› White Paper

Machine Learning With SAS® Enterprise Miner™

How a Team of SAS® Modelers Created and Determined a Champion Model to Predict Churn Using KDD Cup Data

sas
THE POWER TO KNOW®

Contents

Content provided by:

Jonathan Wexler is the Principal Product Manager responsible for SAS Enterprise Miner and SAS High-Performance Analytics. He has an undergraduate degree in mathematics/statistics and a master's degree in biostatistics, with more than 18 years of experience using SAS.

Philip Easterling has 20 years of SAS analytical software sales and consulting experience, including predictive modeling in service industries and government agencies, optimization for logistics industries and a broad range of analytic methods in the oil and gas industry. He also has more than seven years of quantitative business experience in the US commercial airline industry. He has a BA in mathematical sciences and economics from Rice University and a master's degree in applied operations research from the University of California at Berkeley.

Introduction

There are two general goals of statistical analysis:

- Making inferences about populations from the analysis of a statistical sample.
- Making predictions about future outcomes based upon patterns and relationships in observed historical data.

While both of these goals are important to many organizations, making accurate predictions is critical in many commercial, scientific, medical and government problems. Consequently, these organizations build predictive statistical models to reflect relevant patterns in observed historical data and then use these models to make predictions. Obviously, they emphasize building the most accurate predictive models possible, using numerous statistical algorithms and data preparation techniques.

Periodically, there are open competitions where organizers provide a set of historical data with known outcomes to participants and then use the participants' predictive statistical model entries, built from this historical data, to make outcome predictions on another data set that isn't revealed publicly. The objective metric for determining the winner is the predictive accuracy of each participant's statistical model on the unreleased data set, based upon how closely the participants' model predictions match the outcomes (which only the organizer knows).

One widely publicized competition in recent years has been the Knowledge Discovery in Databases (KDD) Cup competition. In 2009, KDD distributed data for the classic marketing problem of churn (predicting which customers will end their relationship with a business and choose a competitor). In this scenario, the historical data set contained a unique record for a customer, where the columns represented independent variables (usually reflecting demographics, transactional history, etc.) and the outcome (churn or non-churn). The data set variable names were masked by the organizer so that they didn't reflect any business relevance. Thus, no one could use business domain knowledge in their solution – only data analysis and mathematical methods.

Recently, a team of modelers from SAS decided to use SAS® Enterprise Miner™ software and the 2009 KDD data to build a highly accurate churn model. As with much real-world data, the SAS team discovered many missing values in the KDD data. In subsequent sections, the team explains its approach to treating missing data values so it could still include these variables in their analysis.

SAS® Data Mining Context

The SAS team modeled the data set from the 2009 KDD competition using the most current version of the SAS® Enterprise Miner™ software. SAS Enterprise Miner has extensive capabilities for all aspects of a comprehensive data mining or machine-learning process. A process-flow-based GUI, drag-and-drop task-oriented icons and prompting wizards make it easy to assemble a data mining or machine-learning solution. The software includes data preparation techniques, variable selection methods, machine-learning predictive modeling algorithms, text mining approaches, model assessment and numerous other tasks. Users have control over the relevant properties and parameter values associated with each task, although these tasks are initially configured with data-driven default values for the properties and parameter values.

All of these SAS Enterprise Miner features and capabilities make it very convenient to take an iterative approach to data mining and machine learning. After viewing the results of any task, simply make changes to the task property settings or parameter values and rerun the relevant task(s). This lets modelers automate previously time-consuming tasks and greatly increases productivity.

Often, modelers use the SAS Enterprise Miner interface to build competing machine-learning predictive models in parallel and compare the results of each model with a common accuracy metric, essentially conducting a modeling "tournament." The SAS team used this approach. Modelers can incorporate SAS/STAT® and SAS/IML® models in their SAS Enterprise Miner projects, and this paper includes an example of a SAS/IML model. Additionally, there are capabilities for incorporating open-source models into a SAS Enterprise Miner project and conveniently comparing those results with the results of SAS Enterprise Miner algorithms. The SAS team also used this capability.

SAS Enterprise Miner can run on a single computer or server as well as in a distributed, massively parallel (MPP) computing environment. In an MPP deployment, the data and the processing are distributed across multiple nodes in the computing environment. The resulting scalability significantly reduces processing times, even for computationally intensive tasks like nonlinear modeling.

This paper shows how the SAS team used SAS Enterprise Miner with the 2009 KDD data to build several kinds of predictive models and determine the most accurate one.

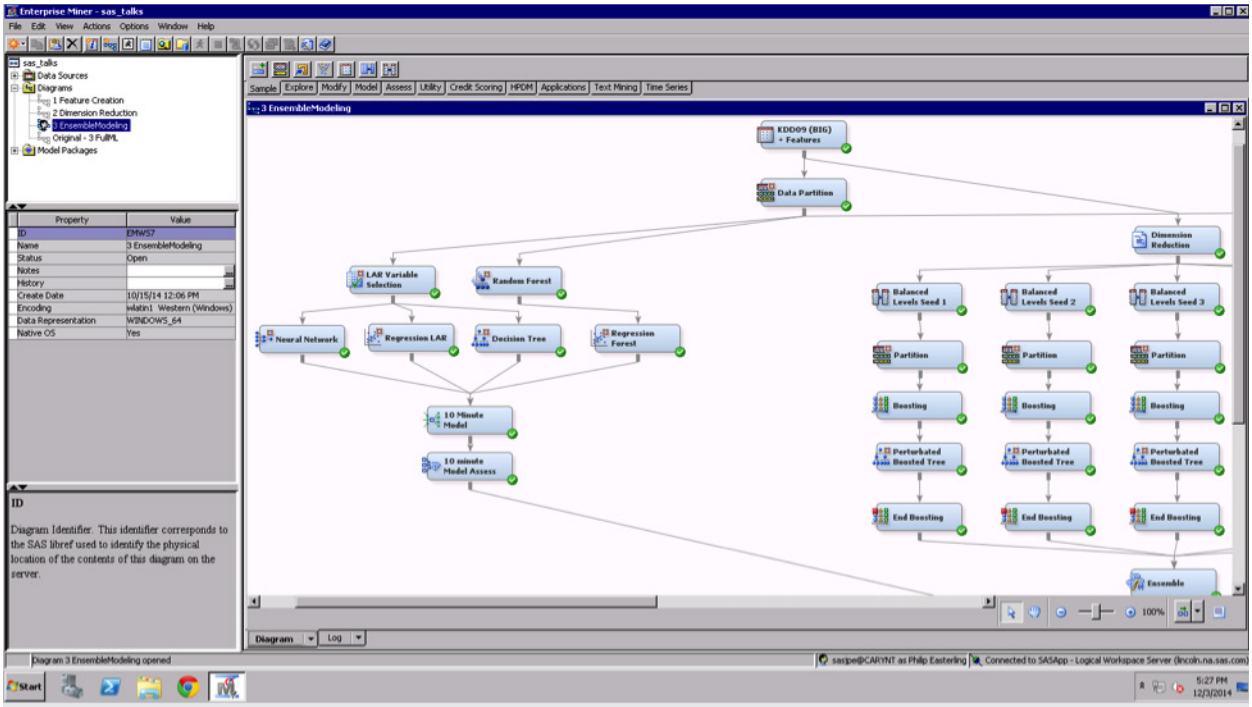


Figure 1: An example of the SAS Enterprise Miner GUI with a sample data mining process flow. The GUI makes it easy to build data mining and machine-learning solutions.

