SAS® Visual Analytics 8.2: Working with SAS® Visual Data Mining and Machine Learning

About SAS Visual Data Mining and Machine Learning

What Is SAS Visual Data Mining and Machine Learning

SAS Visual Data Mining and Machine Learning is an add-on to SAS Visual Analytics and SAS Visual Statistics that enables you to develop and test models using the in-memory capabilities of SAS servers. SAS Visual Analytics enables you to explore, investigate, and visualize data sources to uncover relevant patterns. SAS Visual Data Mining and Machine Learning extends these capabilities by creating, testing, and comparing models based on the patterns discovered in SAS Visual Analytics. SAS Visual Data Mining and Machine Learning can export the score code or ASTORE, before or after performing model comparison, for use with other SAS products and to put the model into production.

Benefits of Using SAS Visual Data Mining and Machine Learning

SAS Visual Data Mining and Machine Learning enables you to rapidly create powerful data mining and machine learning models in an easy-to-use, web-based interface. SAS Visual Statistics provides a model comparison tool that works on the SAS Visual Data Mining and Machine Learning objects. The model comparison tool enables you to evaluate the relative performance of two or more models against each other and to choose a champion model. A wide variety of model selection criteria is available. Regardless of whether you choose to perform a model comparison, you are able to export the model. With the exported model, you can easily apply your model to new data.

Modeling Information

Available Models

The following models are available in SAS Visual Data Mining and Machine Learning:

- Factorization Machine on page 12 implements a factorization model to effectively handle large, sparse data sets.
- Forest on page 14 is an ensemble model that contains a specified number of decision trees. Each tree is trained on only part of the available data.
- Gradient Boosting on page 18 is an iterative approach that creates multiple trees. Each tree is based on an independent sample of the data.
- Neural Network on page 21 creates a series of connections that are designed to mimic the biological structures of the human brain.
- Support Vector Machine on page 25 creates a set of hyperplanes that maximize the distance between two classes.

Overview of Common Information

SAS Visual Data Mining and Machine Learning uses the same basic structures as SAS Visual Statistics. For information about variables, validation data, missing values, filter variables, score code, and predicted values, see the SAS Visual Statistics documentation.

Nondeterministic Behavior

Some SAS Visual Data Mining and Machine Learning models are created with a nondeterministic process. This means that you might experience different displayed results when you run a model, save that model, close the model, and re-open the report or print the report at a later time. This includes creating a model in SAS Visual Analytics and later viewing it in SAS Report Viewer.

When you create a pipeline for use in Model Studio, a stable version of the model is created.

Model Export and ASTORE Information

When you export a SAS Visual Data Mining and Machine Learning model, an Analytic Store (ASTORE) table is created and saved in the Models library of your CAS server. In addition, if any DATA step code is needed to score the model, it is downloaded to your browser's `Downloads` folder.

To generate model score code and score a model, complete the following steps:

1. Right-click in the plot on the left side of the canvas, and select **Export model**. The DATA step code that is required is saved to your browser's `Downloads` folder. You will need this code in a later step.
2. In the Export Model window, specify a base name for all ASTORE tables. Click **OK**. The name of an ASTORE table is case sensitive.

   If you are exporting a model that uses derived values from other models, you can have multiple ASTORE tables. The name of each ASTORE table is the specified base name with appended numbers to create a unique table name.

3 Before completing the remaining steps, open the downloaded code, modify the macro variables as indicated in the code comments, and run the code. If running the code is successful, then you do not need to complete the remaining steps.

Note: To import your model into Model Studio, skip this step, and complete the remaining steps.

4 Sign on to SAS Studio. Start a CAS session that connects to the same server used by SAS Visual Analytics. Your SAS administrator can provide you with the name of the CAS server. For example, the code below connects to the CAS server on port 5570:

```
/* start a CAS session and assign the libnames */
options cashost="<myCASserver>" casport=5570;
cas mysess;

cas;
caslib _all_ assign;
```

5 Load your ASTORE table from the Models library. For example, the code below loads the Gradient_Boosting_1.sashdat ASTORE table. The name and case of the file specified in CASDATA must be an exact match with the name and case of the file that was created when you exported the model:

```
/* load the astore table for scoring
The value of the CASOUT option is used for other commands */
proc casutil;
    Load casdata="Gradient_Boosting_1.sashdat" incaslib="models"
    casout="gb" outcaslib=casuser;
quit;

proc astore;
    describe rstore=casuser.gb epcode="</u/userName/>gb_scorecode.sas";
quit;
```

The location specified in the EPCODE argument must point to a location where you have Write access.

6 If you want to import this code into Model Studio, open the SAS file that was just created, and delete the KEEP statement. Deleting the KEEP statement includes all variables in the scoring action.

Note: Without performing this step, the score data contains only the predicted values and warnings without any identifying columns.

7 Run PROC ASTORE on the table that was exported from SAS Visual Data Mining and Machine Learning. If you want more than the predicted columns, you must use the EPCODE or COPYVARS argument in PROC ASTORE.

```
/* run proc astore, referencing the altered epcode
to keep some or all of the variables */
proc astore;
    score data=casuser.tempdata
    out=CASUSER.gb_scored rstore=casuser.gb
    epcode="</u/userName/>gb_scorecode.sas";
run;
```

If you did not perform any preprocessing on your data, you can use the raw scoring data.

Note: The ASTORE model will fail when the length of a category target variable name plus the length of the measurement level name are excessively long.

For more information, see PROC ASTORE documentation in SAS Visual Data Mining and Machine Learning: Procedures.

Autotuning

To create a good statistical model, you must make decisions about which model and parameters to use. You can try a trial-and-error approach or you can rely on experience and personal preference. However, neither of these guarantees that you will find the best model for your data. SAS Visual Data Mining and Machine Learning can automate the process of determining optimum model parameters.

Autotuning is the process of automatically and algorithmically adjusting model parameters to create a set of competing versions of one particular model. Those competing versions are then compared to determine which set of parameters produces the best model.

When autotuning a model, you can specify the following options:

Maximum seconds
The maximum amount of time the model will run in seconds. You must specify an integer between 1 and 2,147,483,647. The autotuning algorithm always runs for a minimum of 60 seconds, even if you specify a value smaller than 60.

Maximum iterations
Each iteration, a set of models is created to be evaluated against each other. This property determines the number of sets of models that are created. You must specify an integer between 1 and 2,147,483,647.

Maximum evaluations
The maximum number of different models that are created for evaluation. You must specify an integer between 3 and 2,147,483,647.

Every model available for SAS Visual Data Mining and Machine Learning in SAS Visual Analytics can be autotuned. For all models, you can limit the maximum amount of time, training iterations, and model evaluations performed during autotuning. This prevents server resources from being monopolized by a single user or task. Autotuning does run a slight risk of overfitting a model, especially when no partitioning is used.

The parameters that can be tuned are in the following table:

Factorization Machine				
	Parameter	Minimum Value	Maximum Value	Notes
	Factor count	5	30	Only multiples of 5 are evaluated
	Maximum iterations	10	200	Only multiples of 10 are evaluated
	Learn step	0.000001	1	Only multiples of 10 are evaluated
Forest				
	Parameter	Minimum Value	Maximum Value	Notes
	Number of trees	20	150	Integers
	Bootstrap	0.1	0.9	Real values
	Number of predictors to split nodes	1	100	Will choose the lesser of 100 or maximum number of inputs
	Maximum levels	2	30	Integers
Gradient Boosting				

	Parameter	Minimum Value	Maximum Value	Notes
	Number of trees	20	150	Integers
	Learning rate	0.01	1	Real values
	Subsample rate	0.1	1	Real values
	Lasso	0	10	Real values
	Ridge	0	10	Real values
	Number of predictors to split nodes	1	Number of inputs.	Integers
	Auto-stop method	N/A	N/A	All possible methods are evaluated
Neural Network				
	Parameter	Minimum Value	Maximum Value	Notes
	Number of hidden layers	0	2	Integers
	Number of neurons	1	100	Will choose the lesser of 100 and three times the number of inputs
	L1	0	10	Real values
	L2	0	10	Real values
	Learning rate	0.000001	0.1	Real values
	Annealing rate	1.0e-13	0.01	Real values
	Maximum iterations	10	200	Integers
	Auto-stop method	N/A	N/A	All possible methods are evaluated
Support Vector Machine				
	Parameter	Minimum Value	Maximum Value	Notes
	Kernel function	N/A	N/A	All possible kernel functions are evaluated
	Penalty value	1.0e-10	100	Real values
Decision Tree				
	Parameter	Minimum Value	Maximum Value	Notes
	Maximum levels	2	20	Integers
	Predictor bins	20	200	Integers

Getting Started with SAS Visual Data Mining and Machine Learning

Overview

This is a brief overview of using SAS Visual Data Mining and Machine Learning to derive a new variable, create three different models, and compare those models. This example uses the Framingham Heart Study data set, located at http://support.sas.com/documentation/onlinedoc/viya/examples.htm, to compare the performance of the forest, gradient boosting, and neural network models against each other. The goal is to predict a person's age of death based on a collection of health factors. These factors include gender, weight, height, smoking status, blood pressure, and more. The focus of this example is how to use SAS Visual Data Mining and Machine Learning, not how to build the best model.

