

Machine Learning and Predictive Analytics in SAS® Enterprise Miner™ and SAS/STAT® Software

D. Richard Cutler, Utah State University

ABSTRACT

SAS/STAT® software and SAS® Enterprise Miner™ are two excellent environments for applying machine learning and other analytical procedures to a wide range of problems, from small data sets to the very large and very wide. In SAS Enterprise Miner, one can move seamlessly from data cleaning and processing, through preliminary analyses and modeling, to comparisons of predictive accuracy of several predictive methods and the scoring of new data sets. Many powerful machine learning and statistical learning tools including gradient boosting machines, artificial neural networks, and decision trees are nodes in SAS Enterprise Miner, and other SAS® procedures that are not nodes can be accessed through the SAS Code Node. In this talk, I work through some examples from business and other areas to illustrate some of the many capabilities of SAS/STAT and SAS Enterprise Miner.

INTRODUCTION

Modern statistical analyses frequently entail the use of so called *machine learning* or *statistical learning* methods, which are highly computational algorithms for prediction and interpretation of data. SAS® has many such procedures implemented including classification and regression trees (Breiman et al. 1984) in PROC HPSPLIT, support vector machines (Cortes and Vapnik 1995) in PROC HPSVM, random forests (Breiman 2001) in PROC HPFOREST and gradient boosting machines (Friedman 2001) in SAS Viya and SAS Enterprise Miner. These methods may be used in regular SAS programming, and are available in SAS OnDemand. SAS Enterprise Miner offers an alternative environment for conducting statistical analyses, from the simple graphical summaries of data to classical methods such as multiple linear regression and logistic regression through all of the machine learning methods listed above and more. The graphical user interface of Enterprise Miner facilitates multi-step analyses without the need to remember or look up or remember programming syntax, and allows for very simple comparison of the accuracies of multiple methods through the Model Comparison node. In this paper and the associated talk I will contrast similar analyses of real data in the usual SAS programming environment and in SAS Enterprise Miner.

INSURANCE CLAIMS EXAMPLE

The data for the first set of analyses concerns auto insurance claims in Germany. The purpose of the analyses is to relate the probability of a claim, and the amount of the claims, to a variety of predictor variables which include the age of the policy holder, the number of children in the household that are driving, and socioeconomic information including income, house value, type of job, and where the policy holder lives.

PRELIMINARY DATA MANIPULATIONS AND SAMPLING

A brief description of the variables in the dataset are given in Table 1. There are 10,300 observations in the dataset. The response variable are CLAIM_IND, which is coded 1 if an insurance claim was made and 0 otherwise, and CLAIM_AMOUNT, which is the amount of the claim if a claim was made and 0 otherwise.

