Paper SAS3120-2016

Ensemble Modeling: Recent Advances and Applications

Wendy Czika, Miguel Maldonado, and Ye Liu, SAS Institute Inc.

ABSTRACT

Ensemble models are a popular class of methods for combining the posterior probabilities of two or more predictive models to create a potentially more accurate model. This paper summarizes the theoretical background of recent ensemble techniques and presents examples of real-world applications. Examples of these novel ensemble techniques include weighted combinations (such as stacking or blending) of predicted probabilities in addition to averaging or voting approaches that combine the posterior probabilities by adding one model at a time. Fit statistics across several data sets are compared to highlight the advantages and disadvantages of each method, and process flow diagrams that can be used as ensemble templates for SAS® Enterprise Miner™ are presented.

INTRODUCTION

It has been well established in the literature, real-world applications, and data mining competitions—most notably the Netflix Prize, KDD Cup, and Kaggle—that creating ensemble models by combining the results from multiple predictive models can improve prediction accuracy over that of an individual model. The models that you combine can be based on diverse algorithms such as regression, decision trees, and neural networks, or on variations of the same algorithm, such as multiple decision trees that use different splitting and pruning criteria. Even simple, naïve ensembles that do not account for each model's performance have provided more accurate predictions in many cases than the most accurate single algorithm, with the most improvement over individual models seen when model predictions are uncorrelated (Seni and Elder 2010).

When you are dealing with a categorical target or response variable where classification is the goal, simple ensemble methods include voting and averaging. You can easily implement these two methods in SAS Enterprise Miner by incorporating an Ensemble node into your process flow diagram. You can combine any number of the following types of models in this way: Bayesian networks, decision trees, regression models, neural networks, and support vector machines with interior point optimization. Furthermore, by incorporating a small amount of SAS® code into your flow via a SAS Code node and by using the plethora of fit statistics generated by the Model Comparison node, you can extend these simple methods into more complex ensembles that account for each model's performance. This paper discusses several of these weighted averaging, stacking, and ensemble selection techniques, and it discusses an implementation of a clustering-based ensemble selection that ensures that the models you are combining are both representative and diverse, similar to the *k*-means clustering-based selective ensemble (Lessmann et al. 2015; Qiang, Shang-Xu, and Sheng-Ying 2005).

This paper focuses only on the combination of predictions from models that you build on the same training and validation data, not ensemble methods such as bagging and boosting that involve resampling or reweighting of the training data. Previous papers discuss how you can implement bagging and boosting of models in SAS Enterprise Miner, by using either the Start Group and End Group nodes (Schubert 2010) or multiple Sample and replicated modeling nodes along with the Ensemble node (Maldonado et al. 2014).

The paper presents two templates from which you can create good candidate models for an ensemble, as well as a third template with custom SAS code to produce and compare new ensemble models. You will become familiar with some best practices for predictive modeling and with the flexible architecture of SAS Enterprise Miner that enables you to code your own enhancements of methods or visualization of results.

ENSEMBLE METHODS

You can implement the following ensemble methods in a SAS Enterprise Miner process flow diagram for a categorical response variable, either with the Ensemble node or with SAS code in a SAS Code node

that this paper provides in an appendix. Several of these heterogeneous ensemble methods are described and compared extensively in Lessmann et al. (2015).

- **Simple averaging** (for a categorical response, this is equivalent to simple soft voting, described by Zhou 2012) takes the average of the posterior probability for each response level across the models and then classifies the models based on the level that has the maximum average probability.
- **Top-*t* ensemble selection** (Lessmann et al. 2015), for various values of *t* (such as 5, 10, and 25), takes the top *t* models out of the *M* that are generated when the models are ranked by an accuracy measure and uses validation data to determine the best value for *t*. The code for this paper evaluated at all values of *t*: 1, …, *M*. You can think about this selection method as an alternative way to assess multiple combinations of your candidate models by adding one model at a time to a simple averaging ensemble.
- **Hill-climbing ensemble selection** (Caruana et al. 2004; Lessmann et al. 2015), like top-*t* ensemble selection, starts by ranking the models by an accuracy measure. The implementation in this paper calculates the improvement in accuracy of adding any given model, and includes in the ensemble the posterior probabilities of the model that most improves the misclassification rate in the validation set. The final ensemble is selected based on the misclassification rate in the test set.
- **Weighted averaging** (weighted soft voting; Zhou 2012) calculates a weighted average of the posterior probabilities for each response level, with a model-specific weight applied. Both top-*t* ensemble selection and hill-climbing ensemble selection can be considered weighted averaging techniques; top-*t* selects a subset of the available models and assigns an equal weight to them, whereas hill-climbing has a different weight for each model depending on how many times a particular model is included in the ensemble.

  One way to create a weighted averaging ensemble is to train a linear regression that uses the output of the SAS Code node described in Appendix 1, because it sets the posterior probabilities of your candidate models as inputs. To do this, add an HP Regression node and specify the regression type as linear, optionally with the intercept suppressed. The weights of your ensemble are the parameter estimates of your regression. This assumes that your binary target has the values 0 and 1, so it might first require transforming your target in a Transform Variables node or in a SAS Code node.
- **Stacking**, also called blending (Zhou 2012), uses the posterior probabilities from the various models that are used as inputs and the original (observed) response or target variable that is used as the response. In this framework, you can use a linear regression model to generate the weights for a weighted averaging ensemble as mentioned earlier (Ting and Witten 1999); you can also implement several types of penalized linear regression (LASSO, ridge, and elastic net) for stacking, as described by Reid and Grudic (2009). Other models, such as decision trees and random forests, can also be used for stacking.
- **Clustering-based selection** uses a principal-component-based variable clustering algorithm to cluster posterior probabilities that are similar and uses simple averaging to choose the best model from each cluster to combine into an ensemble. This method is intended to help ensure that the models in your ensemble have discordant posterior probabilities. It is similar to the top-*t* ensemble selection method, with the difference that the order in which models are added is determined by using one model of each cluster at a time. One way to do this is to use the Variable Cluster node, as shown in Figure 1.

Figure 1. Diagram of Clustering-Based Selection

For example, the default Variable Clustering node finds these two clusters or types of models for several default models of the German Credit data set (available when you select **Help►Generate Data Sets** from the SAS Enterprise Miner main menu), as shown in Figure 2.