Download the Sample Data

1. In a web browser, navigate to http://support.sas.com/documentation/onlinedoc/viya/examples.htm.
2. Download the file **heart.csv** to your local machine.

Create the Report

This example assumes that you have already signed in to SAS Home.

Here are the steps to create the report:

1. From SAS Home, click **Explore and Visualize Data**. This opens SAS Visual Analytics, from which you open a data source, create a model, or load a project.
2. Click **Data** in the welcome window. A window appears that enables you to select and load the data source for this project.
3. On the **Import** tab, click **Local File**. Navigate to the location where you saved **heart.csv** and select **heart.csv**.
4. In the Open Data Source window, click **Import Item**. After the table is successfully imported, click **OK**.
5. By default, the report is named `Report 1`, which is displayed in the upper left corner. Before continuing with the example, rename the project by saving it.

   Click ⋮ in the upper right, and then select **Save**. This opens the Save As window. In the **Folders** pane, navigate to a location where you have Write permission. In the **Name** field, enter `Heart Study`, and click **Save**.

   Typically, you can save your work in `My Folder`.

For more information about reports, see *SAS Visual Analytics 8.2: Designing Reports*.

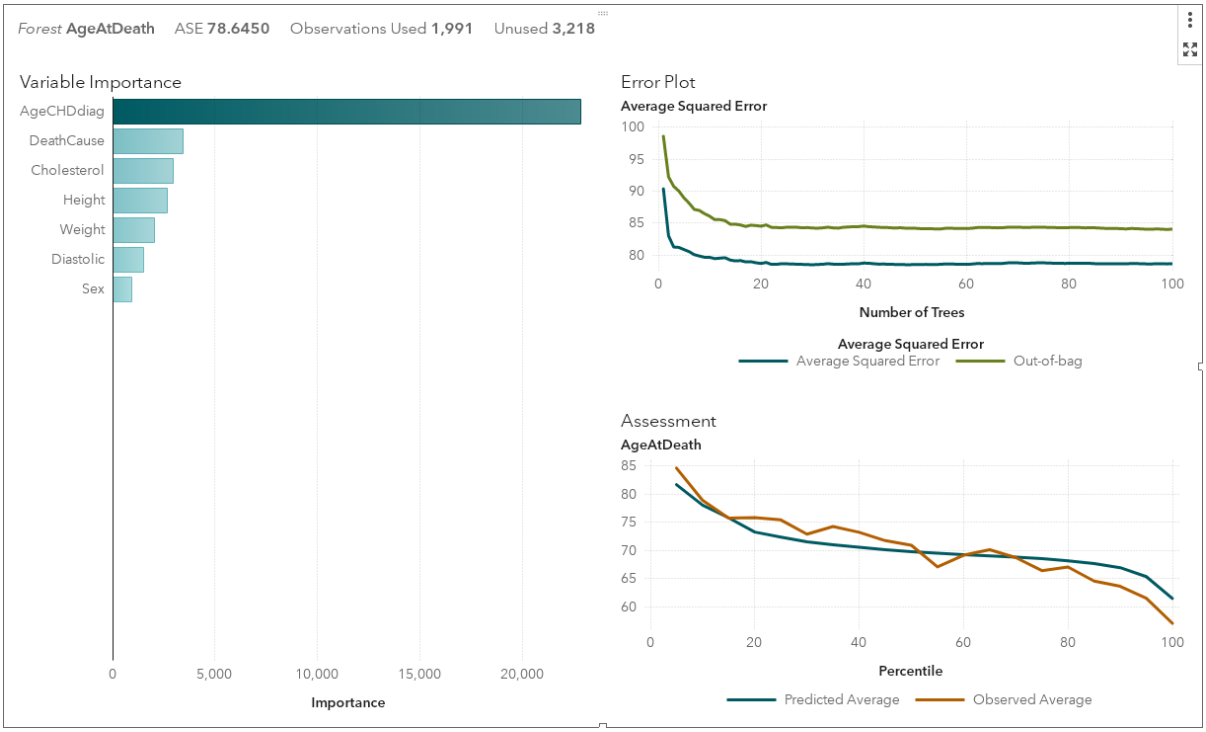
Create a Forest

Here are the steps to create a forest:

1 From the left pane, click the **Objects** icon to select an object. Drag the icon onto the canvas to create a forest.

2 Click **Roles** in the right pane. For **Response**, click **Add**, and select **AgeAtDeath**.

3 For **Predictors** field, click **Add**, and select **Diastolic**, **Weight**, **Height**, **Cholesterol**, **AgeCHDdiag**, **Sex**, and **DeathCause**. Click **OK**. The forest automatically updates.



4 Click to enter maximize mode. For more information, see Maximizing Objects .

In explore mode, you can view more detailed information about your model. This includes the variable importance values and error metric results.

5 Click to exit maximize mode.

6 Save the project.

Create a Gradient Boosting Model

Here are the steps to create a gradient boosting model:

1 Click the + icon to add a new page.

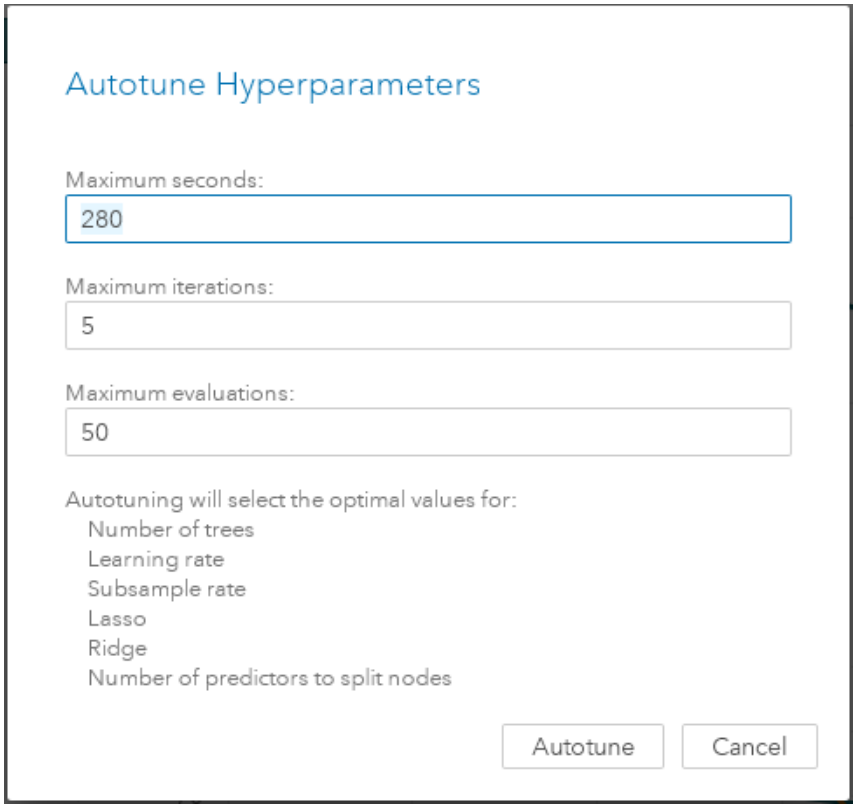
2 From the left pane, click the Objects icon to select an object. Drag the icon onto the canvas to create a gradient boosting.

