

Leveraging Ensemble Models in SAS® Enterprise Miner™

Miguel Maldonado, Jared Dean, Wendy Czika, and Susan Haller

SAS Institute Inc.

ABSTRACT

Ensemble models combine two or more models to enable a more robust prediction, classification, or variable selection. This paper describes three types of ensemble models: boosting, bagging, and model averaging. It discusses go-to methods, such as gradient boosting and random forest, and newer methods, such as rotational forest and fuzzy clustering. The examples section presents a quick setup that enables you to take fullest advantage of the ensemble capabilities of SAS® Enterprise Miner™ by using existing nodes, Start Groups and End Groups nodes, and custom coding.

INTRODUCTION

Ensemble models are the product of training several similar models and combining their results in order to improve accuracy, reduce bias, reduce variance, and provide robust models in the presence of new data. The first section of this paper presents a summary and examples of the most common decision tree ensemble models. The second section illustrates the methods available in the Ensemble node for combining other models. The final section incorporates newer methods by using custom coding and available functionality in SAS Enterprise Miner.

The data set that is used in the ensemble models for this paper summarizes a handwritten digit as 784 interval input variables (**Pixel0 to Pixel783**), each with a value in the range of 0 to 255. Figure 1 plots all nonzero input variables for a given observation from the original data.

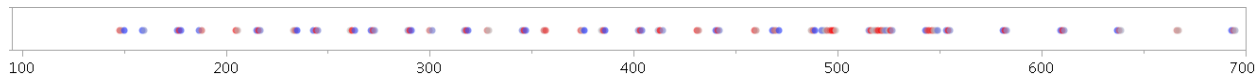


Figure 1. Visualization of an Observation in This Data Set That Represents a Handwritten 4 Digit

To provide a better understanding of what the pixel value of each input variable represents, Figure 2 rearranges all nonzero values into a 28 × 28 scatter plot for the same observation as in Figure 1.

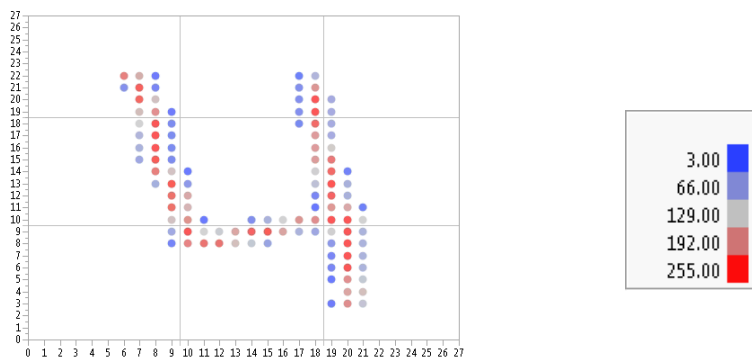


Figure 2. Scatter Plot Rearrangement of the Observation in Figure 1

The following examples introduce you to the elements and advantages of the most common ensemble models.

TREE-BASED ENSEMBLE MODELS

Decision tree models have several advantages, including that they are easy to explain and can handle missing values. Disadvantages include their instability in the presence of different training data and the challenge of selecting the optimal size for a pruned tree. Two ensemble models that were created to address these problems are bagging and boosting.

This section introduces bagging and boosting by using explanatory diagrams. Although it is unlikely that you would build these diagram flows in practice, they help you understand the more complex models that are presented later in this paper.

Bagging Diagram Flow

Bagging stands for *bootstrap aggregating*. It consists of creating several samples to train models in parallel and combining the predicted probabilities.

One way to do bagging is to build a diagram as in Figure 3 to train multiple decision trees by using different samples of the training data. You can build this diagram by including all nodes as the default, except for the random seeds of the sample nodes. You specify a different random seed for each of the sample nodes, connect a decision tree, and then use a default Ensemble node to average the predicted probabilities of all connected models. An advantage of this averaging ensemble method is that it smooths the linear cut points of a decision tree, making your model more robust in handling new data.

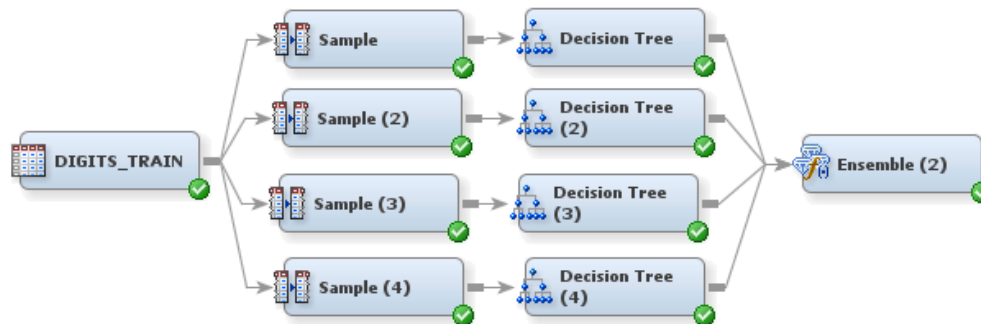


Figure 3. Diagram Flow for a Bagging Ensemble Model

Boosting Diagram Flow

One of the biggest challenges in creating decision trees is selecting the optimal size of the tree, particularly when data are scarce or target events are rare. The idea of hypothesis boosting, or simply *boosting*, was proposed as a sequential iteration of several decision trees, without the need to prune them. This algorithm was formulated under the premise that trees do not need to be very strong predictors, because misclassified observations are used to train models over iterations and the models learn to catch misclassified observations.

The diagram flow in Figure 4 illustrates an approximation of the sequential boosted bootstrap aggregating algorithm. Each iteration assigns a larger weight to observations that were not correctly classified by the previous step by incorporating the following SAS code. Notice that the frequency of observations that are not correctly classified is increased by the same factor at each step. You can try different values for the new frequency, or you can choose a SAS Enterprise Miner boosting node, like the Gradient Boosting node or Start Groups and End Groups nodes, to have these weights calculated for you so that they minimize a loss function. These Enterprise Miner nodes are presented in the next sections.