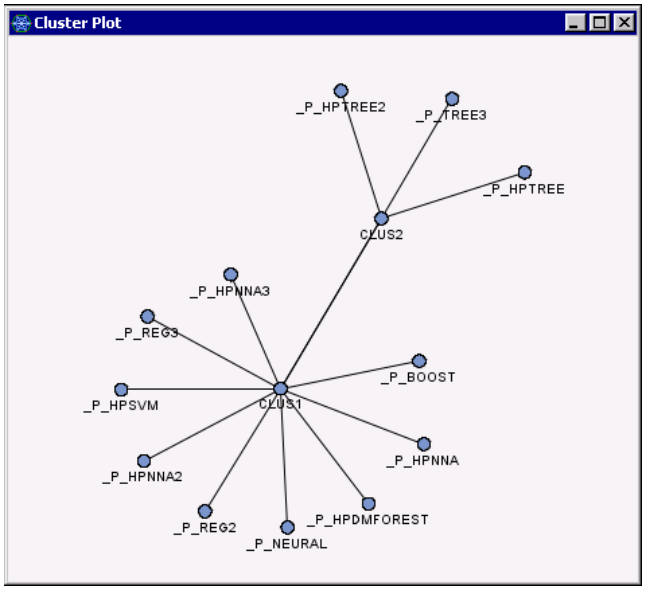


Figure 2. Output from Variable Cluster Node

An alternative to the clustering-based selection method is variable clustering on the residuals of each model, instead of the posterior probabilities.

METHODOLOGY

The main goal of this paper is to provide several code workarounds that you can use to compare the existing ensemble methods in SAS Enterprise Miner with more sophisticated methods. Figure 3 outlines a simple diagram to make this comparison. You have your data, you have a series of models that you want to compare and combine, and then you connect them to a subflow that does what is shown in Figure 3 and described in the list that follows.

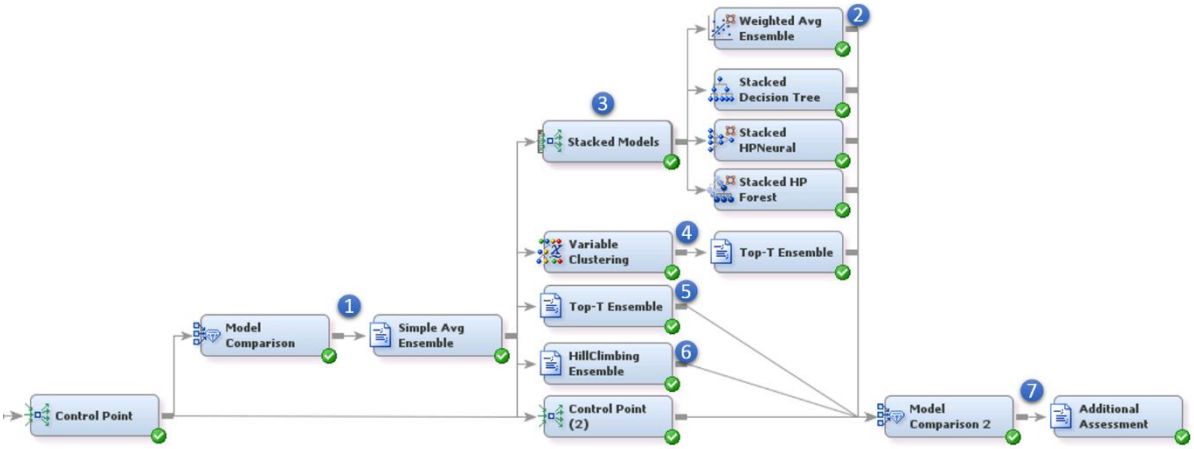


Figure 3. Diagram Flow of Ensemble Methods

1. A Model Comparison node produces a summary of the fit statistics that you need in order to use the code provided for some other ensemble methods. A SAS Code node (detailed in Appendix 1) transposes the posterior probabilities of any models connected to the Model Comparison node and calculates the simple average directly by using the posterior probabilities. This enables you to

combine any models even if they do not produce SAS DATA step score code, which means that you can include models from nodes like HP Forest, HP SVM, and MBR in your ensemble. This is not currently supported by the Ensemble node.

2. One way to create a weighted averaging ensemble is to use a linear regression to calculate the weights for the posterior probabilities of your models. You can do this by adding either an HP Regression node or a Regression node. With the HP Regression node, if you first convert your binary target to an interval target with the values 0 and 1, you can use the LASSO model selection method to perform a penalized linear regression (Reid and Grudic 2009). Alternatively, using the GLMSELECT procedure in a SAS Code node enables you to additionally fit a ridge or elastic net regression, with options for controlling the search or for using cross validation to select the best regularization parameter. To see the weights of your model, select **Results►Estimates** from the SAS Enterprise Miner main menu. In addition, to confirm the weight of your models, you can assess what models in your ensemble are statistically significant based on the *p*-value of their posterior probabilities. You can also turn selection methods on or off and compare the results.

3. To create stacking ensembles, simply connect predictive models to the output of the SAS Code node that you used to calculate a simple average ensemble. This compares a stacked Decision Tree node, an HP Neural Network node, and an HP Forest node.

4. Use a Variable Clustering node to ensure that your top-*t* ensemble combines models in an order that takes model diversity into account. The Variable Clustering node determines what models are alike, and the top-*t* ensemble SAS Code node adds one model of each cluster at a time. Appendix 2 presents the SAS macro code that you can use to produce a top-*t* ensemble.

   To produce truly unbiased results for the top-*t* and hill-climbing ensembles, you should use an additional validation hold-out sample to evaluate the best *t* or the stopping point for the hill-climbing, separate from the validation partition used in modeling. The code in this paper bases the stopping point on statistics from the test partition.

5. If your SAS code for a top-*t* ensemble does not detect a preceding Variable Clustering node, it will add your models one at a time to your ensemble based on the misclassification rate of your validation data.

6. The SAS Code node for a hill-climbing ensemble (detailed in Appendix 3) is a generalization of the code for the top-*t* ensemble, in which you can add any model to the ensemble as long as it decreases the misclassification on your validation data and a model can be added multiple times.. As with the implementation of the top-*t* ensemble, the hill-climbing code provided in this paper selects the best models for the ensemble based on the fit statistics from the testing partition.

7. Use the SAS macro code detailed in Appendix 4 to produce additional assessment plots that help you choose the best model for your business problem. In certain cases, several models have a similar metric for the ROC index or misclassification rate. These plots are intended to help you visually assess the performance of your model in more than one dimension.

NUMERICAL COMPARISON

DATA SETS

This comparison uses data sets and models similar to those in Lessmann et al. (2014). The target variable in these data sets is a good or bad flag, with a bad flag indicating payment default in a credit scoring scenario. A general description of the input variables in each data set follows:

- **Australian Credit** contains 14 unidentified variables, half of them categorical, half of them numerical.
- **German Credit** contains demographic information.