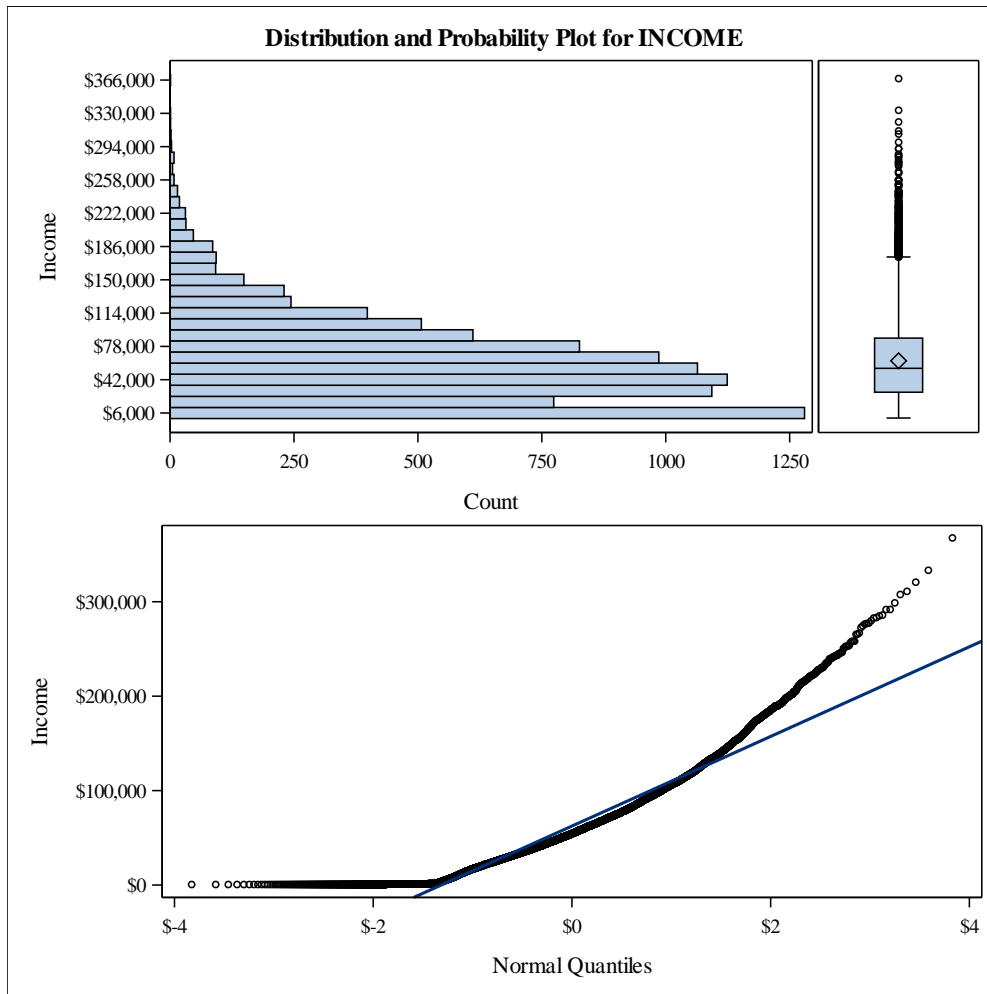
Table 1. Descriptions of Variables in Auto Claims Dataset

Variable Name	Type	Variable Description
AGE	Num	Age
AREA	Char	Home/Work Area
CAR_USE	Char	Car use (y/n)
CHILDREN	Num	Number of children
CHILD_DRIV	Num	Number of children driving
CLAIM_AMOUNT	Num	Claim Amount
CLAIM_IND	Num	Claim indicator (1 = claim made; 0 = No claim)
CLM_FREQ	Num	Number of claims in last 10yrs
DISTANCE	Num	Distance to work
EDU_LEVEL	Char	Highest level of education
GENDER	Char	Gender of driver
HOUSE_VAL	Num	Value of house
ID	Char	ID number
INCOME	Num	Income
JOB	Char	Occupation of driver
MVR_PTS	Num	Record points
REVOKED	Char	License revoked in last 10 years? (Yes, No)
STATE_CODE	Char	
STATUS	Char	Marital status (Yes = Married; No = Not married)
VEHICLE_AGE	Num	Age of vehicle
VEHICLE_TYPE	Char	Type of vehicle
VEHICLE_VAL	Num	Value of vehicle
YOJ	Num	Years on job

Variables such as INCOME, CLAIM_AMOUNT, and HOUSE_VAL are often right skewed so I carried out some preliminary graphical summaries of these and other variables. All three variables were indeed right-skewed. Figure 1 is a panel of summary plots for INCOME obtained in PROC UNIVARIATE. The boxplot, normal quantile plot, and histogram all show skewness. Accordingly, log transformations were applied to all three variables with an offset of 1. That is,