3 In this example, the variable of interest is **AgeAtDeath**.

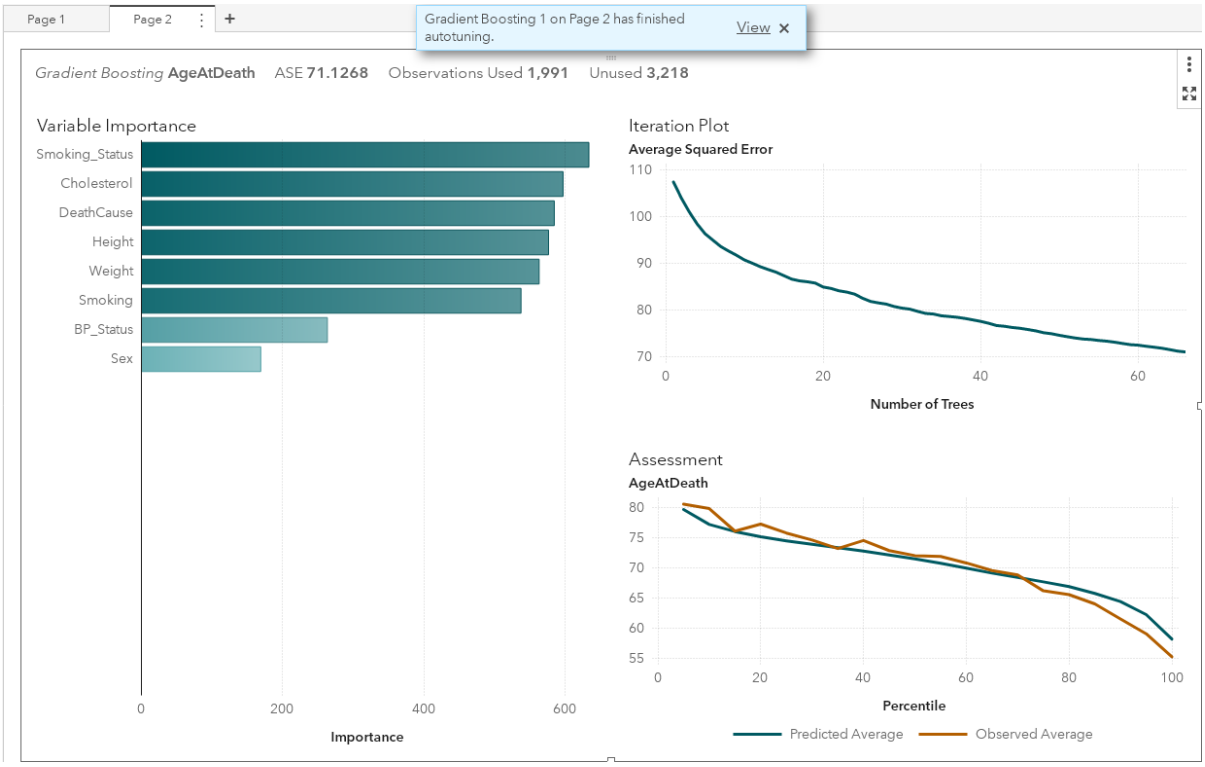
Click **Roles** in the right pane. For **Response**, click **Add**, and select **AgeAtDeath**.

4 In the **Predictors** field, click **Add**, and select **BP_Status**, **DeathCause**, **Sex**, **Smoking_Status**, **Cholesterol**, **Height**, **Smoking**, and **Weight**.

5 In the right pane, select **Options** and click **Autotune**.



When you autotune your model, SAS Visual Data Mining and Machine Learning automatically and algorithmically determines the optimal values for the specified properties. In the Autotune Hyperparameters window, click **Autotune**.



The Assessment window indicates that the observed average and predicted average vary slightly across the bins.

6 Save the project.

Create a Neural Network

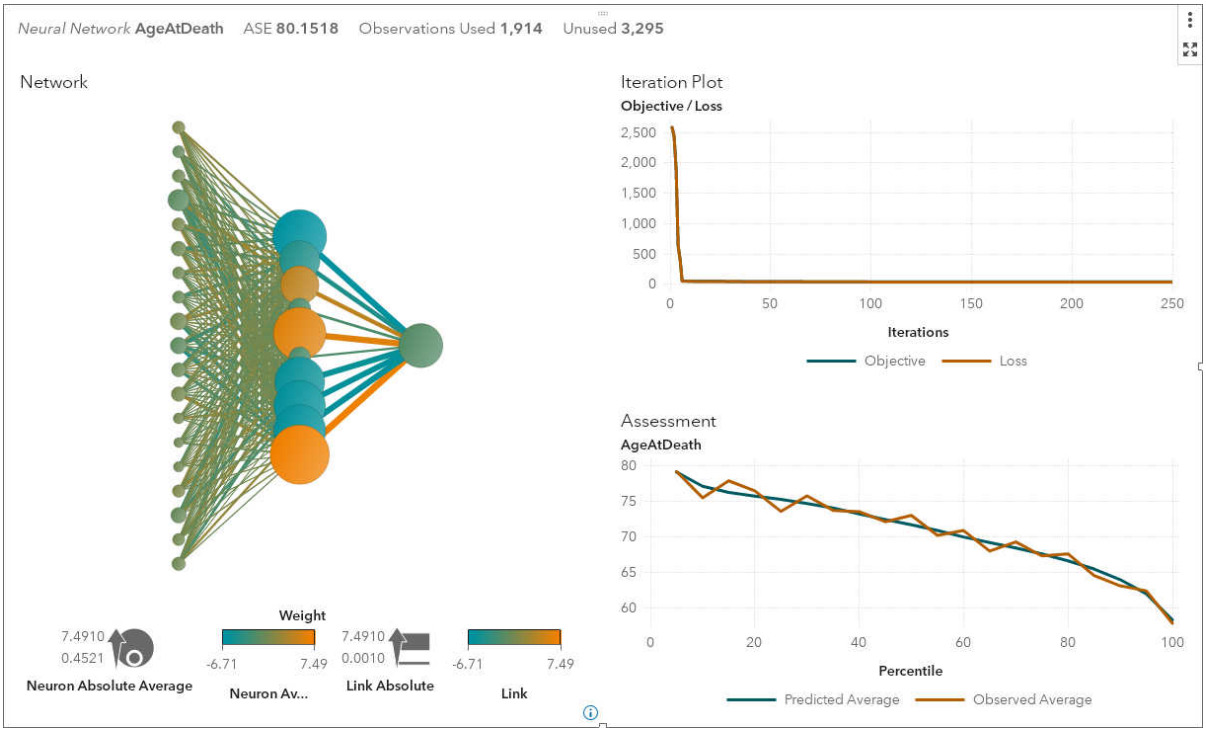
Here are the steps to create a neural network:

1 Click the + icon to add a new page.

2 From the left pane, click the Objects icon to select an object. Drag the icon onto the canvas to create a neural network.

3 Click **Roles** in the right pane. For **Response**, click **Add**, and select **AgeAtDeath**.

4 In the **Predictors** field, click **Add**, and select **BP_Status**, **DeathCause**, **Sex**, **Smoking_Status**, **Cholesterol**, **Height**, **Smoking**, and **Weight**.