- **Home Equity** contains internal credit information ratios such as debt income, number of years at a job, number of years with a checking account, and so on.
- **Give Me Credit** was part of a Kaggle competition in 2010. It contains demographic information.
- **PAKDD** was part of the challenge in the sixth edition of the Pacific-Asian Knowledge Data Discovery. It contains demographic information about the residence and work location.

CANDIDATE MODELS

This section outlines two templates that are designed to create good candidate models for your ensemble. The first template is based on the rapid predictive modeler (SAS Institute Inc. 2010). It consists of three subflows that preprocess inputs by calculating several transformations, imputing, and performing variable selection when necessary. It then uses those modified inputs to train decision trees, regressions with different selection criteria, and a neural network. Figure 4 presents the template for this process diagram flow.

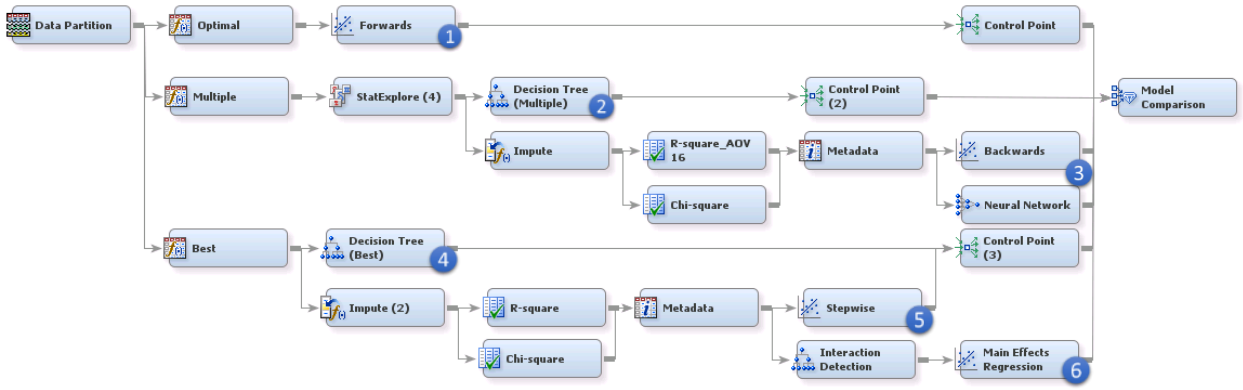


Figure 4. Diagram Subflow for RPM-Based Models

You can use these suggested RPM-based practices for predictive modeling after you create a data partition:

1. Include a Transform Variables node that keeps the optimal transformations for each input and a Regression node with forward selection turned on.
2. Include a Transform Variables node that creates multiple transformations for each input and a StatExplore node that keeps only the 500 most important inputs according to decision tree variable importance. Then use a decision tree as a model that uses the preselected inputs from all the available transformed inputs.
3. With the selected inputs from the previous step, you can impute missing values, use two types of variable selection (in this case R^2 and chi-square), and then use a Metadata node with the property Combine=All set in order to use the selected variables from either node. With these preselected inputs, you can train a backward selection regression and a neural network.
4. Use a Transform Variables node that keeps the best transformations, and then use those transformed inputs to train a decision tree.
5. Using the transformed inputs from the previous step, impute missing values, calculate both R^2 and chi-square variable selection, and use a Metadata node to combine the selected inputs of both variable selection nodes. Use the combined selected inputs to train a stepwise regression.
6. Using the selected and imputed inputs from the previous step, train a decision tree to detect interactions as described in Thompson (2012). To do this, specify the leaves of your decision tree as inputs. Using the extra information, train a main-effects regression.

The second template includes common practices of predictive modeling based on some SAS literature or custom settings for specific models. Figure 5 illustrates this template.

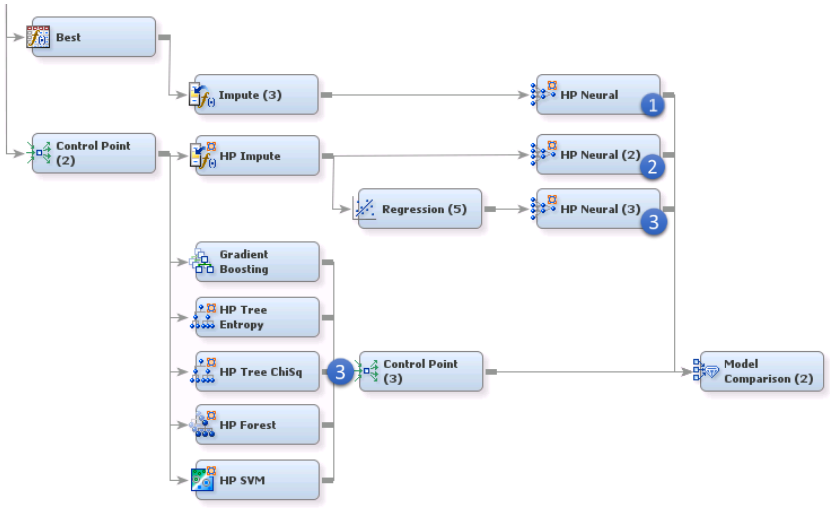


Figure 5. Diagram Flow with Candidate Models Based on Common Practices

The first part of the common practices diagram compares three neural networks:

1. Transformed inputs are imputed if necessary and used to train a neural network model that has 10 hidden units.
2. Inputs are imputed if necessary and used in a slightly larger neural network model (15 hidden units) without any specific transformation.
3. A common practice suggested in the course “Applied Analytics Using SAS Enterprise Miner” (Georges, Thompson, and Wells 2015) is to use a regression node to calculate the two-way interactions and then use them as inputs in a neural network that has a slightly different architecture (two layers).

The second part of the diagram compares models from nodes whose defaults are adjusted as follows:

4. A Gradient Boosting node with a lower Train Proportion value to help find more splits in case of rare events, two tree models from the HP Tree node with maximum depth and pruning options that mimic the Decision Tree node, and an HP Forest node with Maximum Number of Trees set to 100 and Minimum Use in Search of 1 to ensure that your forest model is combining a large number of trees that reached a large depth. A default HP SVM node with no imputation will be a model that is expected to be predictive for data without missing input values.

RESULTS

For each data set, the models of both templates were combined into the ensembles described in Figure 3. The results for every model and ensemble are summarized in more detail in Appendix 5.

Table 1 summarizes the misclassification rate for the best model from each template and the best-performing ensemble model. The misclassification improvement of each ensemble is calculated as the percentage of misclassification improvement relative to the better-performing model from either template.