In the SAS Code node of each step in Figure 4, include the following:

```
data &EM_EXPORT_TRAIN;
  set &EM_IMPORT_DATA;
  if _freq_ = . then _freq_ = 1;
  if f_%EM_TARGET ne i_%EM_TARGET then _freq_ = _freq_ * 30;
run;
```

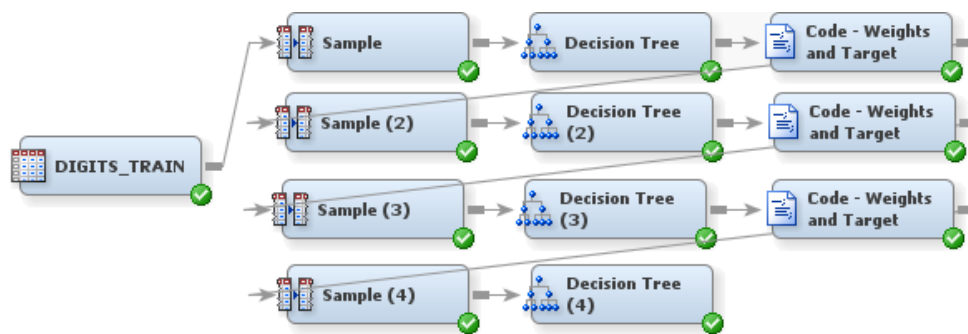


Figure 4. Diagram Flow for a Boosting Ensemble Model

Bagging and Boosting with Start Groups and End Groups Nodes

To take advantage of the bagging and boosting capabilities of SAS Enterprise Miner, build the diagram shown in Figure 5 and specify the Model property as either Bagging or Boosting in the Start Groups nodes. You can also use the index count property in the Start Groups node to specify the number of parallel trees for a bagging ensemble and the number of iterations for a boosting ensemble. After running this flow, notice that the misclassification rate is fairly low for both models. You combine these two models into a slightly better ensemble in the section [“MODEL AVERAGING ENSEMBLES.”](#)



Model	Data	Misclassification Rate
Bagging	Train	0.185
Boosting	Train	0.161

Figure 5. Diagram Flow and Misclassification Comparison of Bagging and Boosting Ensemble Models

The main differences between these models is that the bagging model samples with replacement, whereas the boosting model creates samples without replacement at each iteration. Also, boosting readjusts the weight for misclassified observations at each step, giving observations that are often misclassified a much higher weight than observations correctly classified. This weight readjustment is done as long as it minimizes a loss function.

The bagging and boosting methods can easily be applied using the Start Groups and End Groups nodes and are not limited only to decision trees. Several authors, including Breiman (1996), assert that a stepwise logistic regression or a support vector machine would perform similarly to a decision tree in a bagging or boosting ensemble. It is possible to use bagging and boosting with most predictive modeling nodes in SAS Enterprise Miner.

HPForest Node

The HPForest node creates predictive models by using a random forest ensemble methodology. It is similar to bagging in that it trains many decision trees by using different samples and then combines the predictions by averaging the posterior probabilities for interval targets or by voting for class targets. Two additional characteristics are that the HPForest node calculates the association between inputs and the target at each split of a decision tree, and it can also train trees in a different processor when running on the grid. Instead of considering every variable for splitting, the HPForest node selects a subset of input variables, calculates the association of each input variable to

the target variable, and then finds the best splitting rule on the input variable that has the highest degree of association with the target. The HPForest node is one of the SAS High-Performance Data Mining nodes: it is capable of using multiple cores in single-machine mode on your SAS server and can also run on multiple machines in distributed mode on a high-performance analytics appliance. Because of this parallelism, the HPForest node builds and evaluates many tree models simultaneously, considerably reducing the time needed for analysis.

Figure 6 shows that the misclassification rate for this same data set is considerably lower than that of the previous diagram flow.

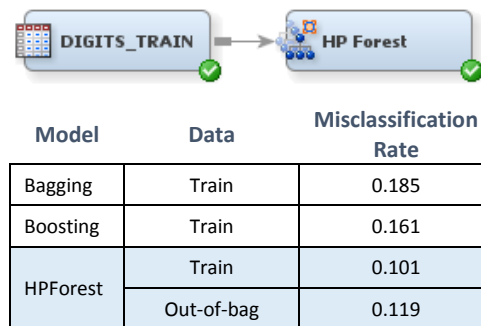


Figure 6. Diagram Flow and Misclassification Rate for HPForest Node

The calculation of out-of-bag (OOB) statistics is very characteristic of a forest model. Every time the HPForest node samples observations for a subset of training data, the observations that are not part of the sample are set aside in a holdout sample called the out-of-bag sample. The fit statistics of the out-of-bag data are obtained by averaging the fit statistics of all out-of-bag samples. This measure is expected to be higher than the same model performance measure for all the data, and it is considered a better measure of how the model is expected to perform using new data.

You can use the misclassification rate iteration plot in Figure 7 to assess your model. For this particular data set, notice that after 40 trees, the out-of-bag misclassification rate stays fairly steady at around 0.12. This means that the default of 50 trees for the HPForest node is suitable in this case. To interpret this plot, consider the misclassification rate for all data to be a pessimistic value, the out-of-bag statistic to be an optimistic value, and the misclassification rate for this model to be a value between the two of them. This last value is considerably lower than the misclassification rate for bagging and boosting. A considerable advantage of this HPForest model is that it takes the equivalent amount of time to run a 50-tree forest as it takes to run a 10-iteration bagging or boosting.

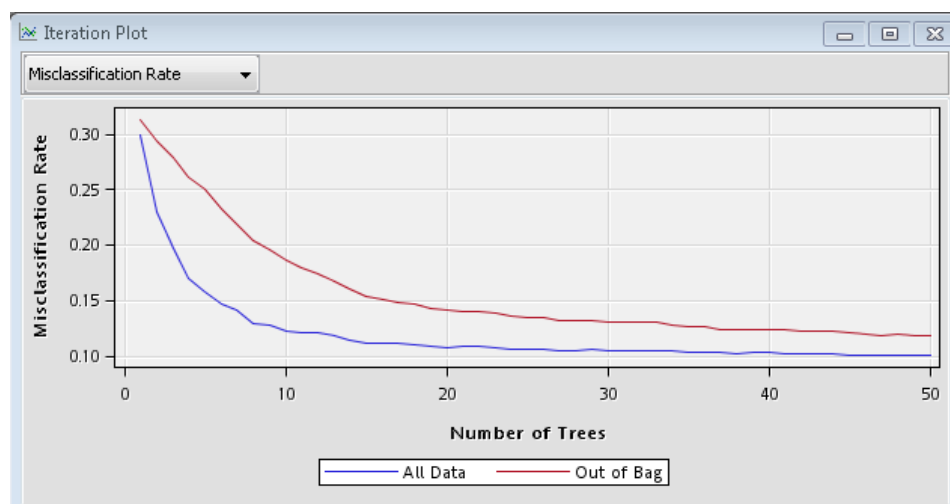
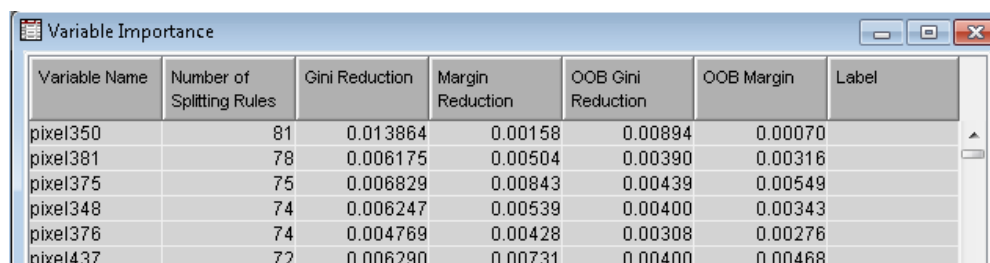


