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

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Main New Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 2018</td>
<td>EM 14.3</td>
<td>SAS/CONNECT not needed to connect SAS Viya Code to Viya</td>
</tr>
<tr>
<td>NOV 2016</td>
<td>EM 14.2</td>
<td>SAS Viya Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analytic Item Store</td>
</tr>
<tr>
<td>JUL 2015</td>
<td>EM 14.1</td>
<td>HP Bayesian Network Node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP Variable Selection: New tree-based selection method</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP Clustering: Automatic selection of number of clusters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Score code support PMML 4.2</td>
</tr>
<tr>
<td>AUG 2014</td>
<td>EM 13.2</td>
<td>HP Regression creates VIF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support for SAP HANA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enhanced HP Cluster, HP Forest &amp; HP TS Dimensionality</td>
</tr>
<tr>
<td>DEC 2013</td>
<td>EM 13.1</td>
<td>3 New Time Series Nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>More HP Nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Open Source Integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Register Models Node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Save Data Node</td>
</tr>
<tr>
<td>JUL 2013</td>
<td>EM 12.3</td>
<td>SAS 9.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HP Data Mining Tab</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Link Analysis</td>
</tr>
<tr>
<td>AUG 2012</td>
<td>EM 12.1</td>
<td>Time Series Data Mining Nodes Production</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interactive Grouping Node Redesigned</td>
</tr>
<tr>
<td>DEC 2011</td>
<td>EM 7.1</td>
<td>High Performance Data Mining: procedures and nodes PMML Scoring</td>
</tr>
<tr>
<td>M1</td>
<td></td>
<td>Incremental Response Modeling</td>
</tr>
<tr>
<td>JUL 2011</td>
<td>EM 7.1</td>
<td>SAS 9.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Web Services Architecture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In-database (GA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fast GLM for Insurance Ratemaking &amp; Survival Nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVM &amp; Time Series Nodes (Experimental)</td>
</tr>
<tr>
<td>AUG 2010</td>
<td>EM 6.2</td>
<td>Rapid Predictive Modeler - RPM (production)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In-database (limited availability)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interactive Tree &amp; Credit Scoring Updates</td>
</tr>
<tr>
<td>DEC 2009</td>
<td>EM 6.1</td>
<td>Rapid Predictive Modeler - RPM (early adopter)</td>
</tr>
<tr>
<td>M2</td>
<td></td>
<td>Interactive Tree &amp; Credit Scoring</td>
</tr>
<tr>
<td>MAY 2009</td>
<td>EM 6.1</td>
<td>SAS 9.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>File import node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data summarization Variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Native interactive decision trees/segmentation strategy builder Algorithms: LARS high dimensional feature selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optimized score code generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved Reporter node output</td>
</tr>
<tr>
<td>Date</td>
<td>Release</td>
<td>Main New Features</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEC 2007</td>
<td>EM 5.3</td>
<td><strong>SAS 9.1.3</strong>&lt;br&gt;Updated PFD&lt;br&gt;Group Processing&lt;br&gt;Algorithms: Gradient Boosting, Variable Clustering, &amp; Hierarchical Associations&lt;br&gt;Native interactive binning for credit scoring (easy admin and improved usability)&lt;br&gt;Graphical data exploration and persistence&lt;br&gt;EM 4.3 Project Conversion Utility&lt;br&gt;Interactive SAS Code node</td>
</tr>
<tr>
<td>NOV 2005</td>
<td>EM 5.2</td>
<td>Decision, Replacement &amp; SOM/Kohonen nodes&lt;br&gt;Transform Formula Builder&lt;br&gt;Interactive selections for outliers, market baskets, principal components&lt;br&gt;SAS Code node training and scoring editors with drag and drop macros&lt;br&gt;Exported data node property&lt;br&gt;New path plots&lt;br&gt;EM SMC plug in&lt;br&gt;Grid Processing&lt;br&gt;EM Desktop release</td>
</tr>
<tr>
<td>2005</td>
<td>EM 5.1</td>
<td><strong>SAS 9.1</strong>&lt;br&gt;Rich Java Client/SAS Server&lt;br&gt;Parallel and Batch processing&lt;br&gt;XML diagram exchange&lt;br&gt;Model packages&lt;br&gt;Graph explorer&lt;br&gt;Credit Scoring nodes</td>
</tr>
<tr>
<td>2002</td>
<td>TM 3.1</td>
<td>Text Miner Add-on&lt;br&gt;(SAS is first DM vendor to integrate text with structured data)</td>
</tr>
<tr>
<td>2001</td>
<td>EM 4.1</td>
<td>Link Analysis&lt;br&gt;Memory-Based Reasoning (MBR)&lt;br&gt;Time Series</td>
</tr>
<tr>
<td>2000</td>
<td>EM 4.0</td>
<td><strong>SAS 8.2</strong>&lt;br&gt;C &amp; Java Scoring Code&lt;br&gt;Tree Desktop Viewer&lt;br&gt;DMTCOL application</td>
</tr>
<tr>
<td>1998</td>
<td>EM 2.01</td>
<td><strong>EM Inaugural Release on SAS 6.12</strong>&lt;br&gt;Client Server Data Mining Workbench&lt;br&gt;Windows and Unix Servers&lt;br&gt;Process flow diagram drag-and-drop GUI based on SEMMA model development process&lt;br&gt;Integrated model comparison&lt;br&gt;SAS scoring logic including preliminary transformations</td>
</tr>
<tr>
<td>1996</td>
<td>Pre-EM</td>
<td>SAS develop Neural Network Application (NNA)</td>
</tr>
<tr>
<td>1994</td>
<td>Pre-EM</td>
<td>%TNN feed forward and %TREEDISC macros by Warren Sarie</td>
</tr>
</tbody>
</table>
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.

Click New Project.
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.

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.
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.

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

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.
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.

EXPLORE DATA
Drag the HMEQ data set into the diagram.
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.

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.
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.
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).

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.

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**.

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.
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.

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.

```sas
%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.

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 (VDMMML) 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.

```sas
/* 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.
*/
/*---------------------------------------------------------------*
* The em_viya_assess 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_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.

![Diagram](image)

Our results should look similar to the following.
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";
```
Connect our Viya Tree1 Code node and run it to get results similar to those below.
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.

```sas
/* 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.
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.

![SAS Viya ROC curve](image)

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


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

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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.