Overview

Seeking an ambitious challenge, the SAS team modeled the big data set using the churn target variable because the contest winners stated that churn was the most difficult target to predict.[1] Because the data set variables and values were masked, we could not apply business domain knowledge to the solution. This required the use of nuanced data science practices to approach the problem.

We divided the problem into three primary categories: feature creation, dimension reduction and predictive modeling. The team used a variety of data transformations, standardizations, variable selection and ensemble modeling techniques to arrive at an area under the curve (AUC/ROC) of 0.853 using a validation data set partition to assess the viability of the model.

We ran SAS Enterprise Miner on a standard SAS server to arrive at the final model. We were able to deploy the model and score the full data set inside Hadoop in under a minute.

Now we'll walk you through the methodology used in our process.

Feature Creation

Data Cleansing

First, we downloaded the large data set from the KDD website in pieces as chunked text files. Then we assembled the chunks and converted them into to a single SAS data set for the rest of the process. We used the entire data set, which had 50,000 records and 15,005 variables, including multiple targets and ID variables.

Several variables were sparsely populated (greater than 90 percent missing values), so we needed to address missing values. Normally, we would have used imputation methodology to replace missing values with statistically generated values. However, we theorized that such a high number of imputed missing values could skew the distribution of the original variables and bias our predictions. Instead, we addressed missing values by optimal binning, bucketing and binary encoding. We considered dropping the highly missing variables, but, given the high number of missing values, we realized these might have a significant effect in a predictive model.

[1] Guyon, Isabelle, et al. "Analysis of the KDD Cup 2009: Fast scoring on a large Orange customer database." (2009)

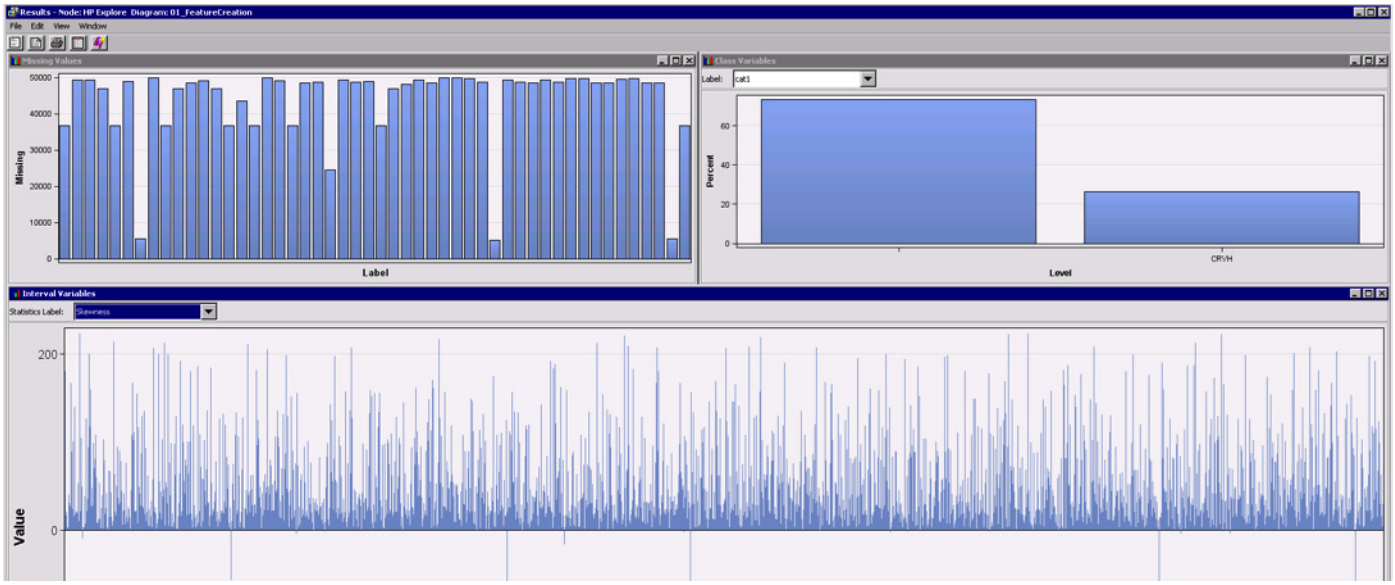


Figure 2: Variables contained many missing values and were skewed.

Many of the numeric variables were heavily skewed (an absolute value of skewness greater than one indicates heavy skew). As shown above in Figure 2, many variables were extremely skewed (with an absolute value of skewness greater than 200), which implies extreme values. Nearly all variables were right-skewed (positive skewness values), meaning the tails of the distribution were favored to the right. A handful of variables were left-skewed (negative skewness values).

Data Transformations

For numeric variables with missing values, we decided to create "optimal bins." Optimal bins are created by converting a numeric variable into a categorical variable in such a way that its relationship with churn is maximized in terms of a purity measure, such as chi-square or Gini coefficient. SAS Enterprise Miner automatically determines the number of bins per variable, so no manual binning is required. (See Figure 3.)

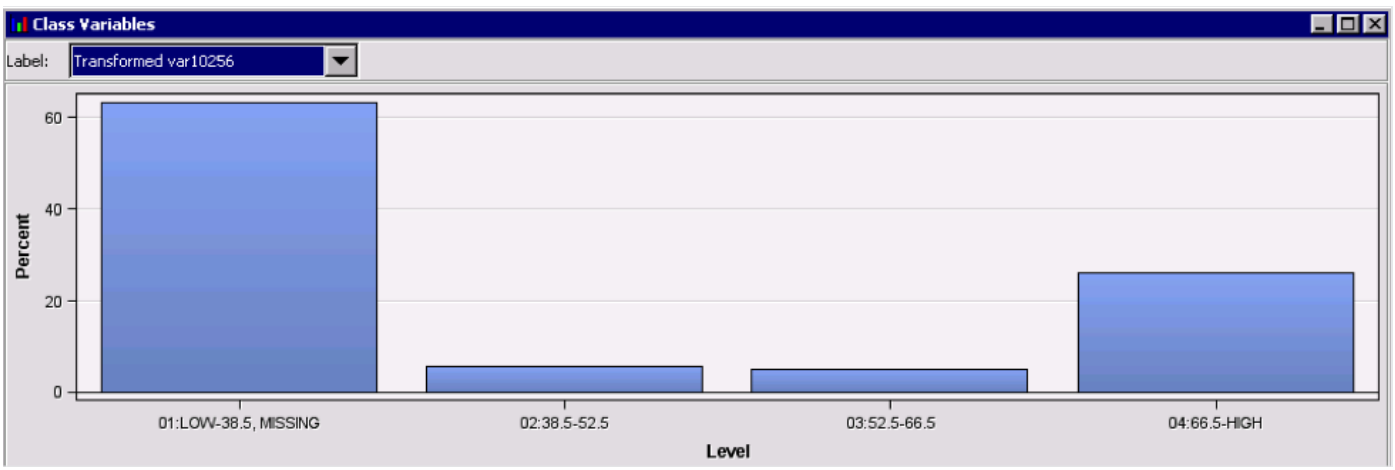


Figure 3: SAS Enterprise Miner automatically performs optimal binning for numeric variables.

