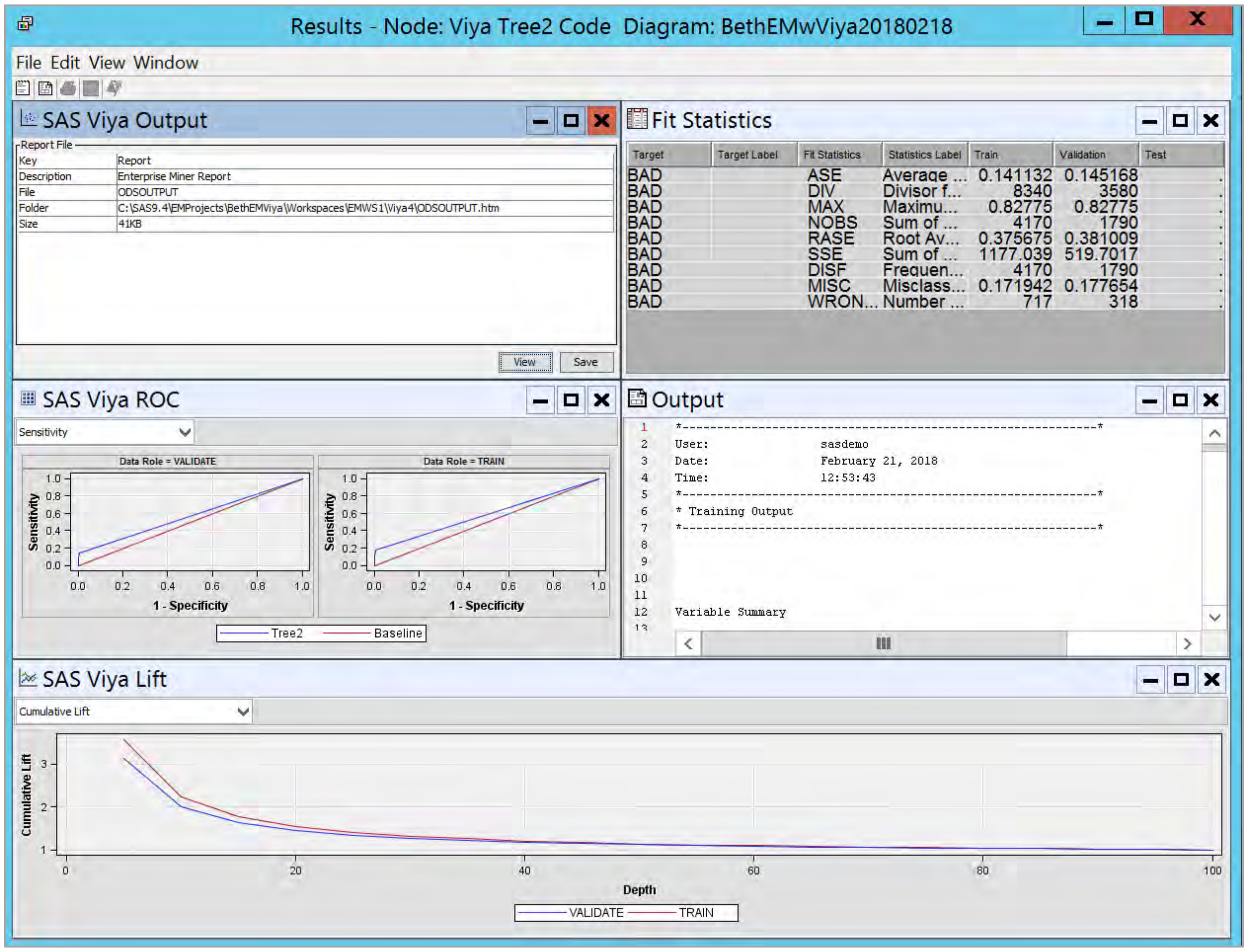
Now let's create a second tree node and label it **Viya Tree 2 Code**. Replace the code in our **Viya Decision Tree** node with the following code. This time we override the defaults of PROC TREESPLIT, set the maximum depth of the decision tree at 5, and set the number of bins for binning interval inputs at 8.

```
/* Viya Tree 2 */

%let caslib  =casuser;
%let casuser =sastest1;

%em_viya_signon;
```

```
   %em_viya_dataprep;

   %em_register(key=varImportance2, type=DATA);
   %em_viya_setMacroVars;

proc treesplit data=&em_casdata maxdepth=5 numbin=8;
  target &em_dec_target    / level=&em_dec_level;
  input %em_interval_input / level=interval;
  input %em_nominal_input  / level=nominal;
  prune reducederror;
  &em_partition_statement;
  code file="&em_file_scorecode";
  ods output variableImportance=&em_user_varimportance2;
run;

%em_viya_assess (name=Tree2);

%em_report(key=varImportance2, viewtype=DATA, description=Variable
Importance Tree2, Block=Custom SAS Viya Reports);
%em_viya_report;

%em_viya_terminate;
```