$$\text{LOG_INCOME} = \log(1 + \text{INCOME}).$$

Figure 1. Graphical Summaries of the Distribution of the Variable INCOME



Preliminary data summaries also revealed that several of the interval-valued variables contained 500-600 missing values. At this point the data analyst has the choice of simply omitting the observations that have missing values on one or more of these variables, which is satisfactory is the total number of missing values is small relative to the total sample size, or imputing the missing observations. For these analyses I chose to impute the missing data using the very simplest alternative, the mean of the non-missing values of each variable, using PROC HPIMPUTE. The code I used is given below:

```

title2 "Imputing Missing Values in YOJ, LOG_HOUSE_VAL and LOG_INCOME";
proc hpimpute data=SASGF.claim2 out=SASGF.claim3;
  input YoJ log_income log_house_val vehicle_age;
  impute YoJ / method = mean;
  impute log_income / method = mean;
  impute log_house_val / method = mean;
  impute vehicle_age / method = mean;
  ID Claim_Ind Area Edu_Level Gender Job Revoked State_Code Vehicle_Type
  Status Log_Claim_Amt Age Children Child_Driv Clm_Freq Distance
  Vehicle_val;
run;

```

A summary of the results of the imputation is given in Table 2.

Table 2. Summary of Imputation of Four Variables in Insurance Claims Dataset

Imputation Results					
Variable	Imputation Indicator	Imputed Variable	N Missing	Type of Imputation	Imputation Value (Seed)
YOJ	M_YOJ	IM_YOJ	548	Mean	13.47395
LOG_INCOME	M_LOG_INCOME	IM_LOG_INCOME	570	Mean	10.52021
LOG_HOUSE_VAL	M_LOG_HOUSE_VAL	IM_LOG_HOUSE_VAL	575	Mean	12.70385
VEHICLE_AGE	M_VEHICLE_AGE	IM_VEHICLE_AGE	639	Mean	10.29790

The imputed variable has the prefix **IM_** and, for each variable, a new variable indicating which observations have been imputed is generated and carries the prefix **M_**.

The ID statement in the code above tells SAS which variables in the input dataset that are not being imputed to add to the output dataset. Failure to put in this line of code will result in the output dataset, SASGF.claim3, containing only the new imputed variables and the indicator variables for which observations were imputed.

For the purposes of evaluating and comparing different predictive methods it is very desirable to have a completely separate test dataset. I used PROC HPIMPUTE to randomly partition the original dataset into a training dataset (SASGF.claimTrain) with 67% ($n = 6,901$) of the observations in the original dataset and a test dataset (SASGF.claimTest) with the remaining 33% ($n = 3,399$) observations, using the following piece of SAS code:

```

title2 "Partitioning the Dataset into Training and Test Pieces";
proc hpsample data=SASGF.claim3 out=SASGF.claim4 partition seed=4571
    partition sampopt=33;
    class Claim_Ind Area Edu_Level Gender Job Revoked State_Code
        Vehicle_Type Status;
    var Log_Claim_Amt Age Children Child_Driv Clm_Freq Distance
        IM_Log_House_val IM_Log_Income IM_Vehicle_Age Vehicle_val IM_YoJ;
run;

data SASGF.claimTrain SASGF.claimTest;
    set SASGF.claim4;
    if _PARTIND_ eq 0 then output SASGF.claimTrain;
    else if _PARTIND_ eq 1 then output SASGF.claimTest;
run;

```

Note that I set the value of the seed for the random number generator so I can replicate the partitioning if I ever need to.