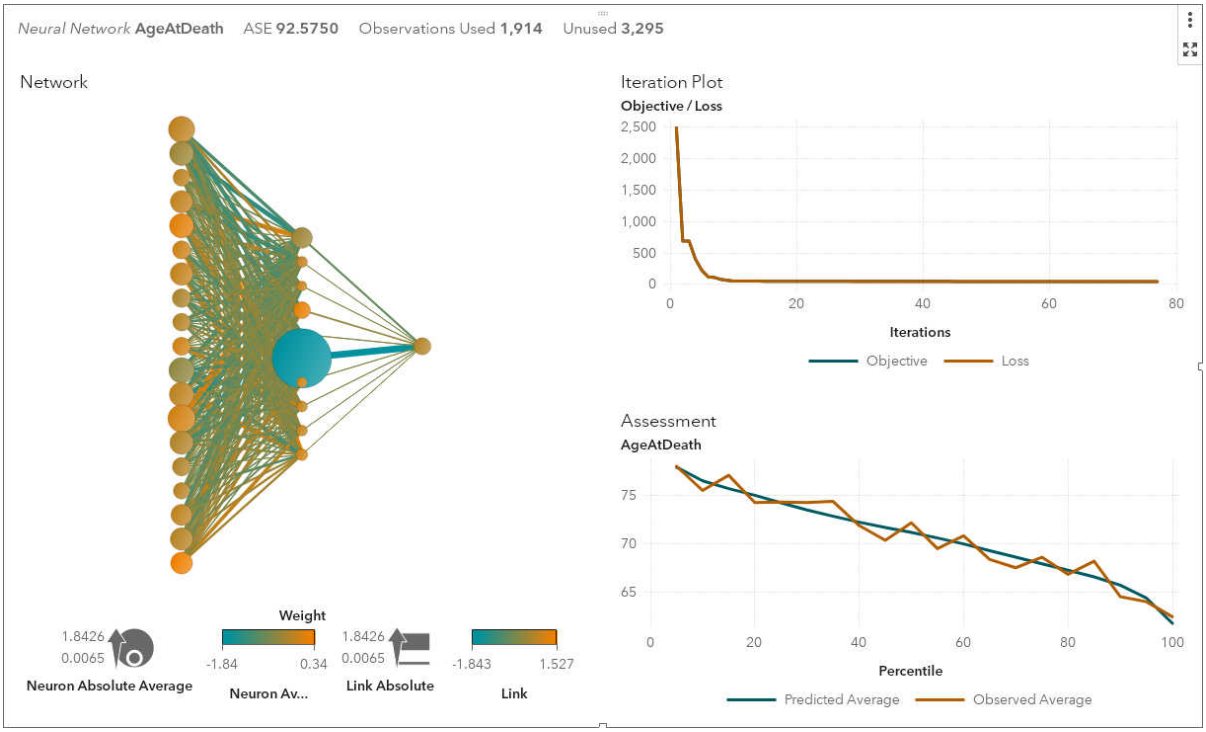
5 Click **Options** in the right pane. The **Distribution** property enables you to specify the distribution of the response variable and to build a model based on that distribution. The default distribution is **Normal**.

To determine whether the normal distribution applies to the response variable, click in the **Data** pane. In the Measure Details window, select **AgeAtDeath**.

6 Notice that **Age at Death** is not normally distributed and is slightly skewed left. Click **Close**.

7 Although the distribution is not exactly Poisson, use the Poisson distribution for this example. For the **Distribution** property, select **Poisson**.

Note: You are encouraged to repeat this example with different distributions and standardization techniques and compare their performances and to familiarize yourself with SAS Visual Data Mining and Machine Learning.



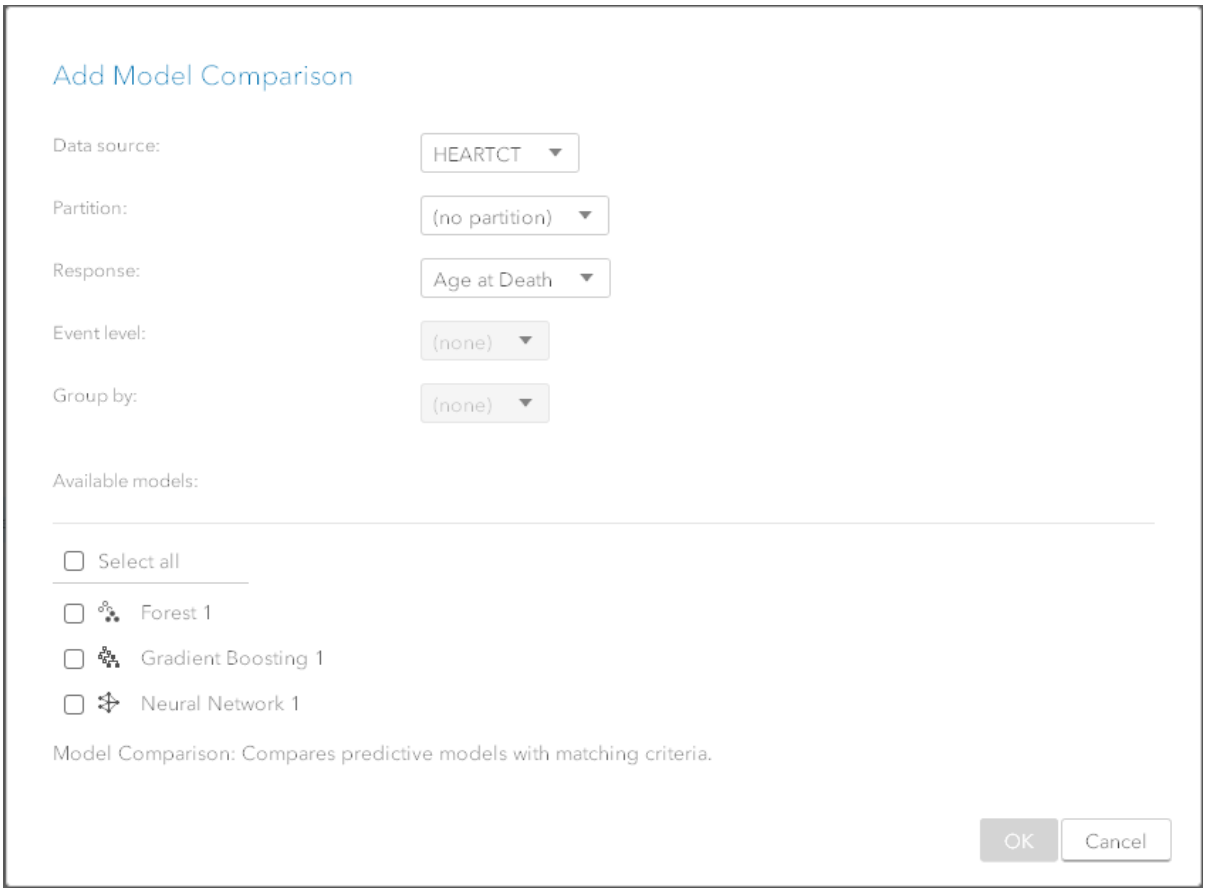
8 Save the project.

Perform a Model Comparison

Here are the steps to perform a model comparison:

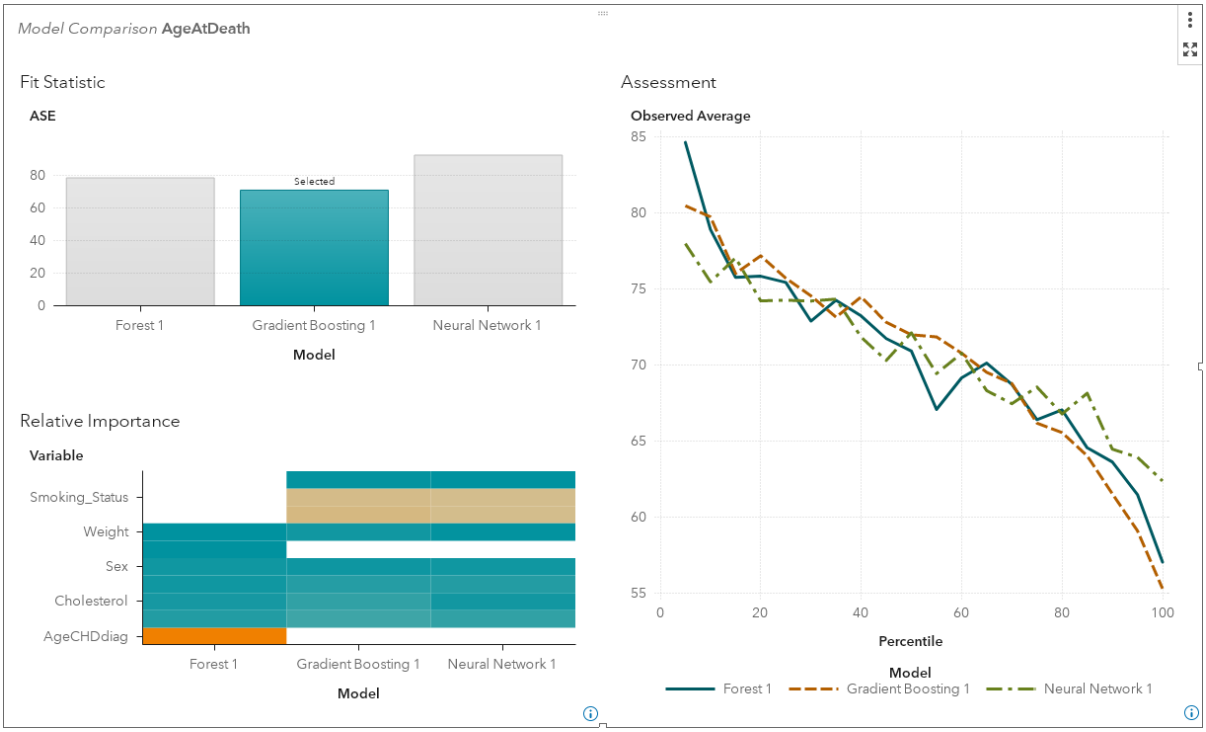
1 Click the + icon to add a new page.