Connect and run the **Viya Tree2 Code** node to obtain the results as shown below.





Report File

Key	Report
Description	Enterprise Miner Report
File	ODSOUTPUT
Folder	C:\SAS9.4\EMProjects\BethEMViya\Workspaces\EMWS1\Viya4\ODSOUTPUT.htm
Size	41KB

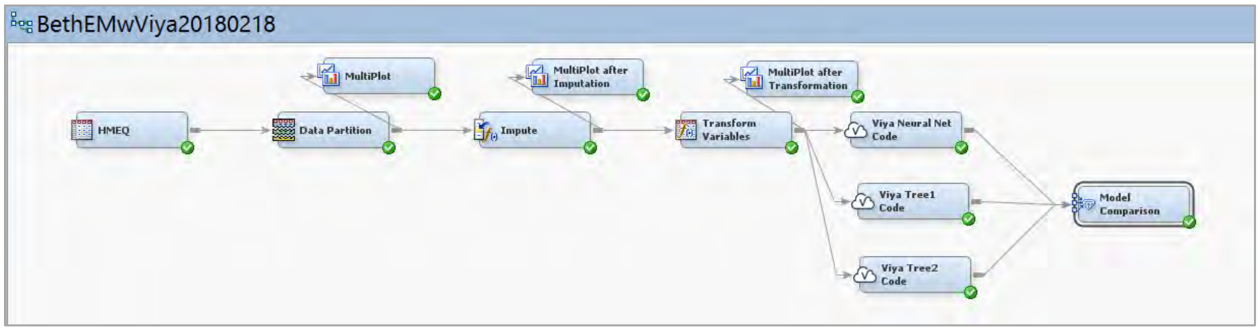
Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
BAD		ASE	Average ...	0.141132	0.145168	.
BAD		DIV	Divisor f...	8340	3580	.
BAD		MAX	Maximu...	0.82775	0.82775	.
BAD		NOBS	Sum of ...	4170	1790	.
BAD		RASE	Root Av...	0.375675	0.381009	.
BAD		SSE	Sum of ...	1177.039	519.7017	.
BAD		DISF	Frequen...	4170	1790	.
BAD		MISC	Misclass...	0.171942	0.177654	.
BAD		WRON...	Number ...	717	318	.

```
*------------------------------------------------------------*
User:                  sasdemo
Date:                  February 21, 2018
Time:                  12:53:43
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary
```