Figure 7. HPForest Node Results: Misclassification Rate for All Data and for Out-of-Bag Sample

The HPForest node also computes the variable importance of each input, based on margin reduction. The margin of an observation is defined as the probability of the true class minus the maximum probability of the other classes. The margin contribution of a variable is the weighted average of the margin across all observations. When variable selection is turned on, the variables with an out-of-bag margin reduction less than or equal to zero are rejected.

A partial output of the variable importance table is shown in Figure 8.



Variable Name	Number of Splitting Rules	Gini Reduction	Margin Reduction	OOB Gini Reduction	OOB Margin	Label
pixel350	81	0.013864	0.00158	0.00894	0.00070	
pixel381	78	0.006175	0.00504	0.00390	0.00316	
pixel375	75	0.006829	0.00843	0.00439	0.00549	
pixel348	74	0.006247	0.00539	0.00400	0.00343	
pixel376	74	0.004769	0.00428	0.00308	0.00276	
pixel437	72	0.006290	0.00731	0.00400	0.00468	

Figure 8. HPForest Node Results: Partial Output of Variable Importance Table

Gradient Boosting Node

The Gradient Boosting node runs a stochastic gradient boosting that is very similar to standard boosting, with the additional characteristics that on each new iteration the target is the residual of the previous decision tree model and that the training data are a sample of the data that were not classified correctly in the previous decision tree model.

Figure 9 shows that the misclassification rate is almost as low as that of the HPForest model.



Model	Data	Misclassification Rate
Bagging	Train	0.185
Boosting	Train	0.161
HPForest	Train	0.101
	Out-of-Bag	0.119
Gradient Boosting	Train	0.107

Figure 9. Diagram Flow and Misclassification Rate for Gradient Boosting Node

By default, the Gradient Boosting node runs for 50 iterations. You can use the iteration plot from the results, which is similar to the one in Figure 10, to assess whether you should increase the number of iterations. For this case, at the default of 50, it seems that increasing the number of iterations can lower the misclassification rate. After you increase the number of iterations to 75 and rerun the node, you get the plot shown in Figure 10 and a misclassification rate similar to the one for the HPForest node.

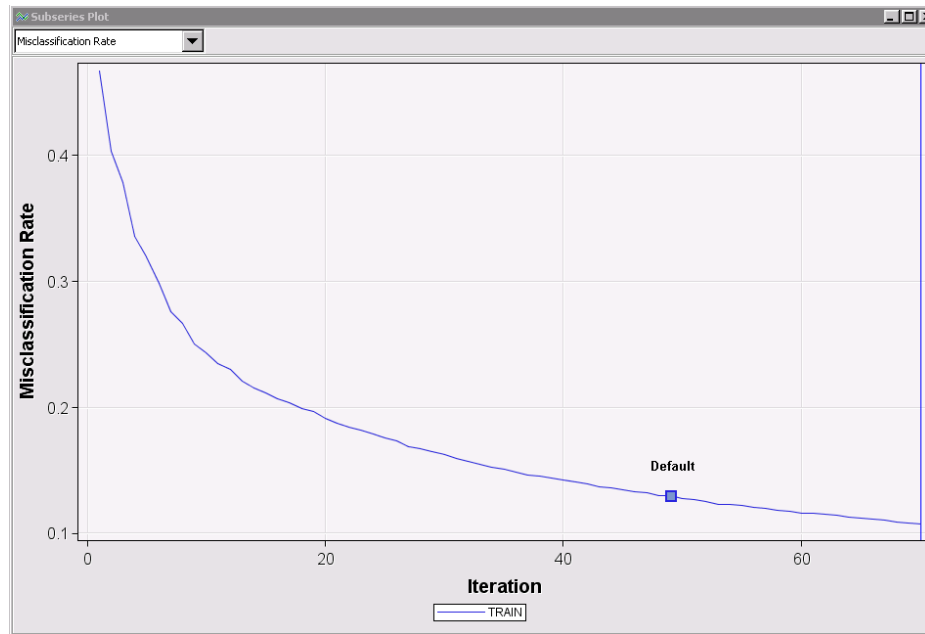


Figure 10. Subtree Plot for Gradient Boosting Node Misclassification Rate

The Gradient Boosting node has two options to assess your variables. It calculates variable importance as the change in the sum of square error that results from all the nodes in which the variable is found. It also calculates the H-statistic that is detailed in Friedman and Popescu (2005), which is a measure that quantifies variable interaction on a scale from 0 to 1. You can turn off this second option to reduce the run time of the Gradient Boosting node.

In general, gradient boosting ensembles are considered good classifiers. It is an advantage that the Gradient Boosting node computes variable interaction and variable importance. However, the sequential nature of this algorithm makes this node run more slowly than the HPForest node in most cases.

MODEL AVERAGING ENSEMBLES

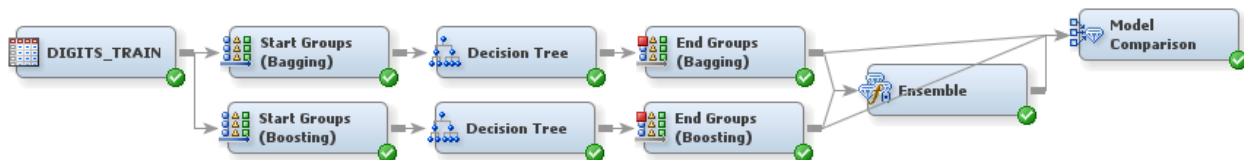
By default, the Ensemble node averages the predicted probabilities of the preceding modeling nodes. The main motivation for using ensembles is that two or more models that are accurate and discordant have better accuracy when their posterior probabilities are averaged.