2 From the left pane, click the Objects icon to select an object. From the **Visual Statistics** group, drag the icon onto the canvas to create a model comparison.



The **Response** variable is already set to **Age at Death**, and **Event level** and **Group by** are unavailable. With these settings, the available models are **Forest 1**, **Gradient Boosting 1**, and **Neural Network 1**.

3 Select **Select all**, and click **OK**.



4 By default, the fit statistic average squared error, **ASE**, is used to compare the models. The other available fit statistics are **SSE** and **Observed Average**. Because smaller values are preferred, the gradient boosting model is chosen as the champion when **ASE** or **SSE** is the criterion. The models are very similar.

   When the fit statistic is **Observed Average**, the **Percentile** slider is available. This slider specifies the percentile where the observed average and predicted average are compared. The champion model varies by percentile.

   If you view the Assessment plot, both the **Observed Average** and **Predicted Average** plots show that in some cases the models are significantly different.

5 Now that you have a champion model, you can export the model score code for that model to score new data.

   Here are the steps to export the model score code:

   a Open **Gradient Boosting 1** by navigating to **Page 2**.

   b Right-click in the Variable Importance window, and select **Export model**.

   c In the Export Model window, click **OK**.

   d The model is stored as an ASTORE. For more details, see the PROC ASTORE documentation and Model Export and Astore Information on page 2 .

Working with Factorization Machines

Overview of Factorization Machines

A factorization machine is a predictive model that creates factorization model. By modeling all variable interactions with factorized parameters, factorization machines are able to handle large, very sparse data and can be trained in linear time.

A common application of factorization machines is for recommendation engines. A factorization machine can consider all items that a user has rated and predict ratings for other items.

How to Create a Factorization Machine

To create a factorization machine, complete the following steps:

1 Drag the Factorization Machine icon onto the canvas.

2 On the right pane, select the pane. Specify a single measure variable as the **Response** variable.

3 Specify at least one two category variables and any number of measure variables in the **Predictors** field.

Factorization Machine Options

The following properties are available for the factorization machine:

Autotune
enables you to specify the constraints that control the autotuning algorithm. The constraints determine how long the algorithm can run, how many times the algorithm can run, and how many model evaluations are allowed.

The autotuning algorithm selects the **Factor count**, **Maximum iterations**, and **Learn step** values that produce the best model.

Factor count
specifies the number of factors to use for each predictor level during the fitting process.

Nonnegative factorization
forces nonnegative factors with bias terms to be set to zero. This can improve performance when the training data is very sparse.

Maximum iterations
specifies the maximum number of iterations before model training stops.

Learn step
specifies the minimum change in the loss function in order for model estimation to proceed.

Number of bins (5–100)
specifies the number of bins that the response variable is binned into when you do not use the default number of bins.

Tolerance
specifies the minimum change in optimization function value in order for optimization to proceed.

Factorization Machine Model Display Options

The following display properties are available for the factorization machine:

Plot layout
specifies how the results windows are displayed on the canvas. **Fit** aligns all of the objects on the canvas automatically. **Stack** displays the objects as if they are in a slide deck. Only one object is displayed at a time. When **Stack** is specified, a control bar lets you move between objects.

Show recommendations for
For each category variable, the factorization machine displays either the most recommended or least recommended categories. When **Nonnegative factorization** is enabled, the recommendation is sorted by the sum of all factors.

Count
specifies how many recommended categories are displayed.

Plot to show
specifies which assessment plot is displayed.

Use histogram
specifies whether the Scored Response display is a histogram instead of a plot.

Factorization Machine Results

Iteration Plot

The Iteration Plot displays the value of the loss function at each iteration in the model building process.

Scored Response

The Scored Response plots the computed responses against the true, observed values. You can also view this information as a histogram.

Relative Importance

The Relative Importance plots the importance value of each input variable. The variables are ranked using their first-split log worth when applied to the scored training data.

Assessment Plot

For a factorization machine, the Assessment plot plots the predicted average and observed average response values against the binned data. Use this plot to reveal any strong biases in your model. Large differences in the predicted average and observed average values can indicate a bias.

Rankings Plot

The Rankings Plot is used by the factorization machine to display the top or bottom ranked event levels for category variables.

Details Table

When you click ⛶ in the model toolbar in the upper right of the canvas, the details table is displayed at the bottom of the canvas. The details table contains the following information:

Model Information
A brief description of the elements of the model.

Iteration History
The loss function iteration results. This tab shows the value of the loss function at each iteration.

Measures
Provides summary statistics for the input measure variables.

Categories
Provides information about the input category variables.

Final Exact Loss
Provides error statistics for the model.

Factors
Provides the bias and factor weights for each observation in the input data.

Assessment
The binned assessment results that are used to generate the Assessment plot.

Assessment Statistics
The value of any assessment statistics computed for the model.

Working with Forests

Overview of Forests

A forest is an ensemble model that contains a specific number of decision trees. To ensure that a forest does not overfit the data, two key steps are taken. First, each tree in the forest is built on a different sample of the training data. Second, when splitting each node, a set of candidate inputs for the split are selected at random, and the best split is selected from those. Other than these two steps, the trees in a forest are trained like standard trees.

The training data for each tree in the forest excludes some of the available data. This data is called the out-of-bag sample. For each leaf in a tree, the prediction for an interval target is the average of the target values. The

posterior probability for a target category of a nominal target is the proportion of the target category among observations in the leaf. These values can be based on training some of the out-of-bag sample. To predict the target value of a new observation, that observation is fed through each tree, and the predicted value is determined by the **Vote** property.

By default, in the SAS procedures that implement forests, the training sample is used.

How to Create a Forest

To create a forest, complete the following steps:

1. Drag the icon onto the canvas.
2. On the right pane, select the pane. Specify a single variable as the **Response** variable.
3. Specify at least one measure variable or category variable in the **Predictors** field.
4. Optionally, you can specify **Partition ID**.

Forest Options

The following properties are available for the forest:

Event level
enables you to choose the event level of interest. When your category response variable contains more than two levels, SAS Visual Data Mining and Machine Learning treats all observations in the level of interest as an event and all other observations as nonevents.

Autotune
enables you to specify the constraints that control the autotuning algorithm. The constraints determine how long the algorithm can run, how many times the algorithm can run, and how many model evaluations are allowed.

The autotuning algorithm selects the **Number of trees**, **Bootstrap**, **Number of predictors to split nodes**, and **Maximum levels** values that produce the best model.

Number of trees
specifies the number of trees in the forest.

Bootstrap
specifies the bootstrap value. This is the percentage of data used to grow each tree in the forest.

Vote
specifies the method used to determine the predicted value for each observation when you specify a classification response. For **Majority** voting, the event level that is predicted most often is selected. For **Probability** voting, the average of the probabilities predicted by each tree is computed and compared against the **Prediction cutoff** value.

Measure responses always use the average across all trees as the predicted value.

Splitting criterion
specifies the splitting criterion used to create branches.

Set fixed number of predictors to split node
specifies whether the default number of predictors are used to split nodes in each decision tree of the forest.

Number of predictors to split nodes
specifies the number of predictors used in each decision tree of the forest.

Missing assignment
specifies how observations with missing values are included in the model.

Minimum value
specifies the minimum number of observations allowed to have missing values before missing values are treated as a distinct category level.

Maximum branches
specifies the maximum number of branches allowed when splitting a node.

Maximum levels
specifies the maximum depth of the decision tree.

Leaf size
specifies the minimum number of observations allowed in a leaf node.