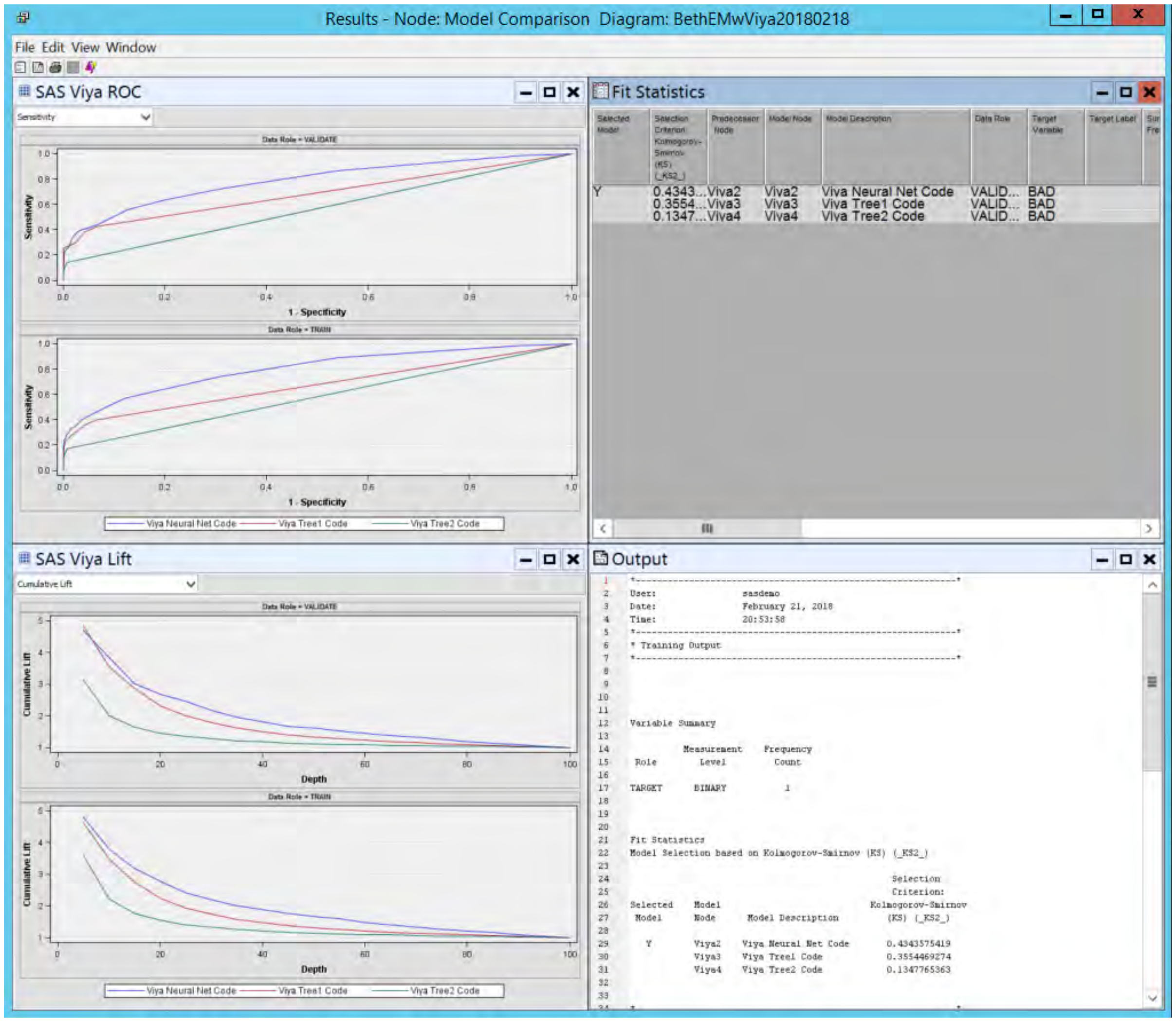
COMPARE MODELS

Finally, from the **Assess** tab we can drag a **Model Comparison** node into the diagram. This node runs on the workspace server attached to the SAS Environment Manager 14.3 environment, rather than in CAS, although it uses the results generated from the algorithms that run in CAS. Connect the **Viya Neural Network** node and the **Viya Tree1** and **Viya Tree2** nodes to the **Model Comparison** node to compare the selected decision tree to the neural network. Note that we are now doing this comparison on the SAS Environment Manager 14.3 workspace server, not in CAS.