	RPM-Based Model		Common Practice Model		Ensemble Model		
Data Set	**Model**	**Test Misclassification Rate**	**Model**	**Test Misclassification Rate**	**Ensemble**	**Test Misclassification Rate**	**Misclassification Improvement (%)**
Home Equity	Stepwise Regression	0.1034	Gradient Boosting	0.1006	Stacked Forest	0.0872	13.33%
German Credit	Decision Tree (best transformation)	0.2525	HPSVM	0.2259	Top-T	0.2326	-2.94%
Give Me Credit	Stepwise Regression	0.0640	HPNeural (2-way interactions)	0.0635	Top-T	0.0634	0.17%
PAKDD	Neural Network	0.2606	Several models	0.2609	Top-T	0.2608	-0.08%
Australian	Decision Tree (multiple transformations)	0.1531	HPNeural (best transformation)	0.1435	VarClus Top-T	0.1340	6.67%

Table 1. Misclassification Rates for Best-Performing Models and Ensembles

The diversity of the models in the templates seems to be advantageous, because no single type of model outperforms the others in all cases. With the exception of the stepwise regression model, which was the best RPM model for two data sets, each data set has a different best-performing model in both the RPM-based and common practice templates.

The template to test ensemble models was also very useful, because after it was coded, it was relatively easy to add to a diagram to test the improvement in misclassification of other ensemble models. Its run time was relatively short, and it found improved predictions.

For three out of five data sets, there was an ensemble model that improved the misclassification rate of the test partition. For the other two data sets, the best-performing ensemble made the misclassification slightly worse. In such cases, you should keep your best-performing model instead of an ensemble.

Table 2 compares the misclassification rates of all the ensembles in the templates. Notice that the top-*t* ensemble tended to have the lowest misclassification rate among the seven ensemble models for these data sets. Adding a variable clustering step before the top-*t* algorithm resulted in different misclassification curves, but both algorithms found ensembles with the same misclassification values for the test partition. In only one case (Australian data set), variable clustering helped decrease the misclassification rate further.

		Test Misclassification Rate				
		Home Equity	**German Credit**	**Give Me Credit**	**PAKDD**	**Australian**
Ensembles	Simple Average	0.10615	0.25581	0.06369	0.26090	0.14354
	Weighted Average	0.10330	0.27575	0.06403	0.26480	0.14426
	Stacked Tree	0.10978	0.26246	0.66710	0.27305	0.14353
	Stacked Neural	0.09050	1.00000	0.06362	0.27176	0.14354
	Stacked Forest	**0.08715**	1.00000	0.06686	0.28043	0.17225
	Varclus Top-T	0.10391	**0.23256**	0.06362	0.26083	**0.13397**
	Top-T	0.09888	**0.23256**	**0.06340**	**0.26077**	0.13876
	Hillclimbing	0.10110	0.23750	0.06586	0.26173	0.14333

Table 2. Misclassification Rate Comparison of Ensemble Models

No model or ensemble model consistently or very significantly outperformed the rest of the models. It seemed very valuable to compare all of them, and it was easy to use a SAS Enterprise Miner diagram as a template to benchmark, compare, and choose the appropriate model.

CONCLUSION

Ensemble methods are a good modeling technique to complement your data mining or predictive modeling because they take relatively little effort to code and they can find a model that improves your fit statistics.

Like any predictive model, an ensemble model is not guaranteed to outperform other models. It is still possible for a single model to be better than an ensemble, and it is up to an analytics professional to choose what model to put into a production environment based on the advantages and disadvantages of the models or ensembles under consideration.

The numerical analysis in this paper showed that several models and their ensembles tend to have similar fit statistics and showed that it is useful to have extra assessment plots. For the specific case of the data sets analyzed in the paper, the plot of events correctly predicted versus misclassification was particularly useful.

Beginning users are encouraged to experiment with the templates that provide modeling techniques based on RPM. More advanced users should compare their own models and ensembles to the ensemble template, as well as experiment with additional assessment plots. Users from any background will benefit from getting familiar with the architectural flexibility of SAS Enterprise Miner, which makes it simple and straightforward to code and benchmark new types of ensemble models.

REFERENCES

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). "Ensemble Selection from Libraries of Models." In *Proceedings of the 21st International Machine Learning Conference*. New York: ACM Press.

Georges, J., Thompson, J., and Wells, C. (2015). *Applied Analytics Using SAS Enterprise Miner Course Notes*. Cary, NC: SAS Institute Inc.

Lessmann, S., Baesens, B., Seow, H.-V., and Thomas, L. C. (2015). "Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring: An Update of Research." *European Journal of Operational Research* 247:124–136.

Maldonado, M., Dean, J., Czika, W., and Haller, S. (2014). "Leveraging Ensemble Models in SAS Enterprise Miner." In *Proceedings of the SAS Global Forum 2014 Conference.* Cary, NC: SAS Institute Inc. https://support.sas.com/resources/papers/proceedings14/SAS133-2014.pdf.

Qiang, F., Shang-Xu, H., and Sheng-Ying, Z. (2005). "Clustering-Based Selective Neural Network Ensemble." *Journal of Zhejiang University SCIENCE* A6:387–392.

Reid, S., and Grudic, G. (2009). "Regularized Linear Models in Stacked Generalization." *Multiple Classifier Systems* 5519:112–121.

SAS Institute Inc. (2010). SAS Rapid Predictive Modeler Product Brief. https://www.sas.com/content/dam/SAS/en_us/doc/productbrief/sas-rapid-predictive-modeler-104690.pdf.

Schubert, S. (2010). "The Power of the Group Processing Facility in SAS Enterprise Miner." In *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc. https://support.sas.com/resources/papers/proceedings10/123-2010.pdf.

Seni, G., and Elder, J. (2010). *Ensemble Methods in Data Mining: Improving Accuracy through Combining Predictions*. San Rafael, CA: Morgan and Claypool.

Thompson, D. (2012). "Methods for Interaction Detection in Predictive Modeling Using SAS." In *Proceedings of MidWest SAS Users Group Conference 2012.* Cary, NC: SAS Institute Inc. http://www.mwsug.org/proceedings/2012/SA/MWSUG-2012-SA01.pdf.

Ting, K. M., and Witten, I. H. (1999). "Issues in Stacked Generalization." *Journal of Artificial Intelligence Research* 10:271–289.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL: CRC Press.

ACKNOWLEDGMENTS

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

RECOMMENDED READING

- *Base SAS Procedures Guide*
- *Base SAS Procedures Guide: High-Performance Procedures*
- *SAS Enterprise Miner Reference Help*
- *SAS Enterprise Miner Extension Nodes: Developer's Guide*

CONTACT INFORMATION

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

Wendy Czika	Wendy.Czika@sas.com
Miguel M. Maldonado	MMaldonado@caribbean-financial.com
Ye Liu	Ye.Liu@sas.com

APPENDICES

APPENDIX 1: SIMPLE AVERAGING

Add the following code in a SAS Code node to create a table that transposes the posterior probabilities of all the models connected to a preceding Model Comparison node and that calculates the simple average probability for an ensemble model.