Predictor bins
specifies the number of bins used to categorize a predictor that is a measure variable.

Number of bins (5–100)
specifies whether you want to use the default number of bins or you want to set your own value. By default, measure variables are grouped into 20 bins.

Tolerance
specifies the tolerance value that is used to determine the convergence of the iterative algorithm that estimates the percentiles. Specify a smaller value to increase the algorithmic precision.

Forest Model Display Options

The following display properties are available for forests:

Plot layout
specifies how the results windows are displayed on the canvas. **Fit** aligns all of the objects on the canvas automatically. **Stack** displays the objects as if they are in a slide deck. Only one object is displayed at a time. When **Stack** is specified, a control bar lets you move between objects.

Statistic to show
specifies which assessment statistic to display in the model.

Plot to show
specifies which assessment statistic is plotted.

Y axis
specifies which statistic is plotted in the Lift plot.

Forest Results

Variable Importance

The Variable Importance plot plots the importance of each variable as measured by its loss reduction variable importance value.

Error Plot

The Error Plot compares the model and out-of-bag misclassification rates. The out-of-bag misclassification rates can be used as a validation measurement to ensure that the model is not overfitting the input data. The error plot displays the misclassification rate across the number of trees for the entire model. The assessment plot displays the assessment results for just a single target level.

Assessment Plot

For a measure response, the Assessment plot displays the predicted average and observed average response values against the binned data. Use this plot to reveal any strong biases in your model. Large differences in the predicted average and observed average values can indicate a bias.

For a category response, the Assessment plot displays either the Lift, ROC, or Misclassification rate.

Lift is the ratio of the percent of captured responses within each percentile bin to the average percent of responses for the model. Similarly, *cumulative lift* is calculated by using all of the data up to and including the current percentile bin.

A receiver operating characteristic (ROC) chart displays the ability of a model to avoid false positive and false negative classifications. A false positive classification means that an observation has been identified as an event when it is actually a nonevent (also referred to as a Type I error). A false negative classification means that an observation has been identified as a nonevent when it is actually an event (also referred to as a Type II error).

The *specificity* of a model is the true negative rate. To derive the false positive rate, subtract the specificity from 1. The false positive rate, labeled **1 – Specificity**, is the X axis of the ROC chart. The *sensitivity* of a model is the true positive rate. This is the Y axis of the ROC chart. Therefore, the ROC chart plots how the true positive rate changes as the false positive rate changes.

A good ROC chart has a very steep initial slope and levels off quickly. That is, for each misclassification of an observation, significantly more observations are correctly classified. For a perfect model, one with no false positives and no false negatives, the ROC chart would start at (0,0), continue vertically to (0,1), and then horizontally to (1,1). In this instance, the model would correctly classify every observation before a single misclassification could occur.

The ROC chart includes two lines to help you interpret the ROC chart. The first line is a baseline model that has a slope of 1. This line mimics a model that correctly classifies observations at the same rate it incorrectly classifies them. An ideal ROC chart maximizes the distance between the baseline model and the ROC chart. A model that classifies more observations incorrectly than correctly would fall below the baseline model. The second line is a vertical line at the false positive rate where the difference between the Kolmogorov-Smirnov values for the ROC chart and baseline models is maximized.

The misclassification plot displays how many observations were correctly and incorrectly classified for each value of the response variable. When the response variable is not binary, the logistic regression model considers all levels that are not events as equal. A significant number of misclassifications could indicate that your model does not fit the data. Tied votes might lead to small differences in the reported misclassification rate.

Details Table

From the object toolbar, select the ⛶ icon to enter maximize mode and display the details table at the bottom of the canvas. The details table contains the following information:

Variable Importance
A summary of the importance and standard deviation for each variable in the model.

Error Metric
The calculated error rate for every forest model sorted by the number of trees in the forest.

Lift
The binned assessment results that are used to generate the Lift plot.

Misclassification
A summary of the correct and incorrect classifications for the model.

ROC
The results that are used to generate the ROC plot.

Assessment Statistics
The value of any assessment statistics computed for the model.

Working with Gradient Boosting Models

About Gradient Boosting Models

Gradient boosting is an iterative approach that creates multiple trees where, typically, each tree is based on an independent sample without replacement of the data. The gradient boosting model hones its predictions by minimizing a specified loss function, such as average square error. The first step creates a baseline tree. Each subsequent tree is fit to the residuals of the previous tree, and the loss function is minimized. This process is repeated a specific number of times. The final model is a single function, which is an aggregation of the series of trees that can be used to predict the target value of a new observation.

The term “stochastic gradient boosting” refers to training each new tree based on a subsample of the data. This typically results in a better model. For gradient boosting models, each new observation is fed through a sequence of trees that are created to predict the target value of each new observation.

How to Create a Gradient Boosting Model

To create a gradient boosting model, complete the following steps:

1. Drag the icon onto the canvas.
2. On the right pane, select the pane. Specify either a single measure variable or a single category variable as the **Response** variable.
3. Specify at least one or more measure or category variables in the **Predictors** field.
4. Optionally, you can specify **Partition ID** variable.

Gradient Boosting Options

On the **Options** tab, you can specify the following options:

Event level
enables you to choose the event level of interest. When your category response variable contains more than two levels, SAS Visual Data Mining and Machine Learning treats all observations in the level of interest as an event and all other observations as nonevents.

Autotune
enables you to specify the constraints that control the autotuning algorithm. The constraints determine how long the algorithm can run, how many times the algorithm can run, and how many model evaluations are allowed.

The autotuning algorithm selects the **Number of trees**, **Learning rate**, **Subsample rate**, **Lasso**, **Ridge**, and **Number of predictors to split nodes** values that produce the best model. Additionally, if and only if data partitioning is applied, then the **Auto-stop method** is also autotuned.

Auto-stop method
specifies the method that controls early termination of the model building algorithm. For **Stagnation**, the algorithm terminates when there is no improvement in a specified number of successive validation error calculations. For **Tolerance**, the algorithm terminates when the magnitude of the relative change in validation error is less than a specified value for a specified number of calculations.

Auto-stop iterations
specifies the number of successive validation error calculations required to trigger the **Stagnation** or **Tolerance** auto-stop methods.

Tolerance value
specifies the minimum change required in the validation error required to trigger the **Tolerance** auto-stop method.

Number of trees
specifies the number of trees in the gradient boosting model.

Learning rate
specifies the learning rate used to update the gradient boosting model.

Subsample rate
specifies the subsample rate to create each tree in the gradient boosting model.

Lasso
specifies the LASSO (Least Absolute Shrinkage and Selection Operator) parameter that is used to select the regression coefficients.

Ridge
specifies the ridge value that is used to update the gradient boosting model.

Set fixed number of predictors to split node
specifies whether the default number of predictor variables are considered when determining node splits.

Number of predictors to split nodes
specifies the number of predictors that are considered when splitting a node.

Missing assignment
specifies how observations with missing values are included in the model.

Minimum value
specifies the minimum number of observations allowed to have missing values before missing values are treated as a distinct category level.

Maximum branches
specifies the maximum number of branches allowed when splitting a node.

Maximum levels
specifies the maximum depth of the decision tree.

Leaf size
specifies the minimum number of observations allowed in a leaf node.

Predictor bins
specifies the number of bins used to categorize a predictor that is a measure variable.

Number of bins (5–100)
specifies whether you want to use the default number of bins or you want to set your own value. By default, measure variables are grouped into 20 bins.

Prediction cutoff
specifies the value at which a computed probability is considered an event.

Tolerance
specifies the tolerance value that is used to determine the convergence of the iterative algorithm that estimates the percentiles. Specify a smaller value to increase the algorithmic precision.

Gradient Boosting Model Display Options

The following display properties are available for the gradient boosting model:

Plot layout
specifies how the results windows are displayed on the canvas. **Fit** aligns all of the objects on the canvas automatically. **Stack** displays the objects as if they are in a slide deck. Only one object is displayed at a time. When **Stack** is specified, a control bar lets you move between objects.

Statistic to show
specifies which assessment statistic to display in the model.

Plot to show
specifies which assessment statistic is plotted.

Y axis
specifies which statistic is plotted in the Lift plot.

Gradient Boosting Results

Variable Importance

The Variable Importance plot displays the importance of each variable as measured by its contribution to the change in the residual sum of squared errors value.