RESULTS

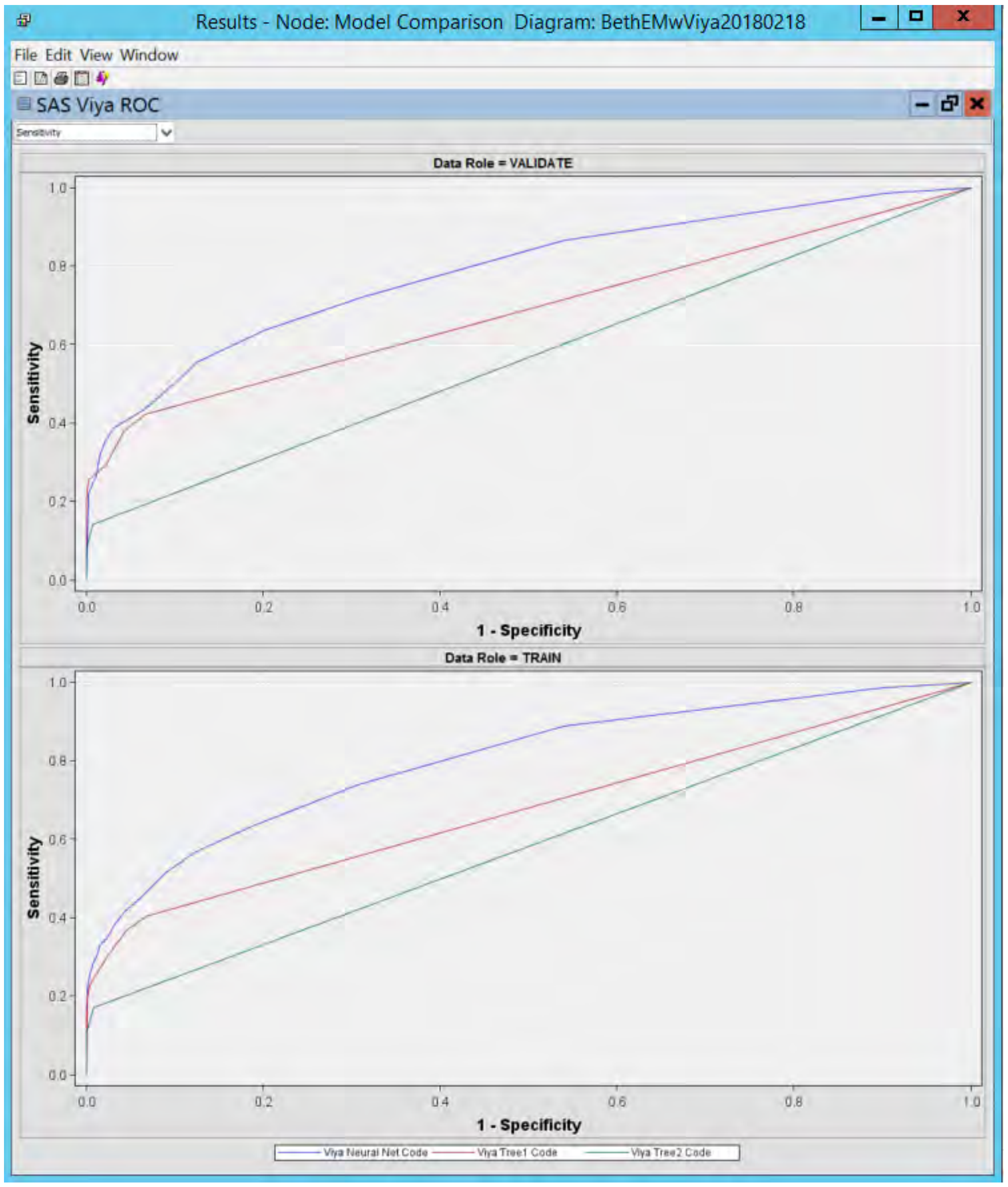
Here we see that the **Neural Network** is selected as the best model, based on the Kolmogorov-Smirnov value of 0.43, compared to 0.36 and 0.13 for **Decision Tree 1** and **Decision Tree 2**, respectively.