In this paper, all the examples use the Ensemble node with the default of model averaging as the function to combine trained models. The other two options for combining the posterior probabilities are maximum and voting. The maximum option sets the posterior probability as the highest posterior probability of the trained models. The voting option for class variables can be done in two ways. The first one is to average the posterior probabilities of the most popular class and ignore all other posterior probabilities. The second one is to recalculate the posterior probability of each class by using the proportion of posterior probabilities that predict that class.

The first example combines the two models from the section [“TREE-BASED ENSEMBLE MODELS”](#) to introduce the concept of discordant models and to illustrate how an ensemble that combines two adequate models can improve the model fit. The second example describes an analytics approach to test whether you should perform ensemble modeling or subsequent modeling to get a better model.

Discordant Models Ensemble Flow

You can use the Ensemble node to combine the posterior probabilities of many model nodes. As an example of an ensemble made from discordant models that improves accuracy, consider the diagram flow in Figure 11, which is the same diagram as in Figure 5 but with bagging and boosting flows connected to an Ensemble node. Run this flow, and you notice that the ensemble model created from a bagging model and a boosting model has a slightly better misclassification rate than either the bagging or boosting model.



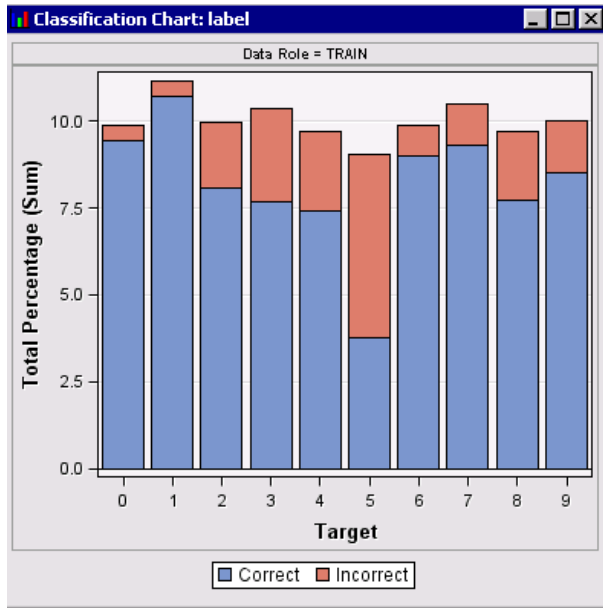
Model	Data	Misclassification Rate
Bagging	Train	0.185
Boosting	Train	0.161
HPForest	Train	0.101
	Out-of-bag	0.119
Gradient Boosting	Train	0.107
Ensemble	Train	0.151

Figure 11. Diagram Flow and Misclassification Comparison for Bagging, Boosting, and the Ensemble of Both

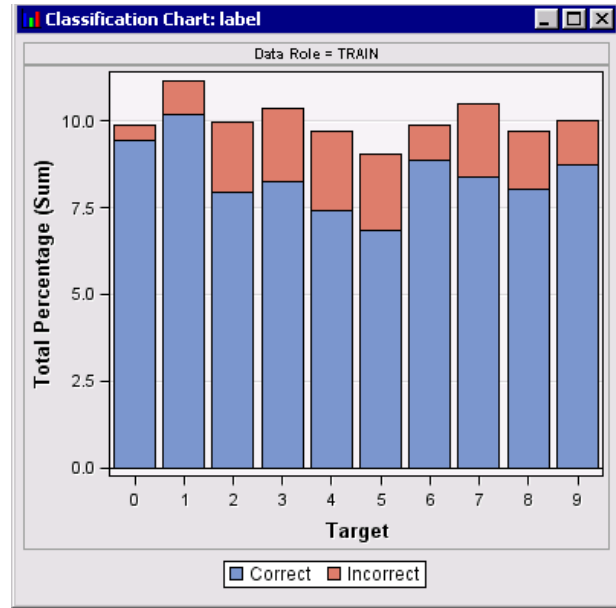
To visualize discordance and the improvement in misclassification rate, look at Figure 12. These charts show the classification for bagging, boosting, and the ensemble of both models. You can use these charts to supervise which level of the target is being correctly predicted and to ensure that your ensemble is improving the proportion of the target correctly predicted across all levels. Notice that the boosting model has a higher percentage of handwritten 5s and 8s correctly classified, whereas the bagging model has slightly more 1s and 7s correctly classified. The ensemble of both models shows better classification of all digits overall.

In general, ensembles result in a more accurate model when each of the component models in the ensemble has good assessment statistics but the component models differ in their predicted probabilities.

a. Bagging



b. Boosting



c. Ensemble of bagging and boosting

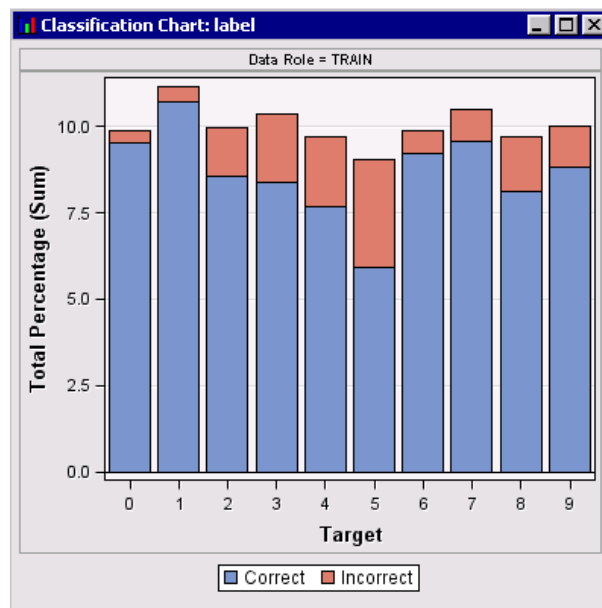


Figure 12. Classification Charts for (a) Bagging, (b) Boosting, and (c) the Ensemble of Both Models

Fit Statistics Improvement through Ensemble Comparison

The examples that have been presented so far have been useful for introducing different types of ensemble models and for illustrating how combining posterior probabilities can lead to improved results. Now you can use an analytics approach to confirm whether tweaking a model or creating an ensemble from several models provides you with a better model.