Output Summary

Output 1 shows the table that is created by this code node. Notice that for each observation, you have the posterior probability of each model with the corresponding SAS Enterprise Miner node ID as a suffix (Tree3, Tree2, HPNNA2, and so on) and the calculated simple average of all models (p).

VIEWTABLE: Work._temp

	_p_Tree3	_p_Tree2	_p_HPNNA2	_p_HPNNA3	_p_HPTree	_p_HPSVM	p
1	0.724137931	0.7266666667	0.5136640862	0.886266634	0.12597201	0.1418897302	0.6363136256
2	0.7708333333	0.8017621145	0.2380240987	0.3628699074	0.12597201	0.1528931826	0.7093020756
3	0.724137931	0.7266666667	0.0899142082	0.0947588553	0.12597201	0.1224494976	0.3605760674
4	0.7708333333	0.8017621145	0.9999471636	0.9731007178	1	0.3623079709	0.9008187228
5	0.7708333333	0.8017621145	0.3107757084	0.5862941291	0.12597201	0.1658895997	0.6619984792
6	0.724137931	0.7266666667	0.6033651506	0.5065466334	0.12597201	0.1467920431	0.5660277265
7	0.724137931	0.7266666667	0.1525343407	0.1885254933	0.12597201	0.1471337942	0.4479678013
8	0.9594594595	0.8017621145	0.9999924298	0.9121862381	0.95121951	0.4823150284	0.9252575928
9	0.7708333333	0.8017621145	0.0773843641	0.0670819355	0.12597201	0.0031028269	0.6146449131
10	0.724137931	0.7266666667	0.5187961989	0.6819843387	0.12597201	0.147357641	0.6610912901

Output 1. Output from the Simple-Averaging Workaround

Code

```
%macro simpleaverage(_partition,_nodeid,_outputdata);

data _NULL_;
set &EM_DEC_DECMETA;
   where _type_="TARGET";
   call symput('_TARGET',STRIP(VARIABLE));
   call symput('_EVENT',STRIP(EVENT));
run;

%let _workspaceid=&EM_LIB;
%let _predictedtargetevent=P_&_TARGET&_EVENT;
%let _predictedtarget=&_TARGET;

proc sort data=&_workspaceid..&_nodeid._emreportfit out=_v(keep=model fitstat train validate
test);
where fitstat="_MISC_";
by test;
run;

data _null_;
set _v end=eof;
    call symput('_model'!!strip(put(_N_, BEST.)), strip(model));
    call symput('_fitstat'!!strip(put(_N_, BEST.)), strip(validate));
    if eof then call symput('_num_model', strip(put(_N_, BEST.)));
run;

data &_outputdata;
 merge
 %do i=1 %to &_num_model;
       &_workspaceid..&&_model&i.._&_partition(keep=_dataobs_ &_predictedtargetevent
       &_predictedtarget rename=(&_predictedtargetevent=_p_&&_model&i ))
 %end;   end=eof;;
  by _dataobs_;
  &_predictedtargetevent=mean(of _p_:);
  F_&_TARGET=&_predictedtarget;
```

```
  R_&_TARGET&_EVENT=&_predictedtargetevent;
  run;
%mend;

%simpleaverage(_partition=test     ,_nodeid=&_MdlComp ,_outputdata=&EM_EXPORT_TEST);
%simpleaverage( partition=train    , nodeid=& MdlComp , outputdata=&EM EXPORT TRAIN);
%simpleaverage(_partition=validate ,_nodeid=&_MdlComp ,_outputdata=&EM_EXPORT_VALIDATE);
```

APPENDIX 2: TOP-*T* ENSEMBLE

Add the following code in a SAS Code node to combine models according to a top-*t* methodology. Connect two or more modeling nodes to a Model Comparison node, connect a SAS Code node that calculates a simple average ensemble as described in Appendix 1, and then connect a SAS Code node by using the following code. Remember to specify the property Tool Type as Model. By default, this code outputs the top-*t* ensemble that has the lowest misclassification rate on the test partition.

```
/*Retrieve the event and non-event of the target*/
data _NULL_;
set &EM_DEC_DECMETA;
   if _type_="TARGET" then do;
      call symput('_TARGET',strip(VARIABLE));
      call symput('_EVENT',strip(EVENT));
   end;
   if _type_="PREDICTED" then do;
      if level=symget('_EVENT') then call
symput('_TARGETEVENT',strip(tranwrd(VARIABLE,"P_","")));
       else call symput('_TARGETNONEVENT',strip(tranwrd(VARIABLE,"P_","")));
   end;
run;

/*Retrieve the Node Ids of the Model Comparison and the Simple Average node */
data _NULL_;
set &EM_LIB..em_dgraph;
if TO="&EM_NODEID" then call symput('_SAVGID',strip(FROM));
run;
data _NULL_;
set &EM_LIB..em_dgraph;
if TO="&_SAVGID" then call symput('_NODEID',strip(FROM));
run;

/*Create arrays for models*/
proc sort data=&EM_LIB..&_nodeid._emreportfit out=_validation(keep=model fitstat train validate
test);
where fitstat="_MISC_";
by validate;
run;

data _null_;
set _validation end=eof;
    call symput('_model'!!strip(put(_N_, BEST.)), strip(model));
   if eof then call symput('_num_model', strip(put(_N_, BEST.)));
run;

%macro toptensemble(_partition,_outtable);

%do j=1 %to &_num_model;
/* data _temp&j; */
   data _temp;
   length I_&_TARGET $8;
   merge
   %do i=1 %to &j;
      &EM_LIB..&_SAVGID._&_partition(keep=_dataobs_ &_TARGET _p_&&_model&i)
   %end; end=eof;
   by _dataobs_;

   P_&_TARGETEVENT=mean(of _p:);
   P_&_TARGETNONEVENT=1-P_&_TARGETEVENT;

   R_&_TARGETEVENT=1-P_&_TARGETEVENT;
   R_&_TARGETNONEVENT=1-P_&_TARGETNONEVENT;
```

```
   F_&_TARGET=upcase(strip(&_TARGET));

   if (P_&_TARGETEVENT gt 0.5) then I_&_TARGET=upcase("&_EVENT");
      else I_&_TARGET=upcase(%quote(strip(tranwrd("&_TARGETNONEVENT","&_TARGET",""))));


   if F_&_TARGET eq I_&_TARGET then corr_class=1;
         else corr_class=0;
      obscount+1;
      corr_classcount+corr_class;
      misc=(obscount-corr_classcount)/obscount;
      if eof then call symput('_ensmisc'!!strip(put(&i, BEST.)), strip(misc));
   run;

   data _ens&j;
   label ensid="ensemble id";
   set _temp end=eof;
   if eof;
      keep misc ensid;
      ensid=&j;
   run;
%end;

data &_outtable;
set %do k=1 %to &_num_model; _ens&k %end; ;
rename misc=misc_&_partition;
run;

%mend;

%toptensemble(_partition=test,    _outtable=modelorder)

data modelorder;
set modelorder;
step=_n_;
run;

proc sort data=modelorder out=stop_point;
by misc_test step;
run;

data _null_;
set stop_point;
if _n_=1;
call symput('_stoppoint', strip(step));
run;

%macro calculateSAverage(_partition,_outtable);
data &_outtable;
   length I_&_TARGET $8;
   merge
   %do i=1 %to &_stoppoint;
      &EM_LIB..&_SAVGID._&_partition(keep=_dataobs_ &_TARGET _p_&&_model&i)
   %end; end=eof;
   by _dataobs_;

   P_&_TARGETEVENT=mean(of  p:);
   P_&_TARGETNONEVENT=1-P_&_TARGETEVENT;

   R_&_TARGETEVENT=1-P_&_TARGETEVENT;
   R_&_TARGETNONEVENT=1-P_&_TARGETNONEVENT;

   F_&_TARGET=upcase(strip(&_TARGET));

   if (P_&_TARGETEVENT gt 0.5) then I_&_TARGET=upcase("&_EVENT");
      else I_&_TARGET=upcase(%quote(strip(tranwrd("&_TARGETNONEVENT","&_TARGET",""))));

run;
%mend;
```

```
%calculateSAverage(_partition=train    ,_outtable=&EM_EXPORT_TRAIN)
%calculateSAverage(_partition=validate ,_outtable=&EM_EXPORT_VALIDATE)
%calculateSAverage(_partition=test     ,_outtable=&EM_EXPORT_TEST)
```