LOGISTIC REGRESSION FOR PREDICTION

An obvious simple first analysis for these data is logistic regression which I carried out in PROC LOGISTIC with and without variable selection. Code for the logistic regression with variable selection is follows:

```
title2 "Logistic Regression for Claim_Ind with Backward Elimination";
proc logistic data=SASGF.claimTrain descending;
  class Area Edu_Level Gender Job Revoked State_Code Vehicle_Type Status;
  model Claim_Ind = Area Edu_Level Gender Job Revoked State_Code
    Vehicle_Type Age Children Child_Driv Clm_Freq Distance
    IM_log_House_val IM_log_Income IM_Vehicle_Age Vehicle_val Status
    IM_YoJ / selection = b sls = 0.01 ctable pprob = 0.3 0.5;
  roc;
  score data = SASGF.claimTest out = claimTestscore2;
run;

title3 "Accuracy on Scored Dataset--Cutoff = 0.5";
proc freq data=claimTestscore2;
  tables Claim_Ind * I_Claim_Ind / nocol;
run;

title3 "Accuracy on Scored Dataset--Cutoff = 0.3";
data claimTestscore2;
  set claimTestscore2;
  PredCut3 = 0.0;
  if P_1 ge 0.3 then PredCut3 = 1;
run;

proc freq data=claimTestscore2;
  tables Claim_Ind * PredCut3 / nocol;
run;
```

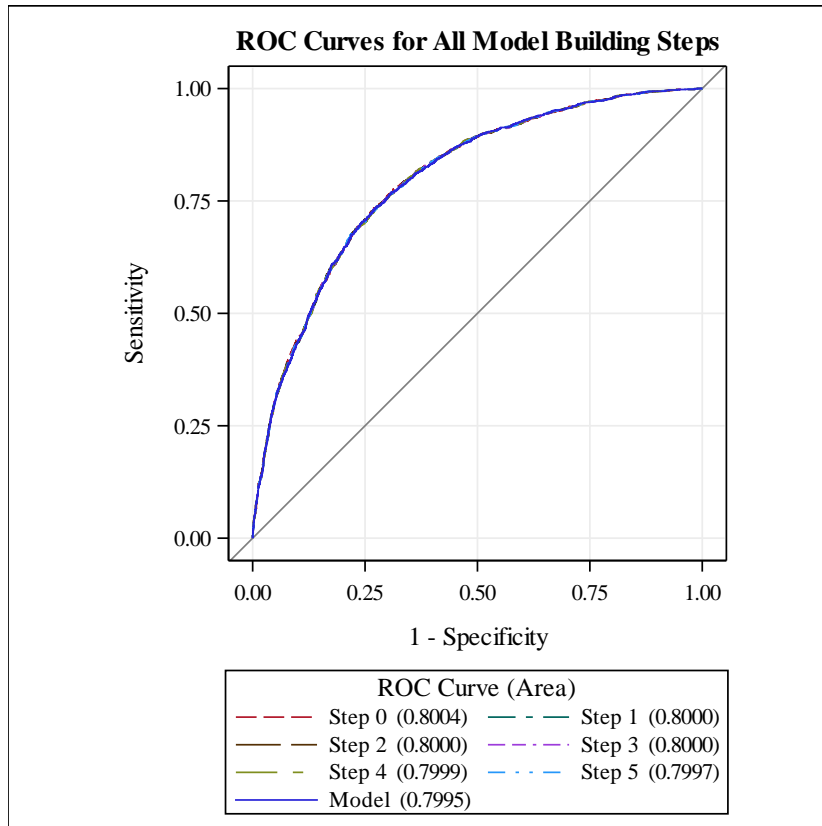
The descending option is needed in the call to PROC LOGISTIC because the response is coded as 0 or 1, with 1 indicating that a claim was made. The roc statement generates roc curves for each model fit, and these may be used to visually summarize the predictive accuracy. The predicted values from PROC LOGISTIC are a form of leave-one-out (LOO) cross-validated predictions. The score statement takes the predictive model fit to the training data and applies it to the test data. In the model statement backward elimination is selected as the variable selection method with a significance level to stay of 0.01. In this analysis some variables are clearly associated with CLAIM_IND and some are not so using values of the significance level to stay of between 0.05 and 0.001 all yield the same final model. The ctable option in the model statement causes SAS to give accuracy metric such as percent correct, sensitivity, specificity, false positive and false negative rates for the classification. In addition to the usual 0.5 cutoff I have specified a cutoff of 0.3. The reason for this is that there are many more non-claims (code 0 on CLAIM_IND) than claims (code 1 on CLAIM_IND) and in such cases with a cutoff of 0.5 the specificity may be high but the sensitivity is often very low, below 50%. The results for the logistic regression analyses are summarized in Table 3.