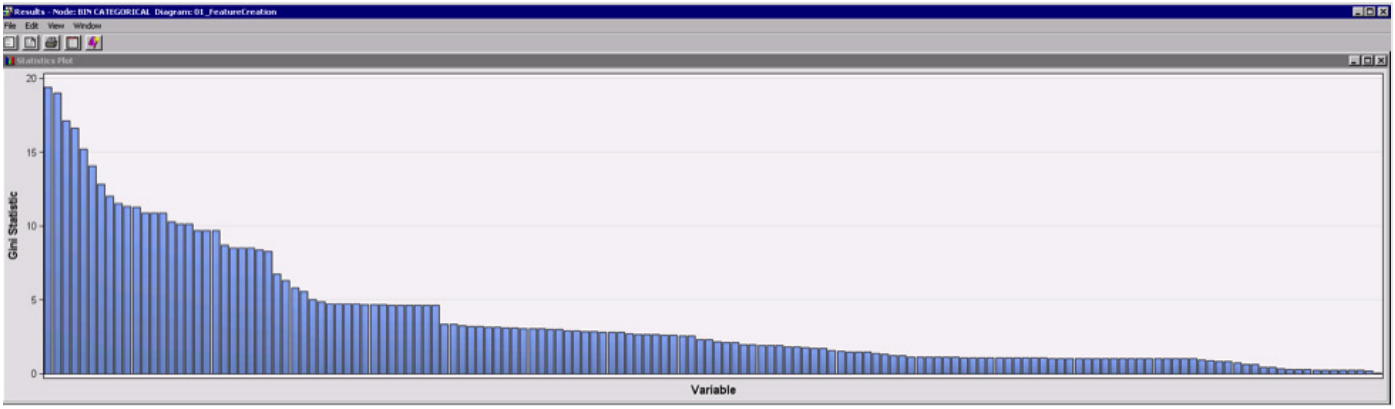


Figure 4: Variable worth (Gini) before binning.

For categorical variables with missing values, we created variables consisting of approximately 10 to 15 buckets, including a bucket for missing values. Categorical variables with many levels were treated in the same manner, but all rare levels were placed into a bucket together. Variables were automatically rank-ordered according to Gini coefficient. Variables with a Gini coefficient value less than 10 were not binned. Please note that, for simplicity, the Gini coefficient calculation was multiplied by 100. (See Figure 4.)

Numeric inputs with no missing values were handled by trimming the extreme values using Winsorized means. This replaces values on the tail ends of the distribution of a variable with more reasonable values. (See Figure 5.) We then standardized all numeric variables because machine-learning algorithms often involve many iterations that use numeric derivatives. These kinds of computations can degrade numeric precision if there is too large of a range of values in any of the numeric input variables.

Interval Variables

Variable	Replacement Method	Lower Replacement Value	Upper Replacement Value
var1	COMPUTED	-0.70205	0.705565
var10002	COMPUTED	-0.87947	0.920033
var10003	COMPUTED	-1.79488	1.91368
var10004	COMPUTED	-15.1561	15.71695
var10005	COMPUTED	-2736591	5037185
var10007	COMPUTED	-73.3509	101.5741
var10008	COMPUTED	-1.47257	1.522726
var1001	COMPUTED	-325.675	415.4154
var10010	COMPUTED	-20.0207	20.5974
var10011	COMPUTED	-0.67582	0.678623
var10014	COMPUTED	-3.42935	3.465831

Figure 5: An example of outlier trimming using Winsorized means and standard deviation.

We encoded all original binary variables, even those consisting only of zero and missing, to (1,-1) so that these variables would fall nicely in the center of the range of the standardized interval variables and not artificially overweight the tails of our input variable distributions. Encoding binary variables to numeric values also gave us the flexibility to treat them as either numeric or categorical, based upon the needs of any given modeling algorithm. If an algorithm is capable of handling binary variables as truly classification variables (dummy variables), like most SAS algorithms can, then it doesn't matter if they are (1,0) or (-1,1). For example, R algorithms treat all inputs as numeric, no matter what. So, if interval variables are standardized between -3 and 3, with a mean of 0, and binary variables are set at (1, 0), the R algorithm will artificially pull the data toward the positive area of the data values.

Below is an example of the syntax for the logic rules that were automatically generated by SAS Enterprise Miner for binary encoding of variables (Figure 6).

Unary variables, in which all values of the variable are identical, were dropped from the analysis.

Our modeling data set was now complete and ready for dimension reduction. It contained 13,113 variables after cleaning and transformations. We shared the data set with the rest of the modeling team through the built-in SAS metadata integration, which incorporates all transformations automatically.

```
if upcase(_xc)= "EHYH" then BE__CAT104_eHYH= 1.0; else BE__CAT104_eHYH= -1.0;
_xc= CAT105;
if upcase(_xc)= "VXUB" then BE__CAT105_vXUB= 1.0; else BE__CAT105_vXUB= -1.0;
_xc= CAT110;
if upcase(_xc)= "2BGT" then BE__CAT110_2BGT= 1.0; else BE__CAT110_2BGT= -1.0;
_xc= CAT111;
if upcase(_xc)= "LBI0" then BE__CAT111_lBI0= 1.0; else BE__CAT111_lBI0= -1.0;
_xc= CAT11;
if upcase(_xc)= "GLQPWFF" then BE__CAT11_glQpWFf= 1.0; else BE__CAT11_glQpWFf= -1.0;
```

Figure 6: An example of automatically generated binary encoding rules.

Dimension Reduction

Dimension Reduction via Feature Creation/Extraction

Feature extraction is a popular way to handle large numbers of variables, as the information from many variables is condensed into a smaller number of features. This focuses on creating new features, which are represented as combinations of the original variables.

We transformed the original set from the traditional sparse representation (i.e., a "regular" data set) into a dense coordinate set (COO) representation by changing every non-zero value in the regular data set into a row in the COO/transactional set as a row, column value tuple. This format is an excellent way to reformulate sparse data into a condensed format for machine learning.