Iteration Plot

The Iteration Plot displays the assessment statistic against the number of trees in the gradient boosting model. The iteration plot displays the misclassification rate across the number of trees for the entire model. The assessment plot displays the assessment results for just a single target level.

Assessment Plot

For a measure response, the Assessment plot displays the predicted average and observed average response values against the binned data. Use this plot to reveal any strong biases in your model. Large differences in the predicted average and observed average values can indicate a bias.

For a category response, the Assessment plot displays either the Lift, ROC, or Misclassification rate.

Lift is the ratio of the percent of captured responses within each percentile bin to the average percent of responses for the model. Similarly, *cumulative lift* is calculated by using all of the data up to and including the current percentile bin.

A receiver operating characteristic (ROC) chart displays the ability of a model to avoid false positive and false negative classifications. A false positive classification means that an observation has been identified as an event when it is actually a nonevent (also referred to as a Type I error). A false negative classification means that an observation has been identified as a nonevent when it is actually an event (also referred to as a Type II error).

The *specificity* of a model is the true negative rate. To derive the false positive rate, subtract the specificity from 1. The false positive rate, labeled **1 – Specificity**, is the X axis of the ROC chart. The *sensitivity* of a model is the true positive rate. This is the Y axis of the ROC chart. Therefore, the ROC chart plots how the true positive rate changes as the false positive rate changes.

A good ROC chart has a very steep initial slope and levels off quickly. That is, for each misclassification of an observation, significantly more observations are correctly classified. For a perfect model, one with no false positives and no false negatives, the ROC chart would start at (0,0), continue vertically to (0,1), and then horizontally to (1,1). In this instance, the model would correctly classify every observation before a single misclassification could occur.

The ROC chart includes two lines to help you interpret the ROC chart. The first line is a baseline model that has a slope of 1. This line mimics a model that correctly classifies observations at the same rate it incorrectly classifies them. An ideal ROC chart maximizes the distance between the baseline model and the ROC chart. A model that classifies more observations incorrectly than correctly would fall below the baseline model. The

second line is a vertical line at the false positive rate where the difference between the Kolmogorov-Smirnov values for the ROC chart and baseline models is maximized.

The misclassification plot displays how many observations were correctly and incorrectly classified for each value of the response variable. When the response variable is not binary, the logistic regression model considers all levels that are not events as equal. A significant number of misclassifications could indicate that your model does not fit the data.

Details Table

When you click the icon to enter maximize mode and display the details table at the bottom of the canvas. The details table contains the following information:

Variable Importance
A listing of the variables, their importance to the model, and their standard deviation.

Iteration History
The error function iteration results. This tab shows the value of the error function at each iteration.

Lift
The binned assessment results that are used to generate the Lift plot.

Misclassification
A summary of the correct and incorrect classifications for the model.

ROC
The results that are used to generate the ROC plot.

Assessment
The binned assessment results that are used to generate the Assessment plot.

Assessment Statistics
The value of any assessment statistics computed for the model.

Working with Neural Networks

Overview of Neural Networks

A neural network is a statistical model that is designed to mimic the biological structures of the human brain. Neural networks consist of predictors (input variables), hidden layers, an output layer, and the connections between each of those. Predictors can be connected directly to the output layer, and the creation of those connections are determined by the default activation function. You can specify an activation function for each hidden layer.

How to Create a Neural Network

To create a neural network, complete the following steps:

1. Drag the icon onto the canvas.
2. On the right pane, select the pane. Specify a single variable as the **Response** variable.
3. Specify at least one measure variable or category variable in the **Predictors** field.
4. Optionally, you can specify **Partition ID** or a **Weight** variable.

Neural Network Options

The following properties are available for the neural network:

Event level
enables you to choose the event level of interest. When your category response variable contains more than two levels, SAS Visual Statistics treats all observations in the level of interest as an event and all other observations as nonevents.

Autotune
enables you to specify the constraints that control the autotuning algorithm. The constraints determine how long the algorithm can run, how many times the algorithm can run, and how many model evaluations are allowed.

The autotuning algorithm selects the **Number of hidden layers**, **Number of neurons**, **L1**, **L2**, and **Maximum iterations** values that produce the best model. Additionally, if and only if data partitioning is applied, then the **Auto-stop method** is also autotuned.

Include missing
specifies whether observations with missing values are included in the model. For category predictors, missing values are assigned their own measurement levels. For measure predictors, missing values are imputed with the measure variable's mean.

Distribution
specifies the distribution used to model the response variable. This option is available only when the response is a measure variable.

Output activation function
specifies the activation function that is used to create the output layer when using a measure response. The exponential activation function is used with a Gamma or Poisson distribution. This option is available only when you specify a measure response and the **Normal** distribution.

Standardization
specifies the method used to standardize the measure predictors.

Maximum iterations
specifies the maximum number of optimization iterations that are used.

Maximum time (sec)
specifies the maximum time-out value.

Optimization method
specifies the optimization method used to train the neural network. You can specify either the **SGD** or **LBFGS** method.

Learning rate
specifies the learning rate parameter used in stochastic gradient descent optimization.

Annealing rate
specifies the annealing rate parameter used in stochastic gradient descent optimization.

L1
specifies the L1 regularization parameter.

L2
specifies the L2 regularization parameter.

Number of hidden layers
specifies the number of hidden layers in the model. The maximum value allowed is 2 if the **Optimization method** is LBFGS. The maximum value is 5 if the **Optimization method** is SGD.

Allow direct connections between input and target neurons
specifies that each input neuron is connected to each output neuron in the network.

Neurons
specifies the number of neurons in the hidden layer.

Activation function
specifies the activation function used for each neuron's output based on the weighted sum of its inputs. This can be changed for each hidden layer. You can specify either a **Hyperbolic Tangent**, **Identity**, **Sine**, **Exponential**, **Logistic**, **Rectifier**, or **Softplus** function as the activation function.

Neural Network Model Display Properties

The following display properties are available for neural networks:

Plot layout
specifies how the results windows are displayed on the canvas. **Fit** aligns all of the objects on the canvas automatically. **Stack** displays the objects as if they are in a slide deck. Only one object is displayed at a time. When **Stack** is specified, a control bar lets you move between objects.

Statistic to show
specifies which assessment statistic to display in the model.

Neuron labels
specifies whether the neurons in the Network diagram are labeled.

Neuron layout
specifies how the neurons are plotted in the Network diagram.

Horizontal spacing
specifies the amount of horizontal space that exists between nodes.

Vertical spacing
specifies the amount of vertical space that exists between nodes.

Percentage of links to display
specifies how many connecting links are displayed in the Network diagram.

Horizontal layout
specifies whether the width of the nodes in the Network diagram are fixed or adjust to the width of the Network diagram.

Number of neurons to display
specifies the maximum number of neurons that are displayed in the Network diagram.

Plot to show
specifies which assessment statistic is plotted.

Y axis
specifies which statistic is plotted in the Lift plot.

Neural Network Results

Network Plot

The Network diagram displays the input nodes, hidden nodes, connections, and output nodes of a neural network. Nodes are represented as circles and links between the nodes are lines connecting two circles. The size of the circle represents the absolute value at that node, relative to the model and the color indicates whether that value is positive or negative. Similarly, the size of the line between two nodes indicates the strength of the link and the color indicates whether that value is positive or negative.

To modify the neural network, right-click on the Network diagram and select one of the following options:

- **Add a hidden layer** — inserts a new hidden layer into the neural network and rebuilds the model.

- **Edit a hidden layer** — specify the hidden layer that you want to modify. In the Edit Hidden Layer window, specify the number of **Neurons** and the **Activation function** for this hidden layer.
- **Remove a hidden layer** — removes a hidden layer from the neural network and rebuilds the model.

Iteration Plot

The Iteration Plot plots the value of the **Objective/Loss** function at each iteration in the network building process.

Relative Importance Plot

The Relative Importance plots the importance value of each input variable. The variables are ranked using their first-split log worth when applied to the scored training data. This plot can be empty if no variables are determined to be important.

Assessment Plot

For a measure response, the Assessment plot displays the predicted average and observed average response values against the binned data. Use this plot to reveal any strong biases in your model. Large differences in the predicted average and observed average values can indicate a bias.