APPENDIX 3: HILL-CLIMBING

The code in this appendix calculates the posterior probabilities of a hill-climbing ensemble for all your partitions. Connect two or more modeling nodes to a Model Comparison node, connect a SAS Code node that calculates a simple average ensemble as described in Appendix 1, and then connect a SAS Code node by using the following code. Remember to specify the property Tool Type as Model.

Code

```
/*Retrieve the event and non-event of the target*/
data _NULL_;
set &EM_DEC_DECMETA;
   if _type_="TARGET" then do;
      call symput('_TARGET',strip(VARIABLE));
      call symput('_EVENT',strip(EVENT));
   end;
   if _type_="PREDICTED" then do;
      if level=symget('_EVENT') then call
symput('_TARGETEVENT',strip(tranwrd(VARIABLE,"P_","")));
         else call symput('_TARGETNONEVENT',strip(tranwrd(VARIABLE,"P_","")));
   end;
run;

/*Retrieve the Node Ids of the Model Comparison and the Simple Average node */
data _NULL_;
set &EM_LIB..em_dgraph;
if TO="&EM_NODEID" then call symput('_SAVGID',strip(FROM));
run;
data _NULL_;
set &EM_LIB..em_dgraph;
if TO="&_SAVGID" then call symput('_NODEID',strip(FROM));
run;

/*Create arrays for models*/
proc sort data=&EM_LIB..&_nodeid._emreportfit out=_validation(keep=model fitstat train validate
test);
where fitstat="_MISC_";
by validate;
run;

data _null_;
set _validation end=eof;
    call symput('_model'!!strip(put(_N_, BEST.)), strip(model));
   if eof then call symput('_num_model', strip(put(_N_, BEST.)));
run;

data modelorder(keep=model misc_test);
model=1;
set  validation(keep=test rename=(test=misc_test));
if _n_=1;
run;

/*calculate probability of event as a simple average*/
%macro
_calculate_SA_prob_event(_partition,_whereclause="",_outputtable=initialtable,_additionalcalc=);
   data _null_;
   set modelorder&_whereclause end=eof;
   call symput('_nsequence'!!strip(put(_N_, BEST.)), strip(model));
   if eof then call symput('_niterations', strip(put(_N_, BEST.)));
   run;

   data &_outputtable;
   length I_&_TARGET $8;
      merge
   %do j=1 %to &_niterations;
      &EM_LIB..&_SAVGID._&_partition(keep=_dataobs_ &_TARGET _p_&&&&_model&&_nsequence&j)
   %end; end=eof;
```

```
   by _dataobs_;

   pacum=mean(of _p:);         /*simple average probability for models already in the sequence*/

   %if "&_additionalcalc"="Y" %then %do; /*simple average probability for final models already in
the sequence*/
   %_calculate_p_r_f_i(_TARGETEVENT=&_TARGETEVENT,_TARGETNONEVENT=&_TARGETNONEVENT,_TARGET=&_TARG
ET,_EVENT=&_EVENT);
   %end;
   %else %do;
   keep _dataobs_ _pacum;
   rename pacum=_pacum;
   %end;

   run;

%mend;

/*calculate probabilities residuals from into variables*/
%macro _calculate_p_r_f_i(_TARGETEVENT,_TARGETNONEVENT,_TARGET,_EVENT);
   P_&_TARGETEVENT=mean(of  p:);
   P_&_TARGETNONEVENT=1-P_&_TARGETEVENT;

   R_&_TARGETEVENT=1-P_&_TARGETEVENT;
   R_&_TARGETNONEVENT=1-P_&_TARGETNONEVENT;

   F_&_TARGET=upcase(strip(&_TARGET));

   if (P_&_TARGETEVENT gt 0.5) then I_&_TARGET=upcase("&_EVENT");
      else I_&_TARGET=upcase(%quote(strip(tranwrd("&_TARGETNONEVENT","&_TARGET",""))));
%mend;


%macro hillclimbing(_partition,_modelorder);

   %_calculate_SA_prob_event(_partition=&_partition);

%do i=1 %to &_num_model;
   data  temp&i;
   length I_&_TARGET $8;
      merge initialtable
              &EM_LIB..&&_model&i.._&_partition(keep=_dataobs_ P_&_TARGETEVENT &_TARGET
                                                    rename=(P_&_TARGETEVENT=_p_&&_model&i))
   end=eof;
   by _dataobs_;

   %_calculate_p_r_f_i(_TARGETEVENT=&_TARGETEVENT,_TARGETNONEVENT=&_TARGETNONEVENT,_TARGET=&_TARG
ET,_EVENT=&_EVENT);

   if F_&_TARGET eq I_&_TARGET then corr_class=1;
         else corr_class=0;
      obscount+1;
      corr_classcount+corr_class;
      misc=(obscount-corr_classcount)/obscount;
      if eof then call symput('_ensmisc'!!strip(put(&i, BEST.)), strip(misc));

   run;

   data _ens&i;
   label ensid="ensemble id";
   set _temp&i end=eof;
   if eof;
      keep misc ensid;
      ensid=&i;
   run;
%end;
```