Neural networks have been widely used as classifiers of handwritten digits. You can run several neural network models and compare their fit statistics with those of several ensembles to determine whether an ensemble or a more

complex neural network gives you better results. The diagram flow in Figure 13 includes four neural networks that have 3, 10, 30, and 50 hidden units. Then you add several ensembles. Because you expect neural networks that contain more hidden units to have a lower misclassification rate, you do not need to create an ensemble of all possible combinations. For this data set, it is suitable to have ensembles that omit the neural network models one by one. In other words, the first Ensemble node combines the predicted probabilities of all neural network models, the second one combines the neural networks of 10 to 50 hidden units, and the last one combines the neural networks of 30 or more hidden units.

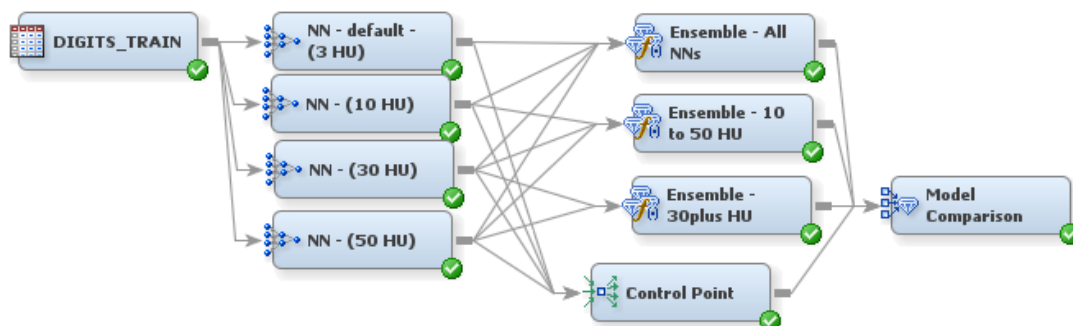


Figure 13. Suggested Approach to Test the Leverage of an Ensemble Model for This Data Set

After you run this flow, you can summarize the results as in Figure 14. Notice from the misclassification graph that increasing the number of hidden units decreases the misclassification rate as expected, up to a certain point. After 30 hidden units, it does not seem that you can get a lower misclassification rate by increasing the number of hidden units. From the table in Figure 14, you can see that the ensemble of the two largest neural network models reduces the misclassification rate to a considerably lower value than any of the neural network models alone can achieve.

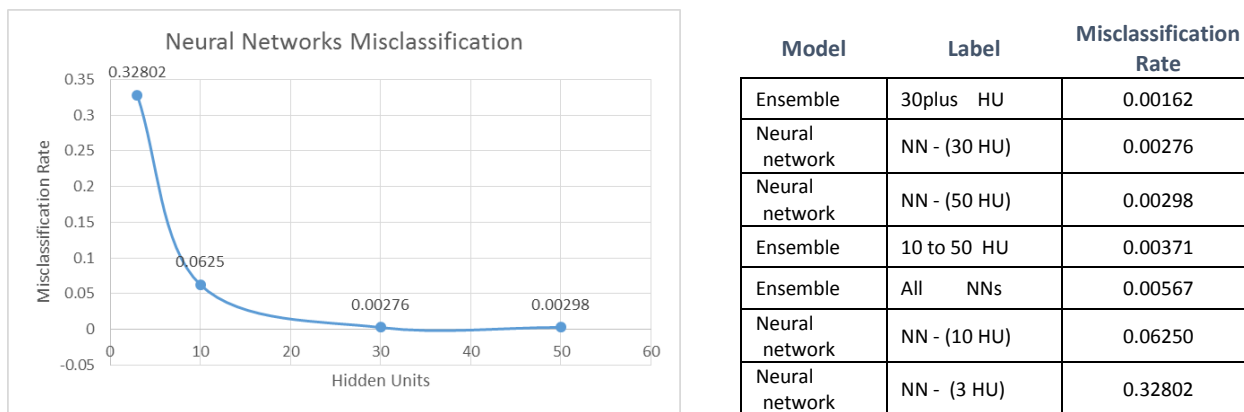


Figure 14. Misclassification for Neural Networks and Ensemble Models

This approach is a very fast way to test whether an ensemble improves fit statistics. Adding extra ensembles to your diagram flow does not significantly increase the run time and can give you good insights into what step to take next—whether to keep tweaking your model or to use an ensemble of some of your models instead. For this particular data set, it seems appropriate to select the ensemble model labeled 30+ *hidden units*.

CUSTOM CODE ENSEMBLE MODELING

The flexibility of SAS Enterprise Miner enables you to incorporate new modeling methods by using either existing nodes or your own code. The first example in this section is a rotation forest (Rodríguez, Kuncheva, and Alonso 2006), which is a probability averaging method. The second example is a fuzzy cluster (Wedding 2009), which, instead of combining the posterior probabilities, assigns shared membership to two or more different class levels at the same time.

Rotation Forest

To approximate the original rotation forest algorithm in Rodríguez, Kuncheva, and Alonso (2006), you can create a diagram flow like the one in Figure 15. Notice that the diagram is very similar to a bagging flow, with two additional characteristics. The first characteristic is that not all variables are used in each classifier model subflow; instead, a SAS Code node passes a random subset of the input variables and rejects the rest. The second characteristic is that a Principal Component node passes only the principal components as inputs instead of the original variables. These two characteristics enable the decision trees in this flow to use the principal components of the subset of inputs to obtain a posterior probability for each level. Then an Ensemble node averages the posterior probabilities to provide a classification for each observation.

The misclassification rate from this ensemble suggests that the rotation forest method is not suitable for this data set. You can use a similar flow if you want to use random subsets of variables in a parallel tree ensemble.