Untransformed data matrix

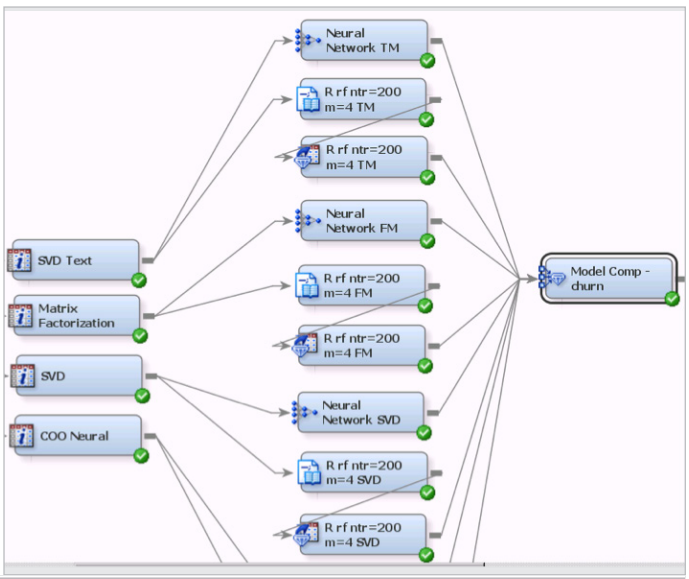
0	0	1
5	0	7
0	3	0

Transformed COO matrix representation

1	3	1
2	1	5
2	3	7
3	2	3

The first column is the row number of the original value, the second column is the column number of the original value and the third column is the original value.

We applied singular value decomposition (SVD), deep learning and factorization machines to create several different sets of features from the variables. (See Figure 7.) We evaluated the efficacy of the features by using them in a predictive model for the churn target. Even though we used high-capacity learning algorithms to do so (SAS neural networks and R random forest, via the SAS Enterprise Miner open-source modeling node), we found that the created features did not do well in predicting churn. The original contest winners had also tried feature extraction and had come to the same conclusion. We hypothesize that many of the 13,113 input variables were simply noise, and compressing them into a smaller number of features, along with the variables which did contain important information, simply diluted the predictive power of the useful variables. Because of this, we moved to variable selection.



Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Roc Index	Train: Total Degrees of Freedom	Train: Degrees of Freedom for Error	Train: Model Degrees of Freedom	Train: Number of Estimated Weights	Train: Akaike's Information Criterion
Y	Neural5	Neural5	Neural Net...	churn		0.598	40000	39299	701	701	21554.49
	Neural7	Neural7	Neural Net...	churn		0.592	40000	39324	676	676	21497.92
	Neural6	Neural6	Neural Net...	churn		0.583	40000	39324	676	676	21832.94
	Neural8	Neural8	Neural Net...	churn		0.555	40000	39324	676	676	22006.87
	MdlImp7	MdlImp7	R rf ntr=200...	churn		0.555	.	.	.	.	.

Figure 7: Machine-learning results using feature extraction and deep learning.

Dimension Reduction via Ensemble Variable Selection

After determining that feature extraction was not suitable for this problem, we needed to assess whether all 13,113 variables were useful for modeling and, if possible, reduce the number of variables. Several variable selection techniques would choose only a handful of variables as important.

To ensure that our variable selection would generalize well, we used an ensemble approach for variable selection. Instead of using the full data set, we balanced all of the churn events with a similar number of non-events chosen at random. We created 10 bagging samples, with replacement, using different seeds. This allowed the solution to generalize and not over-train on any given set of data.

We tried several supervised variable selection methods within each sample, including random forest, decision trees, least absolute shrinkage and selection operator (LASSO), and least angle regression. We chose least angle regression because it commonly performs well with tree ensembles. Least angle regression is an optimal routine that sequentially adds variables to the model, based upon their relationship with model residuals at each step. (See Figure 8.) This routine does not require multiple passes through the data, which allows it to run faster.

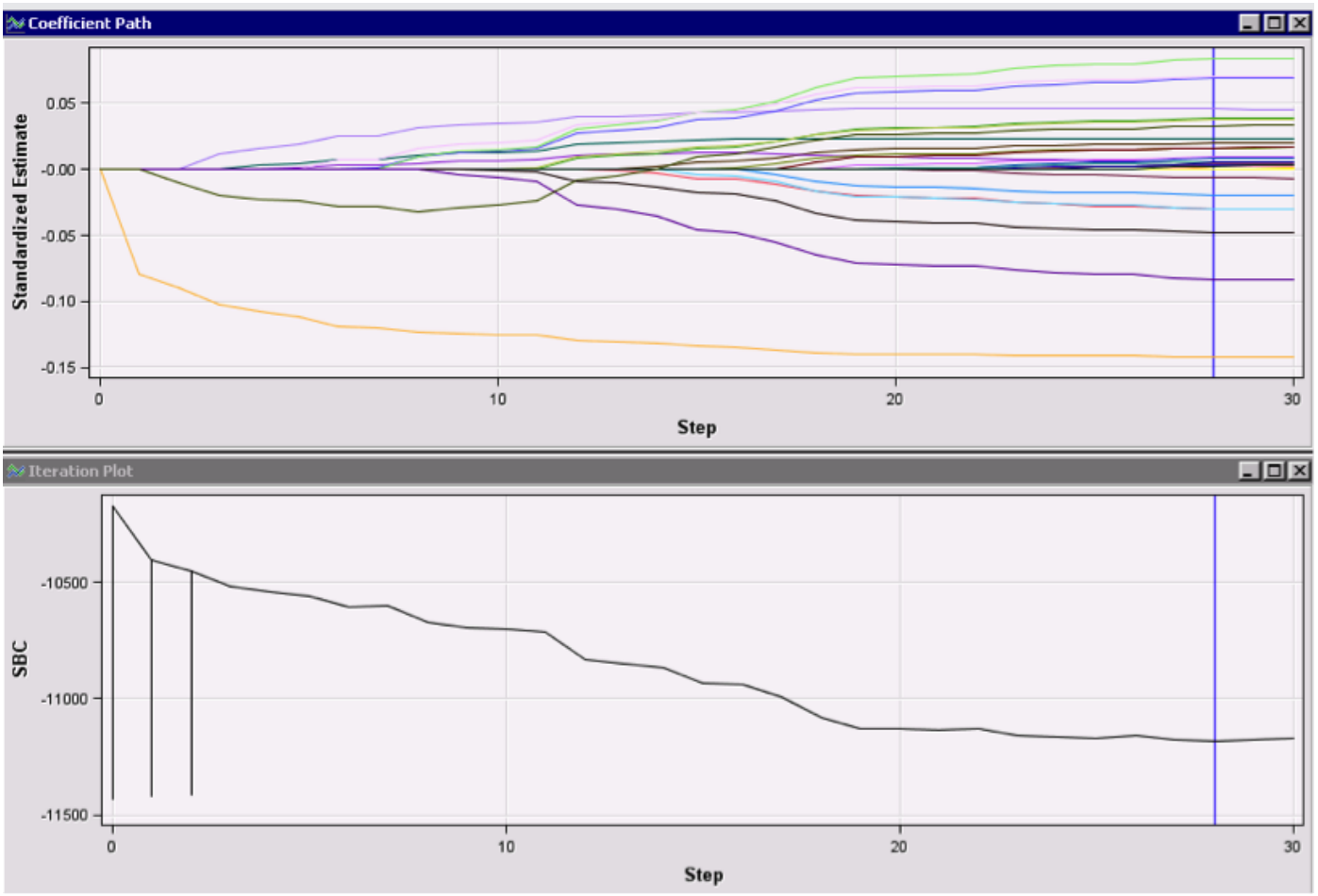


Figure 8: Variable selection using least angle regression.