Table 3. Classification Accuracies for Logistic Regression with and without Variable Selection, on Test Data and Cross-validated on the Training Data Using Cutoffs of 0.5 and 0.3

Variable Selection	Probability Cut-off	Crossvalidation Or Test Data	Percent Correct	Specificity	Sensitivity
No	0.5	CV	77.5%	92.0%	38.1%
		Test	77.5%	91.7%	37.8%
	0.3	CV	73.2%	74.1%	70.6%
		Test	72.7%	74.3%	68.1%
Yes	0.5	CV	77.3%	91.8%	38.0%
		Test	77.8%	92.1%	37.2%
	0.3	CV	73.4%	74.3%	71.0%
		Test	72.7%	74.3%	68.1%

Note that the cross-validated accuracies and the accuracies measured on the test dataset are almost identical. With a cut-off of 0.5 the specificity was over 90% and the sensitivity was less than 40% with an overall percent correct of about 77.5%. Changing the cut-off to 0.3 led to values of the sensitivity and specificity that were much more similar (sensitivity between 68% and 71% and specificity of about 74%) and a slightly lower overall percent correct of 72.7%—73.4%. Finally, with regard to variable selection, six variables were removed: STATE_CODE, IM_VEHICLE_AGE, IM_YOJ, GENDER, AGE and IM_LOG_HOUSE_VAL.

Figure 2. ROC Curves for Variable Selection by Backward Elimination



The overlaid ROC curves for the original model and the models with these variables removed one at a time (Figure 2) are virtually indistinguishable from each other and the difference in AUC values between the model with all the variables and the model with six fewer variables is only 0.0009, an absolutely miniscule difference by any standard. Together the overlaid ROC curves and the AUC values indicate that the predictive accuracy of the logistic regression model was not compromised in any way with the elimination of the six variables. This is also reflected in the metrics in Table 2: looking at the accuracies for the logistic regressions with and without variable selection we see that there is almost no difference in any of the metrics.

DECISION TREES

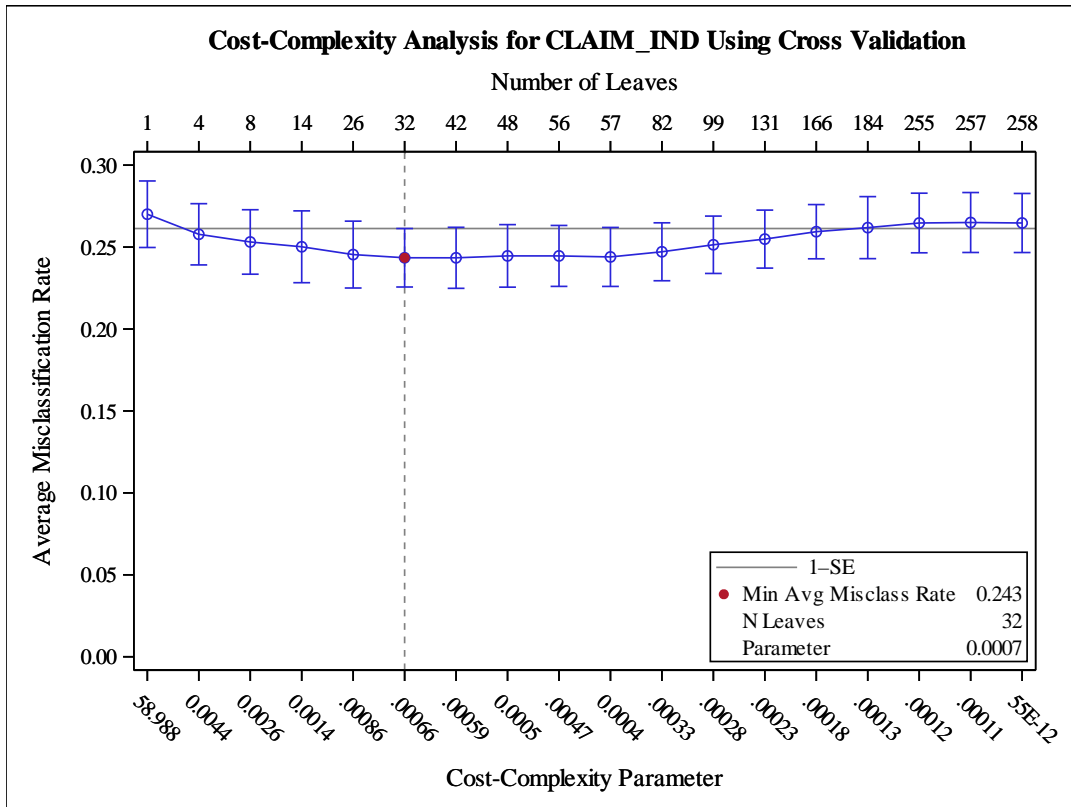
Classification and regression trees (Breiman et al. 1984), also known as decision trees, are an attractive alternative to regression and logistic regression often resulting in very concise summaries of the data with just a few binary partitions and similar accuracy to regression models with a dozen or more predictor variables. Classification trees were among the first machine learning methods to find widespread use among statisticians and are also the basis for some of the most accurate classification procedures, notable gradient boosting machines (Friedman 2001) and random forests (Breiman 2001).