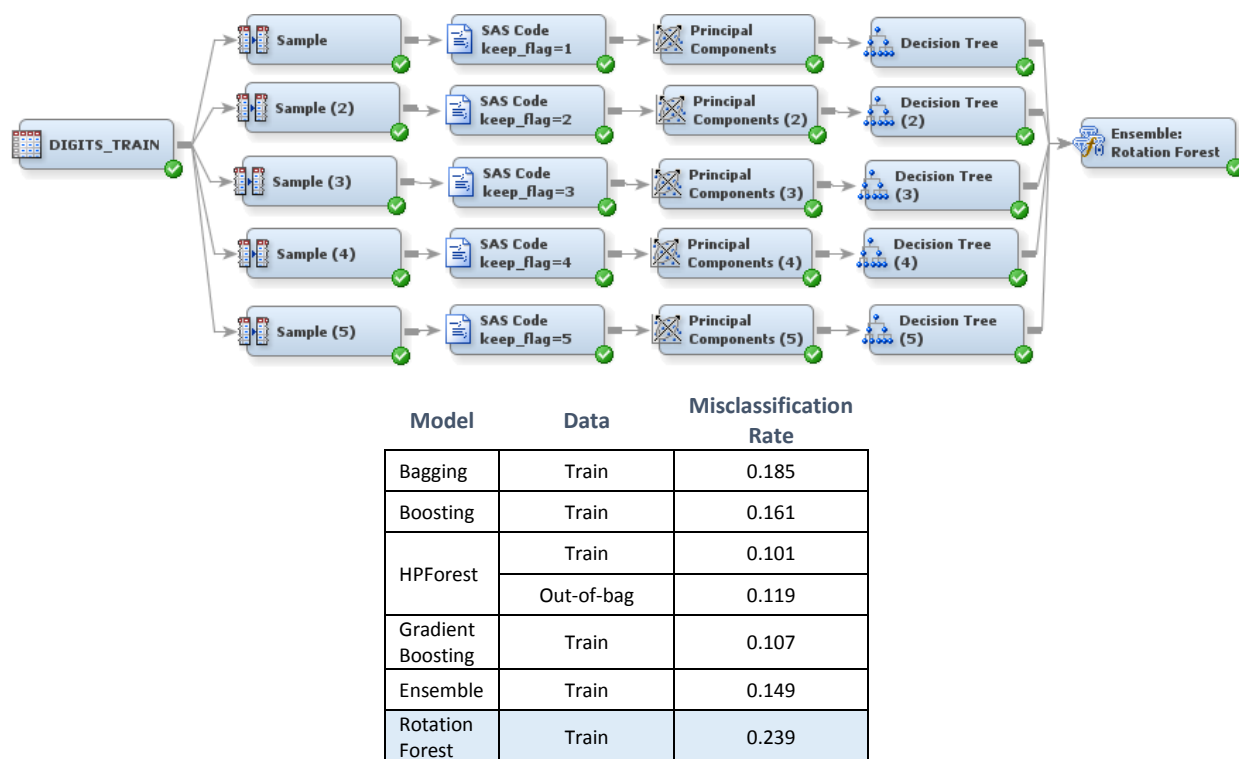


Figure 15. Diagram Flow and Misclassification Rate for Rotation Forest Diagram Flow

The following code stores each of your input variables in a macro variable that has the prefix *my_input*. A random variable *x* is created, and its value is compared to a **keep_flag** value. If the value of the random variable is different from the **keep_flag** value, the input variable is rejected. Every time the SAS Enterprise Miner macro %EM_Metachange rejects a variable, a message is output to the log. You can specify the number of groups that you are using in your ensemble.

The diagram flow in Figure 15 has five SAS Code nodes that contain the following code. The only difference is that you set a different **keep_flag** value in the macro call.

```

%macro keepsomevars(randomseed=12345,n_groups=5,keep_flag=1);
data _null_;
set &EM_IMPORT_DATA_CMETA(where=(role="INPUT")) end=eof;
call symput ('my_input'!!strip(put(_N_, BEST.)),strip(NAME));
if eof then call symput('total_inputs', strip(put(_N_, BEST.)));
run;

%do i = 1 %to &total_inputs;
  %let x =%sysevalf(%sysfunc(ceil(%sysfunc(ranuni(&randomseed))*&n_groups)));
  %put variable myinput&i is &&my_input&i with random value &x;
  %if "&x" ne "&keep_flag" %then %do;
    %put Variable &&my_input&i will be rejected.;
    %EM_METACHANGE(name=&&my_input&i, role=REJECTED);
  %end;
%end;
%mend keepsomevars;

%keepsomevars(randomseed=12345,n_groups=5,keep_flag=1)

```

Fuzzy Clustering as an Unsupervised Ensemble

Decision trees and clustering are the two most common data mining techniques for creating groups or segments. Whereas decision tree algorithms require a target variable, unsupervised clustering algorithms rely only on the input variables to create segments whose observations are as similar as possible to each other and as different as possible from observations in other segments. This section presents a case in which unsupervised modeling is necessary and fuzzy clustering improves the usability of cluster segments.

James Bezdek (1981) introduced fuzzy clustering to address the difficulty in geostatistical analysis of assigning observations to a specific geographical area when the observations are close to the boundary between two regions. He formulated the mathematics to assign an observation to two or more clusters simultaneously by calculating a fuzzy cluster membership. An advantage of fuzzy cluster membership is that you can interpret it as a measure of how strongly (or weakly) an observation belongs to a certain cluster. In the example presented in this section, instead of combining posterior probabilities of two models to build an ensemble, you use the highest and second-highest fuzzy memberships to build rules that add information to your cluster analysis.

To turn the analysis of the digits data set into an unsupervised-learning exercise, pretend that you do not know the target variable for these observations beforehand. In this scenario, you need to prioritize which observations must be visually inspected by a group of analysts to assign the target variable manually. You can do this by creating 10 cluster segments and using the fuzzy membership to determine whether an observation belongs to a cluster because it is very different from other segments or because it is just barely different and is as likely to belong to a different cluster.

Figure 16 summarizes the results from a Cluster node with no internal standardization and 10 as the maximum number of clusters. From the pie chart, you can see that your 10 clusters have a fairly similar size, potentially making your model a good classifier. From the table in Figure 16, you can see three segments that can be mapped to an actual digit. Cluster segment 7 captures most of the 1s, segment 10 most of the 2s, and segment 8 most of the 3s. Some of the segments do a better job than others. Cluster segment 10 captures most of the 2s and not many of the other digits, whereas segment 7 captures almost every 1 digit but also many of the other digits. In the next step of this analysis, you use fuzzy membership to quantify how likely an observation is to belong to each cluster.

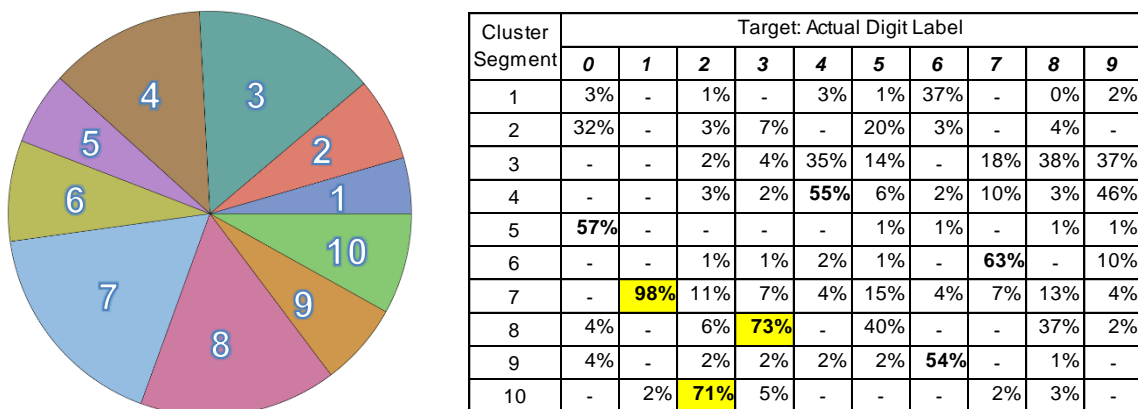


Figure 16. Cluster Node Results: Segment Size Pie Chart and Target per Segment Table

You can use fuzzy clustering as a metric to assess the cluster segmentation. Observations with high membership in just one cluster are more likely to be correctly classified, and observations with low relative membership in two or more clusters can be sent for visual inspection by the analysts.