We then combined the results of each variable selection into a unified set of variables. This ensemble approach (see Figure 9) is similar to bagging, but, instead of combining the predicted posterior probabilities, we ranked the number of times that a variable was selected from a balanced sample. Approximately 30 variables were selected two or more times out of 10 iterations and were kept for further modeling.

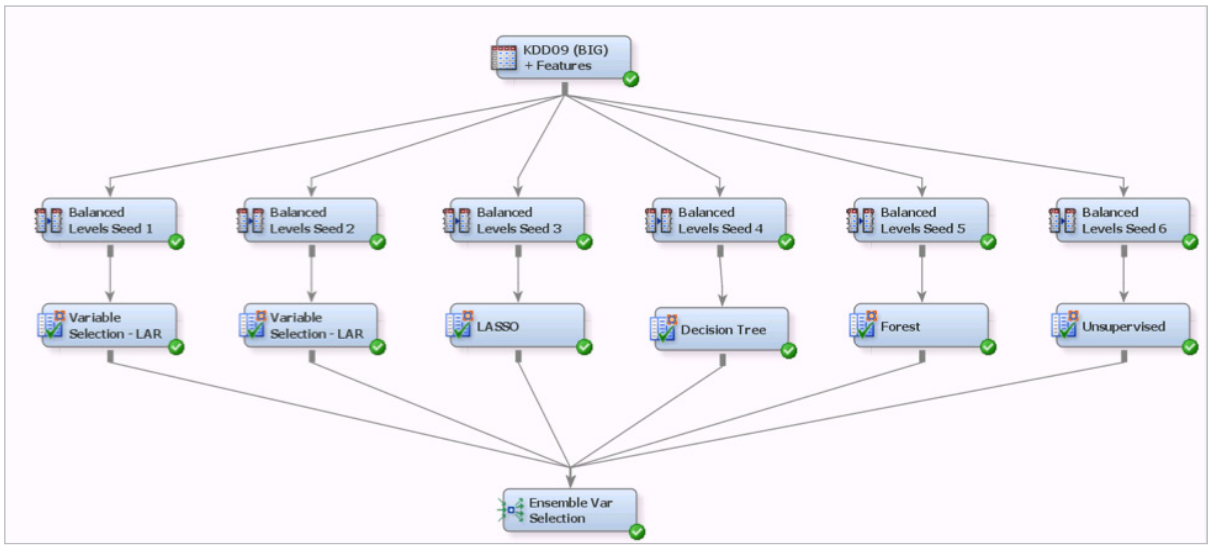


Figure 9: Ensemble variable selection.

We then assessed variable importance using the random forest algorithm in SAS Enterprise Miner. We relaxed the criteria of the forest, running through several hundred trees. We found that our optimally binned and standardized variables showed the greatest contribution to identifying churn.

Variable Importance

Variable Name	Number of Splitting Rules ▼	OOB: Gini Reduction	OOB: Margin Reduction	Label
OPT_var8981	1252	0.00139	0.00292	Transformed var8981
GRP_cat255	831	0.00038	0.00087	Grouped: cat255
OPT_var10533	698	0.00023	0.00053	Transformed var10533
STD_REP_var2570	502	0.00009	0.00030	Transformed: Replacement: var2570
OPT_var8032	457	0.00018	0.00042	Transformed var8032
STD_REP_var13663	441	0.00001	0.00016	Transformed: Replacement: var13663
STD_REP_var3104	408	0.00015	0.00042	Transformed: Replacement: var3104
OPT_var10256	390	0.00016	0.00042	Transformed var10256
STD_REP_var9075	334	0.00014	0.00038	Transformed: Replacement: var9075
STD_REP_var12125	292	-0.00001	0.00006	Transformed: Replacement: var12125
cat22	249	0.00003	0.00011	
BE__OPT_VAR647_02_29117_92_hig	243	0.00003	0.00008	
BE__OPT_VAR990_02_33_4847_high	234	0.00002	0.00007	
BE__OPT_VAR6255_02_105_65_high_	230	0.00002	0.00006	
BE__OPT_VAR8484_02_140_8876_hig	207	0.00001	0.00005	
BE__OPT_VAR3080_02_9_high__MISS	179	0.00012	0.00026	
GRP_cat166	179	0.00002	0.00007	Grouped: cat166
BE__OPT_VAR6049_02_4_5_high__MI	168	0.00009	0.00022	
BE__OPT_VAR5166_02_1_high__MISS	165	0.00011	0.00025	
STD_REP_var13100	157	0.00000	0.00004	Transformed: Replacement: var13100
GRP_cat183	140	0.00003	0.00011	Grouped: cat183
BE__OPT_VAR3624_02_7_5_high__MI	128	0.00003	0.00007	
BE__OPT_VAR13379_02_524_96_high	123	0.00000	0.00002	
STD_REP_var12381	121	0.00000	0.00003	Transformed: Replacement: var12381
STD_REP_var3886	119	-0.00001	0.00001	Transformed: Replacement: var3886
GRP_cat51	96	0.00001	0.00003	Grouped: cat51
BE__OPT_VAR1565_02_69_high__MIS	95	0.00005	0.00013	
STD_REP_var5444	60	0.00000	0.00002	Transformed: Replacement: var5444
GRP_cat164	45	-0.00001	-0.00005	Grouped: cat164
BE__VAR2652_5	2	-0.00000	0.00000	

Figure 10: Variable importance from ensemble variable selection.