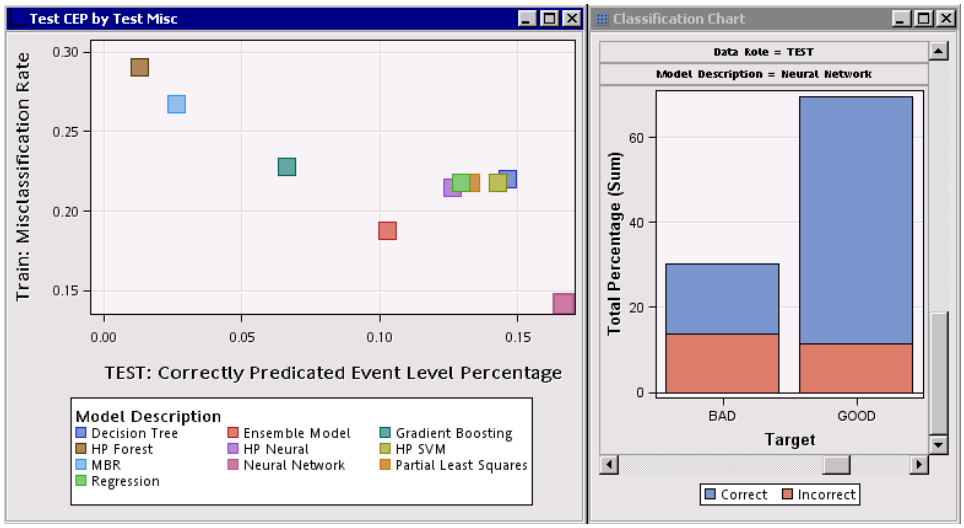
```
data outtable;
set %do k=1 %to &_num_model; _ens&k %end; ;
rename misc=misc_&_partition;
run;

proc sort data=outtable;
by misc_&_partition;
run;

data toappend;
set outtable;
by misc &_partition;
if _n_=1;
run;

data &_modelorder;
set &_modelorder toappend(rename=(ensid=model));
run;

%mend;

%macro iteratehc;
   %do _i=1 %to 10;
      %hillclimbing(_partition=test,_modelorder=modelorder)
   %end;
%mend;

%iteratehc

data modelorder;
set modelorder;
step=_n_;
run;

proc sort data=modelorder out=stop_point;
by misc_test step;
run;

data _null_;
set stop point;
if _n_=1;
call symput('_stoppoint', strip(step));
run;

%_calculate_SA_prob_event(_partition=train    ,_whereclause=(where=(step le
& stoppoint)), outputtable=&EM EXPORT TRAIN,    additionalcalc=Y);
%_calculate_SA_prob_event(_partition=validate,_whereclause=(where=(step le
& stoppoint)), outputtable=&EM EXPORT VALIDATE, additionalcalc=Y);
%_calculate_SA_prob_event(_partition=test  ,_whereclause=(where=(step le
&_stoppoint)),_outputtable=&EM_EXPORT_TEST    ,_additionalcalc=Y);
```

APPENDIX 4: ADDITIONAL ASSESSMENT

Add the following code in a SAS Code node following a Model Comparison node to produce a classification chart for all the partitions in your data set (Train, Test, Validation) as well as different scatter plots for misclassification, area under the ROC curve, and events correctly predicted.

Output Summary

The results of this node include several plots, such as those shown in Output 2.