As a first step in fuzzy clustering, run a Cluster node and connect a SAS Code node, as in Figure 17. Then follow the steps after the figure to use a SAS Code node to calculate the fuzzy membership for all observations to each of the 10 clusters you found using the Cluster node. The code incorporates a macro developed by Wedding (2009) to approximate Bezdek's method of fuzzy clustering.

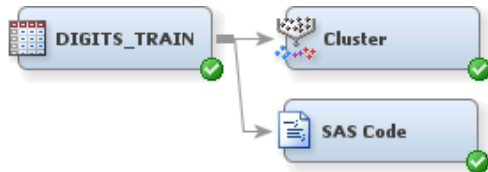


Figure 17. Diagram Flow with Necessary Elements to Do Fuzzy Clustering with a SAS Macro

In the SAS Code node training code, paste the following:

1. Initialization of the fuzzy clustering SAS macro from the Appendix
2. DATA and SET statements with the corresponding export and import macro variables:


```
data &EM_EXPORT_TRAIN;
  set &EM_IMPORT_DATA;
```
3. The score code from the Cluster node, available from the **Results** menu (**View -> Scoring -> SAS Code**)
4. A call to the fuzzy clustering SAS macro, where you specify the following:
 - a. DArray – Prefix used in the cluster score code to call the cluster distances. In the example, you use *Clusvads*, because this is the internal name that the Cluster node gives to cluster distances. If you have more than one Cluster node, the prefixes are *Clus2vads*, *Clus3vads*, and so on.
 - b. UArray – Prefix that you want to give to the output fuzzy cluster membership. The example uses **Ufuzzy** to denote fuzzy membership, but you can choose your own.
 - c. Size – The number of fuzzy memberships that you want. In this case, you want observations to be simultaneously part of the 10 clusters that you calculated so that you can use them as an indicator that the observation belongs in that cluster.
 - d. *m* – Sometimes referred to as fuzzy factor, it is a constant that determines how different the cluster membership (u_k) will be from the hard cluster—the type of cluster in which an observation belongs to only one cluster. The recommended range is m (1, 3], and a rule of thumb is to use $m = 2$. A value close to $m = 1$ will reproduce the results of the Cluster node.

$$u_k = \frac{1}{\sum_{i=1}^j \left(\frac{d_k}{d_i} \right)^{\frac{2}{m-1}}}$$

5. A RUN statement to close the DATA step

An extract of the SAS Code node training code for this example follows. It includes explanatory comments in italics to indicate where lines of code were omitted and the beginning of the score code that you copied from the Cluster node results.

```
%macro fuzzyCluster( DARRAY, UARRAY, SIZE, M=2 );
* ... omitted lines ... *;
%mend fuzzyCluster;

data &EM_EXPORT_TRAIN;
  set &EM_IMPORT_DATA;

* ... Score code copied from Cluster node results ... *;
*****;
*** Begin Scoring Code from PROC DMVQ ***;
*****;
```

```

* ... omitted lines ... *;

*****;
*** End Scoring Code from PROC DMVQ ***;
*****;
length _SEGMENT_LABEL_ $80;
label _SEGMENT_LABEL_='Segment Description';
if _SEGMENT_ = 1 then _SEGMENT_LABEL_="Cluster1";

* ... omitted lines ... *;

if _SEGMENT_ = 10 then _SEGMENT_LABEL_="Cluster10";

%fuzzyCluster(CLUSvads,Ufuzzy,SIZE=10,M=2);

run;

```

The output from the SAS Code node gives you 10 new variables for fuzzy membership (**Ufuzzy1** to **Ufuzzy10**). It is hard to interpret an observation being assigned simultaneously to 10 clusters. Instead, use fuzzy memberships as an indicator that an observation belongs in a certain cluster. Figure 18a shows a density graph for the two highest fuzzy memberships of each observation. Figure 18b is the same plot but grouped by the target (digit label).

You can use these two plots to build rules that prioritize what observations must be sent for visual inspection. You can draw three rules from Figure 18a:

- There is a region whose highest membership is less than or equal to 0.14 and whose second-highest membership is less than or equal to 0.14. Because these memberships are about the same, all the observations in this region should be sent for visual inspection.
- Area D has a region whose highest membership is greater than or equal to 0.22 and whose second-highest membership is 0.12 or less. Observations in this region have a stronger membership in their cluster segment, suggesting that they are likely to be correctly classified. You can send a small sample of these observations for visual inspection, just to confirm that they map their cluster segment well. Figure 18b shows that most of these observations map the 1 digit.
- Area A is the region that has a high concentration of observations whose highest fuzzy membership is too close to their second-highest fuzzy membership. For this reason, you might want to send most of these observations for visual inspection.