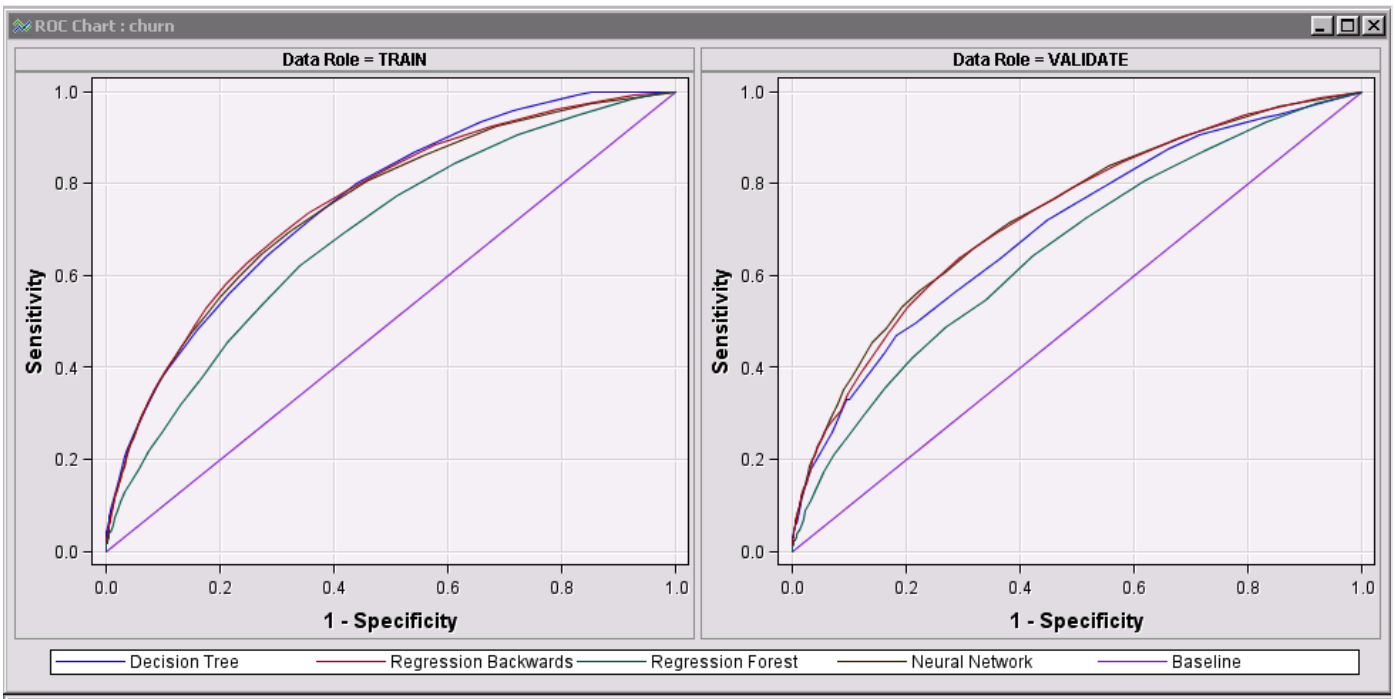
Modeling

We employed several approaches to predictive modeling. The common approach (standard data mining) would yield a satisfactory result (AUC = 0.733) and could be modeled in approximately 10 minutes.

Attempt 1: The "10-Minute Model"

The structure of the standardized, binned, binary encoded data suggested that a neural network might be a suitable model. First, we partitioned the data into 70 percent training and 30 percent validation partitions because it is imperative to validate models on a holdout data set. Often, models will over-train on the modeling partition and not generalize well on holdout data sets. We then ran least angle regression variable selection, which fed into neural network, decision tree and logistic regression (backward) models in our model "tournament." We also tried random forest variable selection, which fed into a logistic regression model.

The neural network model had a good ROC curve value of 0.733, compared to lower values for models like random forest, decision trees and regressions. Directionally, this would be a good model after 10 minutes, but our goal was to be even more accurate.



Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Roc Index	Train: Average Squared Error	Train: Divisor for ASE	Train: Maximum Absolute Error	Train: Sum of Frequencies	Train: Root Average Squared Error	Train: Sum of Squared Errors
Y	HPNNA	HPNNA	Neural Net...	churn		0.733	0.063001	70010	0.986914	35005	0.251001	4410.7
	HPReg2	HPReg2	Regression...	churn		0.728	0.062814	70010	0.996496	35005	0.250626	4397.5
	HPTree	HPTree	Decision Tr...	churn		0.698	0.06223	70010	0.992665	35005	0.24946	4356.7
	HPReg	HPReg	Regression...	churn		0.655	0.065662	70010	0.998208	35005	0.256247	4597.0

Figure 11: The 10-minute model results.

Attempt 2: Ensemble Models

We then tried several types of ensemble methods, which tend to generalize well. The best ensemble model was a bagged ensemble of boosted trees with a validation partition ROC of 0.853. The ensemble consisted of five balanced event to non-event samples, using different random number seeds to select non-churn observations. Within each sample, we portioned the data into 70 percent training and 30 percent validation. We created stratified samples to appropriately cross-validate our results.

Each "perturbed" boosted decision tree went through 10 iterations. After each iteration, misclassified events were given more weight in subsequent runs. We then combined the five boosted trees into an ensemble by averaging the predicted posterior probabilities to produce score code which we expected to generalize well with future data.