For a category response, the Assessment plot displays either the Lift, ROC, or Misclassification rate.

Lift is the ratio of the percent of captured responses within each percentile bin to the average percent of responses for the model. Similarly, *cumulative lift* is calculated by using all of the data up to and including the current percentile bin.

A receiver operating characteristic (ROC) chart displays the ability of a model to avoid false positive and false negative classifications. A false positive classification means that an observation has been identified as an event when it is actually a nonevent (also referred to as a Type I error). A false negative classification means that an observation has been identified as a nonevent when it is actually an event (also referred to as a Type II error).

The *specificity* of a model is the true negative rate. To derive the false positive rate, subtract the specificity from 1. The false positive rate, labeled **1 – Specificity**, is the X axis of the ROC chart. The *sensitivity* of a model is the true positive rate. This is the Y axis of the ROC chart. Therefore, the ROC chart plots how the true positive rate changes as the false positive rate changes.

A good ROC chart has a very steep initial slope and levels off quickly. That is, for each misclassification of an observation, significantly more observations are correctly classified. For a perfect model, one with no false positives and no false negatives, the ROC chart would start at (0,0), continue vertically to (0,1), and then horizontally to (1,1). In this instance, the model would correctly classify every observation before a single misclassification could occur.

The ROC chart includes two lines to help you interpret the ROC chart. The first line is a baseline model that has a slope of 1. This line mimics a model that correctly classifies observations at the same rate it incorrectly classifies them. An ideal ROC chart maximizes the distance between the baseline model and the ROC chart. A model that classifies more observations incorrectly than correctly would fall below the baseline model. The second line is a vertical line at the false positive rate where the difference between the Kolmogorov-Smirnov values for the ROC chart and baseline models is maximized.

The misclassification plot displays how many observations were correctly and incorrectly classified for each value of the response variable. When the response variable is not binary, the logistic regression model considers all levels that are not events as equal. A significant number of misclassifications could indicate that your model does not fit the data.

Details Table

From the object toolbar, select the icon to enter maximize mode and display the details table at the bottom of the canvas. The details table contains the following information:

Model Information
A summary of the features of the model, including number of observations, number of neurons, weight and bias parameters, and the value of the objective function.

Iteration History
The value of the **Objective** and **Loss** functions at each iteration in the network building process.

Convergence
The convergence criterion that was reached and triggered the termination of the model building process.

Lift
The binned assessment results that are used to generate the Lift plot.

Misclassification
A summary of the correct and incorrect classifications for the model.

ROC
The results that are used to generate the ROC plot.

Assessment
The binned assessment results that are used to generate the Assessment plot.

Assessment Statistics
The value of any assessment statistics computed for the model.

Working with Support Vector Machines

Overview of Support Vector Machines

A support vector machine (SVM) is a machine learning model that is used to perform classification by constructing a set of hyperplanes that maximizes the margin between two classes. SAS Visual Data Mining and Machine learning assigns a continuous probability output to each observation based on its distance to the boundary and which side of the boundary it is on.

How to Create a Support Vector Machine

To create a support vector machine, complete the following steps:

1. Drag the icon onto the canvas.
2. On the right pane, select the pane. Specify a single category variable as the **Response** variable.
3. Specify at least one measure variable or category variable in the **Predictors** field.
4. Optionally, you can specify a **Partition ID** variable.

Support Vector Machine Options

On the **Options** tab, you can specify the following options:

Event level
enables you to choose the event level of interest. When your category response variable contains more than two levels, SAS Visual Data Mining and Machine Learning treats all observations in the level of interest as an event and all other observations as nonevents.

Autotune
enables you to specify the constraints that control the autotuning algorithm. The constraints determine how long the algorithm can run, how many times the algorithm can run, and how many model evaluations are allowed.

The autotuning algorithm selects the **Kernel function** and **Penalty value** values that produce the best model.

Kernel function
specifies the kernel function used for spatial classification.

- **Linear** $K(u, v) = u^T v$.
- **Quadratic** $K(u, v) = (u^T v + 1)^2$. The 1 is added in order to avoid zero-value entries in the Hessian matrix.
- **Cubic** $K(u, v) = (u^T v + 1)^3$. The 1 is added in order to avoid zero-value entries in the Hessian matrix.

Include missing
specifies whether missing values are included in the model.

Penalty value
specifies the penalty value. The penalty value balances model complexity and training error. A larger penalty value creates a more robust model at the risk of overfitting the training data.

Tolerance value
specifies a custom tolerance value for model training. The tolerance value balances the number of support vectors and model accuracy. A tolerance value that is too large creates too few support vectors, and a value that is too small overfits the training data.

Maximum iterations
specifies a custom number of iterations for model training.

Standardize measure predictors
specifies whether interval variables are scaled.

Number of bins (5–100)
specifies the number of bins that the response variable is binned into when you do not use the default number of bins.

Prediction cutoff
specifies the value at which a computed probability is considered an event.

Tolerance
specifies the tolerance value that is used to determine the convergence of the iterative algorithm that estimates the percentiles. Specify a smaller value to increase the algorithmic precision.

Support Vector Machine Model Display Options

The following display properties are available for the support vector machine:

Plot layout
specifies how the results windows are displayed on the canvas. **Fit** aligns all of the objects on the canvas automatically. **Stack** displays the objects as if they are in a slide deck. Only one object is displayed at a time. When **Stack** is specified, a control bar lets you move between objects.

Statistic to show
specifies which assessment statistic to display in the model.

Plot to show
specifies which assessment statistic is plotted.

Y axis
specifies which statistic is plotted in the Lift plot.

Support Vector Machine Results

Relative Importance

The Relative Importance plots the importance value of each input variable. The variables are ranked using their first-split log worth when applied to the scored training data. The relative importance plot can be empty if no inputs are determined to be important.

Assessment Plot

For a category response, the Assessment plot displays either the Lift, ROC, or Misclassification rate.

Lift is the ratio of the percent of captured responses within each percentile bin to the average percent of responses for the model. Similarly, *cumulative lift* is calculated by using all of the data up to and including the current percentile bin.

A receiver operating characteristic (ROC) chart displays the ability of a model to avoid false positive and false negative classifications. A false positive classification means that an observation has been identified as an event when it is actually a nonevent (also referred to as a Type I error). A false negative classification means that an observation has been identified as a nonevent when it is actually an event (also referred to as a Type II error).

The *specificity* of a model is the true negative rate. To derive the false positive rate, subtract the specificity from 1. The false positive rate, labeled **1 – Specificity**, is the X axis of the ROC chart. The *sensitivity* of a model is the true positive rate. This is the Y axis of the ROC chart. Therefore, the ROC chart plots how the true positive rate changes as the false positive rate changes.

A good ROC chart has a very steep initial slope and levels off quickly. That is, for each misclassification of an observation, significantly more observations are correctly classified. For a perfect model, one with no false positives and no false negatives, the ROC chart would start at (0,0), continue vertically to (0,1), and then horizontally to (1,1). In this instance, the model would correctly classify every observation before a single misclassification could occur.

The ROC chart includes two lines to help you interpret the ROC chart. The first line is a baseline model that has a slope of 1. This line mimics a model that correctly classifies observations at the same rate it incorrectly classifies them. An ideal ROC chart maximizes the distance between the baseline model and the ROC chart. A model that classifies more observations incorrectly than correctly would fall below the baseline model. The second line is a vertical line at the false positive rate where the difference between the Kolmogorov-Smirnov values for the ROC chart and baseline models is maximized.

The misclassification plot displays how many observations were correctly and incorrectly classified for each value of the response variable. When the response variable is not binary, the logistic regression model considers all levels that are not events as equal. A significant number of misclassifications could indicate that your model does not fit the data.

Details Table

When you click in the model toolbar in the upper right of the canvas, the details table is displayed at the bottom of the canvas. The details table contains the following information:

Model Information
A brief description of the settings used to create the model.

Iteration History
The complementarity and feasibility statistics for each iteration.

Training
A brief description of the components of the model.

Fit Statistics
Provides the values of various fit statistics computed for the model.

Lift
The binned assessment results that are used to generate the Lift plot.

Misclassification
A summary of the correct and incorrect classifications for the model.

ROC
The results that are used to generate the ROC plot.

Assessment Statistics
The value of any assessment statistics computed for the model.