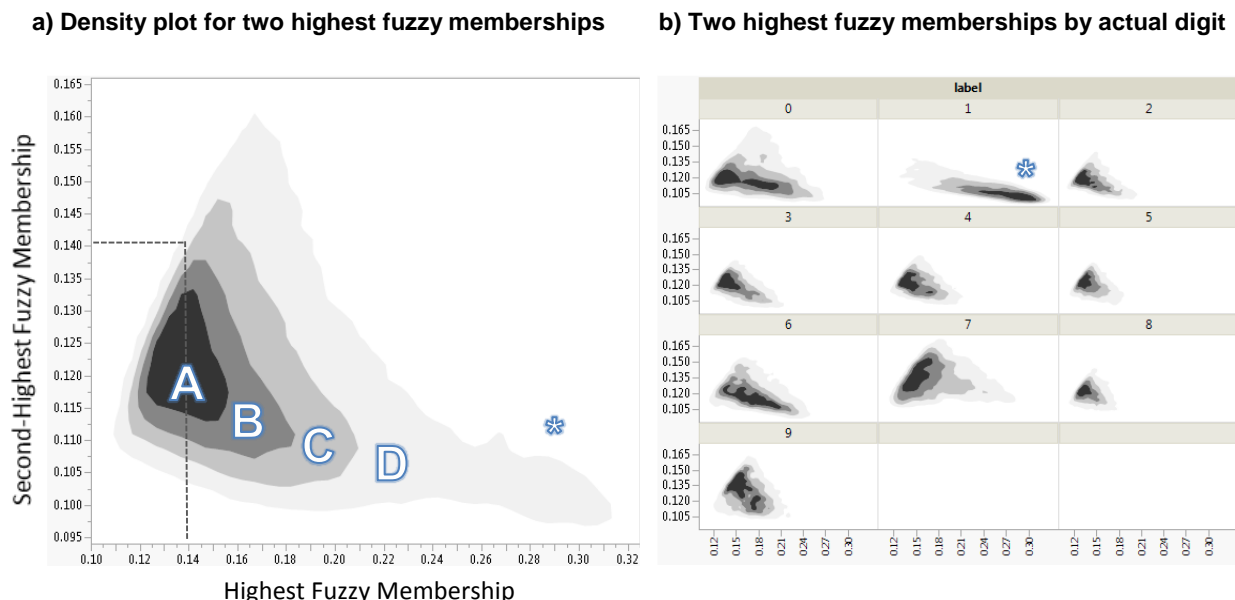


Figure 18. Density Plots of Two Highest Fuzzy Memberships for All Data and for Each Digit in the Data Set

From this example, you can see that fuzzy cluster membership can complement your cluster segment analysis by providing a measure to assess whether an observation is correctly classified.

CONCLUSION

It is recommended that you compare two or more models in a diagram flow with their ensemble model to test for possible model discordance, to reduce bias, and to try to improve the fit statistics. In the particular case of decision trees and stepwise regressions, you can easily add these models into bagging and boosting ensembles by using the Start Groups and End Groups nodes in SAS Enterprise Miner. You can compare decision tree models to specific ensemble model nodes like the Gradient Boosting node and HPForest node. You can combine all other common modeling nodes into an ensemble by using the Ensemble node.

In addition to using SAS Enterprise Miner ensemble functionality to build predictive models, you can use specific nodes to learn more about your data. The HPForest and Gradient Boosting nodes enable you to calculate variable importance and to do variable selection. You can also take advantage of specific metrics like the Friedman H-statistic or out-of-bag statistics.

Now that you are familiar with the theory and the advantages of the most popular ensemble models, you can use this modeling tool by adding ensembles to your diagrams and even code your own ensemble models for both supervised and unsupervised modeling. Whether your ensemble method requires you to modify the training data as in bagging and boosting, or to modify the subset of input variables as in random forests, SAS Enterprise Miner nodes have the flexibility to assist you in your modeling tasks.

APPENDIX

The following SAS macro calculates the cluster membership for fuzzy clustering.

```
%macro fuzzyCluster( DARRAY, UARRAY, SIZE, M=2 );

    array &UARRAY. &UARRAY.1-&UARRAY.&SIZE.;
    array D&UARRAY.[&SIZE.] _temporary_;
    length POWER 8.; drop POWER;
    length DISTSUM 8.; drop DISTSUM;
    length DISTFLAG 8.; drop DISTFLAG;
    length i 8.; drop i;
    length MAXD 8.; drop MAXD;

    POWER = 2/(&M.-1);
    MAXD = -1;

    do i = 1 to &SIZE.;
        D&UARRAY.[i] = sqrt(&DARRAY.[i]);
        if D&UARRAY.[i] > MAXD then MAXD = D&UARRAY.[i];
    end;

    do i = 1 to &SIZE.;
        D&UARRAY.[i] = D&UARRAY.[i] / MAXD;
    end;

    DISTSUM = 0;
    do i = 1 to &SIZE.;
        DISTSUM = DISTSUM + D&UARRAY.[i]**POWER;
    end;

    do i = 1 to &SIZE.;
        &UARRAY.[i] = 1 / ( (D&UARRAY.[i]**POWER)*DISTSUM );
    end;

    DISTFLAG = 0;
    do i = 1 to &SIZE.;
        if missing( &UARRAY.[i] ) then DISTFLAG = 1;
    end;

    if DISTFLAG > 0 then do;
        do i = 1 to &SIZE.;
            if missing( &UARRAY.[i] ) then &UARRAY.[i] = 1;
            else &UARRAY.[i] = 0;
        end;
    end;

    DISTSUM = 0;
    do i = 1 to &SIZE.;
```

```

DISTSUM = DISTSUM + &UARRAY.[i];
end;

do i = 1 to &SIZE.;
  &UARRAY.[i] = &UARRAY.[i] / DISTSUM;
  &UARRAY.[i] = round( &UARRAY.[i], 0.001 );
end;

%mend fuzzyCluster;

```

REFERENCES

- Bezdek, J. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum.
- Breiman, L. 1996. "Bagging Predictors." *Machine Learning* 24: 123–140.
- De Ville, B. and Neville, P. 2013. *Decision Trees for Analytics Using Enterprise Miner*. Cary, NC: SAS Institute Inc.
- Friedman, J. H. and Popescu, B. E. 2005. "Predictive Learning via Rule Ensembles." www-stat.stanford.edu/~jhf/ftp/RuleFit.pdf
- Rodríguez, J., Kuncheva, L., and Alonso, C. 2006. "Rotation Forest: A New Classifier Ensemble Method." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28: 1619–1630.
- Wedding, D. K. 2009. "Incorporating Fuzzy Cluster Memberships within SAS Enterprise Miner." *Proceedings of the Conference SAS M2009*.

ACKNOWLEDGMENTS

The authors would like to thank Padraic Neville, Don Wedding, Patrick Hall, and Ed Huddleston for their contributions to this paper.

RECOMMENDED READING

- *Decision Trees for Analytics Using SAS Enterprise Miner*
- *SAS Enterprise Miner Reference Help*
- *SAS Enterprise Miner Node Developer Guide*
- *Base SAS Procedures Guide*
- *Base SAS Procedures Guide: High-Performance Procedures*

CONTACT INFORMATION

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

Miguel M. Maldonado	MiguelM.Maldonado@sas.com
Jared Dean	Jared.Dean@sas.com
Wendy Czika	Wendy.Czika@sas.com
Susan Haller	Susan.Haller@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.

Other brand and product names are trademarks of their respective companies.