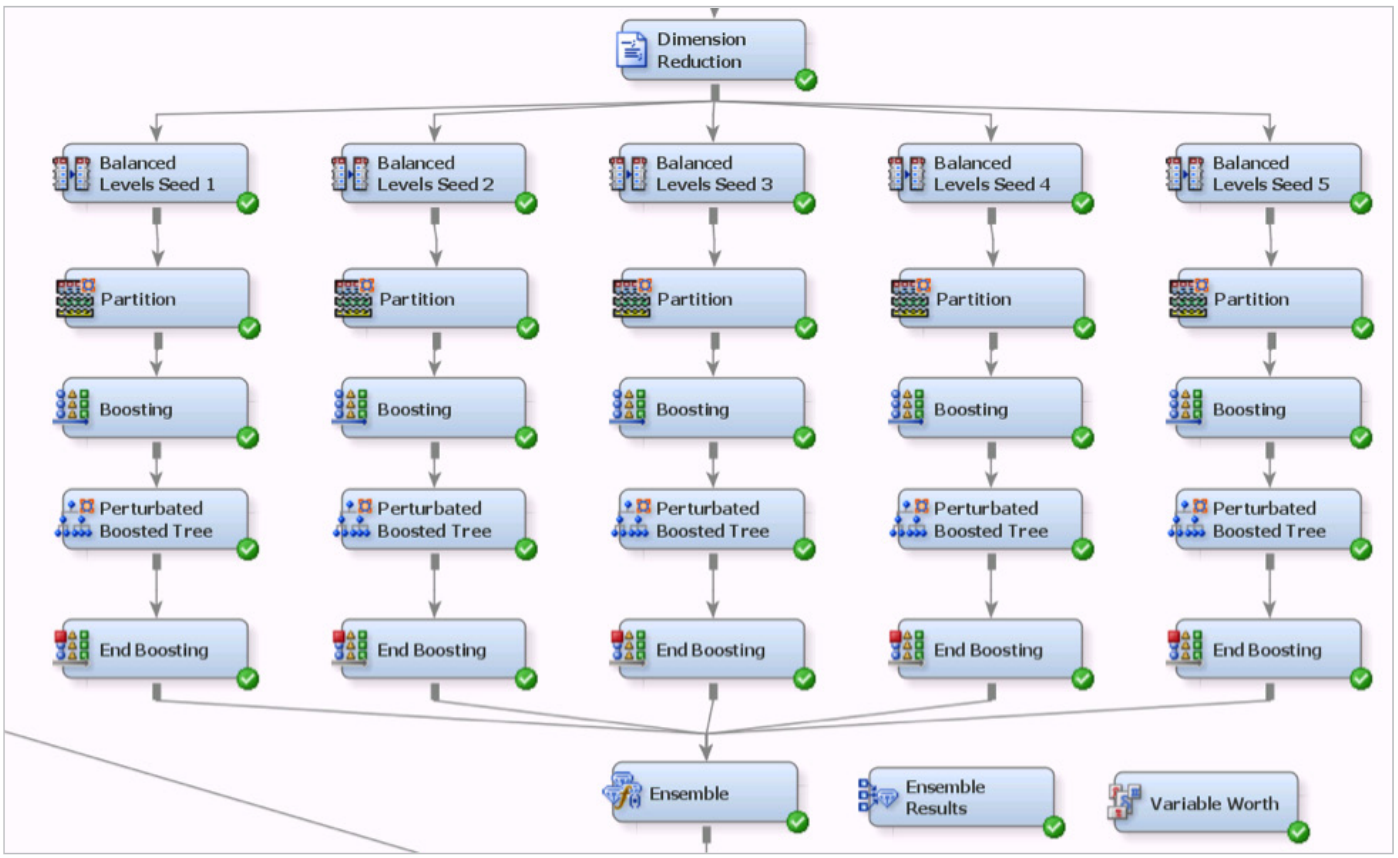


Figure 12: An ensemble perturbed tree model.

Attempt 3: SAS/IML® Code Model

SAS Enterprise Miner is very flexible for data scientists who also want to write models using SAS code. We used SAS/IML, which is an interactive matrix language suitable for managing sparse data sets. We ran a singular value decomposition to break the data into factors, which is a nice feature extraction method. We then fed the resulting nine factors into a logistic regression. The results did not beat the 10-minute model or the ensemble model.

```
data tr; /* CONVERT FROM VIEW TO DS */
   set &EM_IMPORT_DATA (keep= %EM_INTERVAL_INPUT);
run;

proc iml;

   /* CREATE SVD FEATURES FROM EM TRAINING DATA */
   use tr;
   read all var _ALL_ into x;
   call svd(u, q, v, x);

   /* CONVERT SVD FEATURES FROM IML MATRIX TO SAS DATASET */
   create tr_out from u;
   append from u;
   close tr_out;

quit;

data &EM_EXPORT_TRAIN &EM_EXPORT_VALIDATE;
    merge &EM_IMPORT_DATA tr_out;
    if ranuni(12345) < 0.3 then output &EM_EXPORT_TRAIN;
    else output &EM_EXPORT_VALIDATE;
run;
```

Figure 13: Sample SAS/IML SVD code inside SAS Enterprise Miner.

Attempt 4: Open Source

We tried a final approach by running an R model inside SAS Enterprise Miner, following the feature creation and dimension reduction steps we previously used. The Open Source Integration node allows users to insert R code into the SAS Enterprise Miner process flow, enabling SAS to automatically pass variables from prior steps and then assess the validity of the R model automatically. This enables experimentation with new/unproven R methods. Using the "&EMR" macro handles, all variables and results are automatically passed through a modeling process flow in the SAS Enterprise Miner diagram.

Code Editor

```
# Load Library
library(randomForest)

# Train Model
# Model uses variable handles for target and inputs, accessed from data handles (EM Data Source)
&EMR_MODEL <- randomForest(&EMR_CLASS_TARGET ~ &EMR_CLASS_INPUT + &EMR_NUM_INPUT, ntree= 250, mtry=4, maxnodes= 50, data= &EMR_IMPORT_DATA, importance= TRUE)

# Plot available in Train Graphs
png("EMR_forestMsePlot.png")
plot(&EMR_MODEL, main= 'randomForest MSE Plot')
dev.off()

# Export data to flow
&EMR_EXPORT_TRAIN <- predict(&EMR_MODEL, &EMR_IMPORT_DATA, type="prob")
&EMR_EXPORT_VALIDATE <- predict(&EMR_MODEL, &EMR_IMPORT_VALIDATE, type="prob")

# Results
&EMR_EXPORT_TRAIN[1:10,]

# Variable Importance
round(importance(&EMR_MODEL),2)
write.table(round(importance(&EMR_MODEL),2), file = "EMR_forestImportance.csv", sep=",", row.names = TRUE, col.names = TRUE)
```

Figure 14: Open-source R random forest model inside SAS Enterprise Miner.