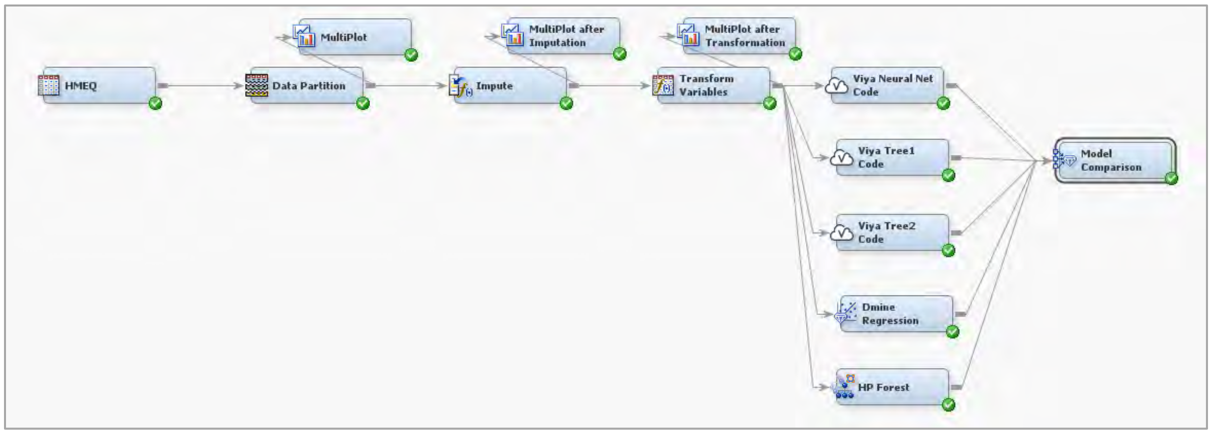
Selected Model	Selection Criterion: Kolmogorov-Smirnov (KS) (_KS2_)	Predecessor Node	Model Node	Model Description	Data Role	Target Variable	Target Label
Y	0.4343...	Viva2	Viva2	Viva Neural Net Code	VALID...	BAD	
	0.3554...	Viva3	Viva3	Viva Tree1 Code	VALID...	BAD	
	0.1347...	Viva4	Viva4	Viva Tree2 Code	VALID...	BAD	

```
*------------------------------------------------------------*
User:                 sasdemo
Date:                 February 21, 2018
Time:                 20:53:58
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*




Variable Summary

         Measurement    Frequency
 Role       Level         Count

TARGET     BINARY           1




Fit Statistics
Model Selection based on Kolmogorov-Smirnov (KS) (_KS2_)

                                                Selection
                                                Criterion:
Selected   Model                            Kolmogorov-Smirnov
 Model     Node      Model Description         (KS) (_KS2_)

   Y       Viya2    Viya Neural Net Code       0.4343575419
           Viya3    Viya Tree1 Code            0.3554469274
           Viya4    Viya Tree2 Code            0.1347765363
```

We can also see this visually in the **SAS Viya ROC** curve in the upper left of the screen shot below. We see that the **Neural Network Validation ROC** curve (the blue curve) is the farthest up and to the left. This indicates that we have the highest sensitivity (true positive rate) and lowest 1-specificity (false positive rate) of the three models used.



We could also easily compare our **SAS Viya Code** nodes to traditional SAS Enterprise Miner nodes or HPDM nodes by bringing those nodes in and connecting them. However, these nodes run on the workspace server and will not take advantage of the CAS engine. With our small HMEQ sample data set, this might not bother us, but if we are using a large data set, we might find our wait time for results to be quite improved when we use the CAS engine to run complex models.