Output 2. Output from the Additional Assessment Code

Code

```
/*Retrieve the Node Id of the Model Comparison node */
data _NULL_;
set &EM_LIB..em_dgraph;
if TO="&EM_NODEID" then call symput('_NODEID',strip(FROM));
run;

/*Retrieve the event and non-event of the target*/
data _NULL_;
set &EM_DEC_DECMETA;
   if _type_="TARGET" then do;
      call symput('_TARGET',strip(VARIABLE));
      call symput('_EVENT',strip(EVENT));
   end;
   if _type_="PREDICTED" then do;
      if level=symget('_EVENT') then call
symput('_TARGETEVENT',strip(tranwrd(VARIABLE,"P_","")));
   end;
run;

%let_workspaceid=&EM_LIB;

%macro CORRECT_EVT_FREQ_CALC;

   proc sort data=&_workspaceid..&_nodeid._emreportfit out=_v(keep=model fitstat train validate
   test);
   where fitstat="_MISC_";
   by test;
   run;
```

```
    data _null_;
    set _v end=eof;
        call symput('_model'!!strip(put(_N_, BEST.)), strip(model));
        call symput('_fitstat'!!strip(put(_N_, BEST.)), strip(test));
                    if eof then call symput('_num_model', strip(put(_N_, BEST.)));
    run;



    %do i=1 %to &_num_model;
     data  temp test && model&i.;
          set  &_workspaceid..&&_model&i.._test(keep=_dataobs_ &_target F_&_TARGET I_&_TARGET);
          if F_&_TARGET eq I_&_TARGET then corr_class=1;
             else corr_class=0;
          obscount+1;
          corr_classcount+corr_class;
          misc=(obscount-corr_classcount)/obscount;

     run;


    proc tabulate data=_temp_test_&&_model&i. out=_temp_test_N_&&_model&i.;
    class corr_class &_TARGET;
    table  &_TARGET,corr_class;
    run;

    proc sql;
    create table classification test && model&i. as
    select  UPPER(&_TARGET) as from, (N/sum(N)) as percent,"&&_model&i." length=10 as Model,
    case
    when (corr_class=1) then 'Correct'
    when (corr_class=0) then 'Incorrect'
    end as correcttext
    from _temp_test_N_&&_model&i.;
    quit;
     %end;

    data _classification_test_all_models;
       set  %do i=1 %to &_num_model;
    classification_test_&&_model&i.
    %end;
    ;
    label datarole='Data Role';
    label percent='Total Percentage';
    label from='Target';
    datarole="TEST";
    *ModelDescription='Ensemble';
    percent=percent*100;
    run;

%mend;

%CORRECT_EVT_FREQ_CALC;

proc sql;
create table lookupmodel as
select distinct model, modeldescription from &EM_LIB..&_Nodeid._emclassification;
quit;

proc sort data=_classification_test_all_models;
by model;
run;

data _classification_test_all_models;
merge _classification_test_all_models lookupmodel;
by model;
run;
```

```
data emclassification;
set &EM_LIB..&_Nodeid._emclassification(keep=from percent modeldescription correcttext datarole)
 classification_test_all_models(drop=model);
run;

%em_register(key=classification, type=DATA);

proc sort data=emclassification out=&em_user_classification;
by modeldescription;
run;

/*Generate Classification Chart*/
%em_report(key=classification, viewtype=lattice, latticetype=bar, latticex=ModelDescription,
latticey=datarole, x=from, group=correctText, freq=percent, description=%nrbquote(Classification
Chart), autoDisplay=Y);

/*Fit Stat Plots*/
%em_register(key=fitstat, type=DATA);
Data correct_pred_evt_percent;
drop percent;
set &em_user_classification(keep=modeldescription percent datarole from correctText);
where from="& EVENT" and correctText="Correct";
correct_pred_evt_percent=percent/100;
run;

data CEP_TRAIN CEP_VALIDATE CEP_TEST;
drop datarole;
set correct_pred_evt_percent;
if datarole="TRAIN" then output CEP_TRAIN;
else if datarole="VALIDATE" then output CEP_VALIDATE;
else output CEP_TEST;
run;

data CEP;
drop from correcttext;
merge CEP_TRAIN(rename=(correct_pred_evt_percent=_CEP_))
CEP_VALIDATE(rename=(correct_pred_evt_percent=_VCEP_))
CEP_TEST(rename=(correct_pred_evt_percent=_TCEP_));
by modeldescription;
run;

data fitstat;
 set &EM_LIB..&_Nodeid._emoutfit(keep=modeldescription _ASE_ _AUR_ _MISC_ _VASE_ _VAUR_ _VMISC_
_TASE_ _TAUR_ _TMISC_);
run;
proc sort data=fitstat out=fitstat;
by modeldescription;
run;

data &em_user_fitstat;
label _CEP_="TRAIN: Correctly Predicted Event Level Percentage";
label _VCEP_="VALIDATE: Correctly Predicted Event Level Percentage";
label _TCEP_="TEST: Correctly Predicted Event Level Percentage";
merge CEP fitstat;
by modeldescription;
run;
```

```
%em_report( key=fitstat, viewtype=Scatter, x=_ASE_, y=_AUR_, block=%bquote(Training Fit
Statistics Scatter Plot), description=%bquote(Training Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_ASE_, y=_MISC_, block=%bquote(Training Fit
Statistics Scatter Plot), description=%bquote(Training Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_AUR_, y=_MISC_, block=%bquote(Training Fit
Statistics Scatter Plot), description=%bquote(Training Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_CEP_, y=_MISC_, block=%bquote(Training Fit
Statistics Scatter Plot), description=%bquote(Training Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);


%em_report( key=fitstat, viewtype=Scatter, x=_VASE_, y=_VAUR_, block=%bquote(Validation Fit
Statistics Scatter Plot), description=%bquote(Validation Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_VASE_, y=_VMISC_, block=%bquote(Validation Fit
Statistics Scatter Plot), description=%bquote(Validation Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_VAUR_, y=_VMISC_, block=%bquote(Validation Fit
Statistics Scatter Plot), description=%bquote(Validation Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_VCEP_, y=_MISC_, block=%bquote(Validation Fit
Statistics Scatter Plot), description=%bquote(Validate Fit Statistics Scatter Plot),
group=modeldescription, autodisplay=Y);


%em_report( key=fitstat, viewtype=Scatter, x=_TASE_, y=_TAUR_, block=%bquote(Test Fit Statistics
Scatter Plot), description=%bquote(Test ASE by Test AUC), group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_TASE_, y=_TMISC_, block=%bquote(Test Fit Statistics
Scatter Plot), description=%bquote(Test ASE y Test Misc), group=modeldescription, autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_TAUR_, y=_TMISC_, block=%bquote(Test Fit Statistics
Scatter Plot), description=%bquote(Test AUC by Test Misc), group=modeldescription,
autodisplay=Y);
%em_report( key=fitstat, viewtype=Scatter, x=_TCEP_, y=_MISC_, block=%bquote(Test Fit Statistics
Scatter Plot), description=%bquote(Test CEP by Test Misc), group=modeldescription,
autodisplay=Y);
```

APPENDIX 5: DETAILED RESULTS

Table 3 shows the misclassification rate for the 15 candidate models of the ensemble and for the 8 ensemble models. The lowest misclassification rate for each section (RPM-based model, common practice model, or ensemble model) is shown in bold.

			Test Misclassification Rate				
			Home Equity	German Credit	Give Me Credit	PAKDD	Australian
Models	RPM-Based	Forward Selection	0.10447	0.25249	0.06395	0.26157	0.15790
		Backward Selection	0.10670	0.26578	0.06409	0.26410	0.16268
		Neural Network	0.10391	0.26578	0.06482	**0.26057**	0.17703
		Decision Tree (Mult)	0.12514	0.27575	0.06478	0.26130	**0.15311**
		Main Effects Regression	0.10447	0.25581	0.06520	0.26190	0.17703
		Stepwise Selection	**0.10335**	0.25581	**0.06395**	0.26183	0.15790
		Decision Tree (Best)	0.12291	**0.24253**	0.06498	0.26090	0.15790
	Common Practice	HPNeural (Best)	0.10391	0.26578	0.06389	0.26197	**0.14354**
		HPNeural	0.14525	0.23920	0.06353	0.26503	0.15311
		HPNeural (2-way)	0.14525	0.23588	**0.06351**	**0.26090**	0.15790
		Gradient Boosting	**0.10056**	0.25581	0.06375	**0.26090**	0.15790
		HPForest	0.10950	0.28904	0.06502	**0.26090**	0.16268
		HPSVM	0.19274	**0.22591**	0.06458	0.31469	0.16746
		HPTree (Entropy)	0.16089	0.28571	0.06513	0.26103	0.39234
		HPTree (ChiSq)	0.16369	0.28904	0.06466	0.26090	0.39713
Ensembles		Simple Average	0.10615	0.25581	0.06369	0.26090	0.14354
		Weighted Average	0.10330	0.27575	0.06403	0.26480	0.14426
		Stacked Tree	0.10978	0.26246	0.66710	0.27305	0.14353
		Stacked Neural	0.09050	1.00000	0.06362	0.27176	0.14354
		Stacked Forest	**0.08715**	1.00000	0.06686	0.28043	0.17225
		Varclus Top-T	0.10391	**0.23256**	0.06362	0.26083	**0.13397**
		Top-T	0.09888	**0.23256**	**0.06340**	**0.26077**	0.13876
		Hillclimbing	0.10110	0.23750	0.06586	0.26173	0.14333

Table 3. Misclassification Rate Summary for Models and Ensemble Models