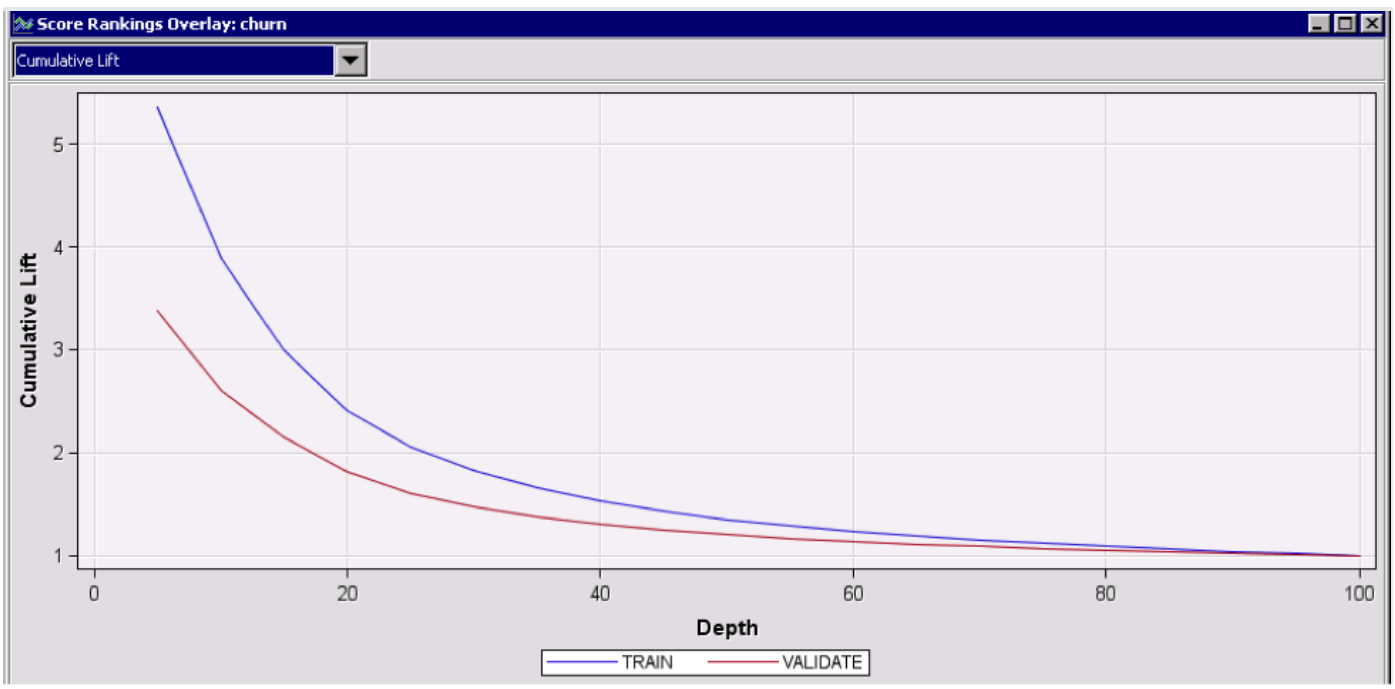
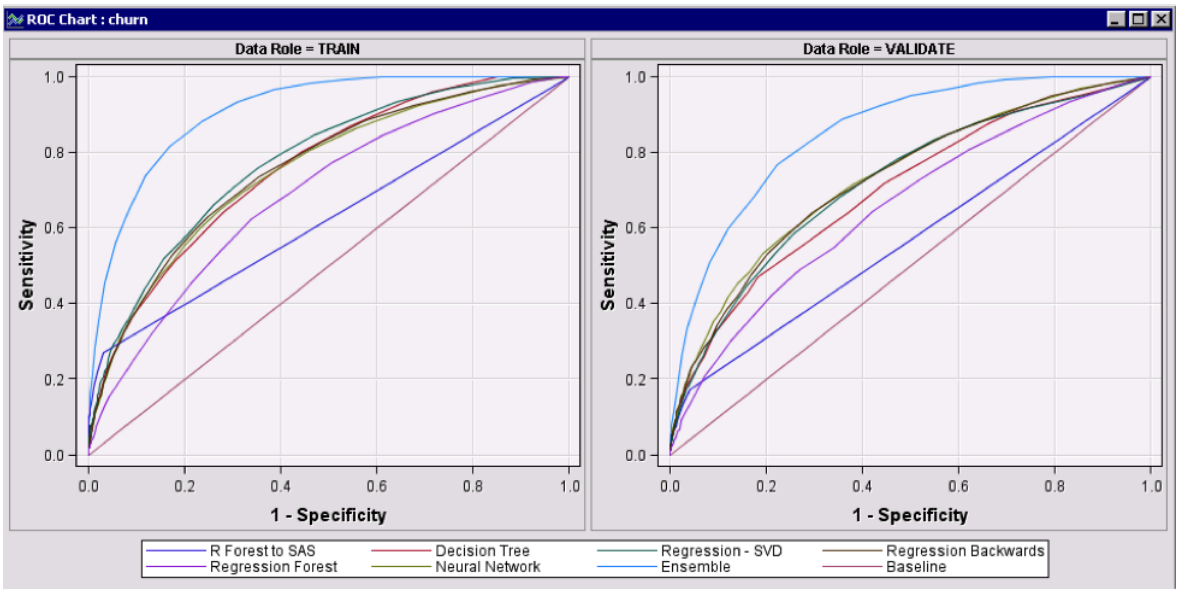


Figure 15: Automatically generated assessment results for the open-source R random forest model.

Final Model Selection and Scoring

We assessed all model results from the four tries in one model comparison. Using the required validation assessment criteria AUC/ROC, the ensemble model was the champion with an AUC of 0.853.



Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Roc Index	Train: Average Squared Error	Train: Divisor for ASE	Train: Maximum Absolute Error	Train: Sum of Frequencies	Train: Root Average Squared Error
Y	Ensmbl	Ensmbl	Ensemble	churn		0.853	0.210054	10284	0.61719	5142	0.458316
	HPNNA	HPNNA	Neural Net...	churn		0.733	0.063001	70010	0.986914	35005	0.251001
	HPReg2	HPReg2	Regression...	churn		0.728	0.062814	70010	0.996496	35005	0.250626
	HPReg3	HPReg3	Regression...	churn		0.718	0.06269	20440	0.994803	10220	0.250379
	HPTree	HPTree	Decision Tr...	churn		0.698	0.06223	70010	0.992665	35005	0.24946
	HPReg	HPReg	Regression...	churn		0.655	0.065662	70010	0.998208	35005	0.256247
	MdlImp	MdlImp	R Forest to ...	churn		0.599	0.070536	70010	1	35005	0.265586

Figure 16: Model assessment of the 10-minute model, ensemble model, SAS/IML SVD and open-source R random forest.

Scoring

For demonstration purposes, we scored the big input data set inside Hadoop. Assuming that in a production deployment, the scoring data set would live inside a large-scale environment capable of managing more than 15,000 variables, we moved the data to a Cloudera Hadoop cluster. Using SAS Scoring Accelerator for Hadoop, we published the final champion model to Hadoop with one mouse click (45,000 lines of score code), which automatically converted the SAS scoring code into SAS DS2 code. This enabled parallel scoring inside Hadoop, preserving 100 percent of the scoring code without sacrificing accuracy. There was no need for manual translation to Hadoop score code. This score code ran in less than a minute on the entire data set.

Conclusions

The SAS team was able to use the features and capabilities of SAS Enterprise Miner to conduct a machine-learning data mining analysis and deployment in a real world context. The team used an independent data source (KDD Cup data), with all of its missing values, and applied several data preparation, feature creation and dimension reduction techniques to get the data ready for modeling.

Then, they used several machine-learning approaches, sometimes in series and/or ensembles, to iteratively create an accurate predictive model. These included a basic data mining approach (the 10-minute model), an ensemble, a SAS/IML model and an open-source R model. As noted above, using the required validation assessment criteria (AUC/ROC), the ensemble model was the champion model, with an AUC of 0.853.

Finally, to replicate a common real-world model deployment scenario, the SAS team loaded the original data into a distributed Cloudera Hadoop cluster, automatically generated 45,000 lines of SAS score code for the champion ensemble model, published the champion model to Hadoop with one mouse click and scored the data in the distributed Hadoop cluster in less than a minute.

This exercise illustrates many of the benefits of using SAS Enterprise Miner for machine-learning data mining:

- High predictive accuracy.
- Fast processing.
- Completeness of capabilities.
- Timely results.
- Increased team productivity.
- Integration with other SAS software and external models.
- Convenient deployment into production systems.

Learn More

For more details on the extensive predictive modeling features in SAS Enterprise Miner, visit sas.com/enterpriseminer.

To contact your local SAS office, please visit: sas.com/offices

107521_S133184.0215