CONCLUSION

By running computer intensive machine learning algorithms such as neural networks in CAS, where they can run multithreaded, we can get results very quickly and bring those results back to SAS Enterprise Miner. Now you see how easy it is to take advantage of the power of the CAS engine in SAS Viya from the familiar interface of our old, trusted friend—SAS Enterprise Miner.

REFERENCES

FOR DATA SCIENTISTS AND ENTERPRISE MINER USERS

- Czika, Wendy. "Tip: Using the SAS Viya Code node in SAS Enterprise Miner 14.3." Available: https://communities.sas.com/t5/SAS-Communities-Library/Tip-Using-the-SAS-Viya-Code-node-in-SAS-Enterprise-Miner-14-3/ta-p/401829. Accessed on March 1, 2018
- SAS Institute Inc. 2017 SAS Software/em-bridge2viya/EM-14.3. Accessed March 1, 2018. Available: https://github.com/sassoftware/em-bridge2viya/tree/master/EM-14.3
- Ebersole, Beth. "Tree-related Models: Supervised Learning in SAS Visual Data Mining and Machine Learning." Available: https://communities.sas.com/t5/tkb/articleprintpage/tkb-id/library/article-id/1865. Accessed on March 1, 2018
- Ebersole, Beth. "Neural Network Models: Supervised Learning in SAS Visual Data Mining and Machine Learning." Available: https://communities.sas.com/t5/SAS-Communities-Library/Neural-Network-Models-Supervised-Learning-in-SAS-Visual-Data/ta-p/343330. Accessed on March 1, 2018

FOR INSTALLERS AND ADMINISTRATORS ON CONFIGURING ENTERPRISE MINER TO WORK WITH VIYA

- Thomas, Mark. "Configuring Enterprise Miner to Chat with CAS." Available: http://sww.sas.com/blogs/wp/gate/19765/configuring-enterprise-miner-to-chat-with-cas/martho/2018/01/29. Accessed March 1, 2018

- Collum, Rob. "Connecting to CAS as a Different User from SAS Enterprise Miner." Available: http://sww.sas.com/blogs/wp/gate/15732/connecting-to-cas-as-a-different-user-from-sas-enterprise-miner/rocoll/2017/06/02. Accessed on March 1, 2018.
- Thomas, Mark. "Crossing the Viya Bridge." Available: http://sww.sas.com/blogs/wp/gate/10036/10036/martho/2016/06/01. Accessed on March 1, 2018.
- Rogers, Stuart. "SAS Enterprise Miner 14.1: An IWA Gotcha…" Available: http://sww.sas.com/blogs/wp/gate/6447/sas-enterprise-miner-14-1-an-iwa-gotcha/suksar/2015/09/04. Accessed on March 1, 2018.
- Rogers, Stuart. "Configuring SAS Enterprise Miner 13.1 with Integrated Windows Authentication." Available: http://sww.sas.com/toolpool/pool/Configuring_SAS_Enterprise_Miner_13.1_with_Integrated_Windows_Authentication/index.html. Accessed on March 1, 2018.
- SAS Institute Inc. 2017. "Create an Authinfo File." In *Client Authentication Using an Authinfo File*. Cary, NC SAS Institute, Inc. Available: http://pubshelpcenter.unx.sas.com:8080/test/?docsetId=authinfo&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en#n1stv9zynsyf6rn1wbr3ejga6ozf (Accessed March 1, 2018)

ACKNOWLEDGMENTS

A hearty thank you to Mark Thomas, Rob Collum, and Stuart Rogers for sorting out configuration, authentication, and security issues associated with running CAS actions from Enterprise Miner.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Beth Ebersole
1530 Wilson Boulevard, Suite 800
Arlington, VA 22209 USA
SAS Institute, Inc.
Beth.Ebersole@sas.com
http://www.sas.com

Wendy Czika
100 SAS Campus Drive
Cary, NC 27513 USA
SAS Institute, Inc.
Wendy.Czika@sas.com
http://www.sas.com