A very first step in predicting using a decision tree is to select the appropriate size of the tree. Decision trees are fit in SAS using the HPSPLIT procedure. The key diagnostic for selecting tree size is the plot of the cross-validated accuracy of the tree against tree size (the number of *terminal nodes* or *leaves*) which is denoted by CVCC in the SAS code that follows. Cross-validation entails *randomly* splitting the dataset into 10 equal size pieces. By selecting the value of random seed I am able to ensure that I get the same partitioning of the data each time and can replicate my results. As is the case in many SAS regression-like procedures, categorical variables are put in a class statement. The difference between PROC HPSPLIT and, say, PROC LOGISTIC, is that if the response variable is categorical then it also goes in the class statement. Thus, CLAIM_IND is the first variable in the class statement and the response variable in the models statement. Last, we must specify a metric for measuring the purity of the subgroups of data generating by the binary partitioning process. I have chosen the Gini index, which is the most commonly used metric but other metrics such as entropy are available. The following is the SAS code I used to get the CVCC plot to help me to determine candidate tree sizes:

```
title2 "Fitting a Classification Tree";
title3 "Determining the Size of the Tree";
proc hpsplit data=SASGF.claimTrain cvmethod=random(10) seed=456 cvmodel=fit
    plots(only)=cvcc;
    class Claim_Ind Area Edu_Level Gender Job Revoked State_Code
        Vehicle_Type Status;
    model Claim_Ind = Area Edu_Level Gender Job Revoked State_Code
        Vehicle_Type Age Children Child_Driv Clm_Freq Distance
        IM_log_House_val IM_log_Income IM_Vehicle_Age Vehicle_val Status
        M_YoJ;
    grow gini;
run;
```

From the output we obtain the plot of cross-validated accuracy against tree size. The highest accuracy (which is the minimum misclassification rate) occurs for a tree with 32 leaves (terminal nodes). The 1-SE rule of Breiman et al. (1984) suggests a much smaller tree with four terminal nodes. That is, the original (training data) of over 6,000 observations is divided into just 4 subgroups. For further analyses I decided to fit classification trees with 4, 14, and 32 leaves and to compare their accuracies.

Figure 3. Plot of Cross-validated accuracy against Tree Size



The following code is for fitting a tree with four leaves and scoring the test dataset:

```

title3 "Fitting a Tree with 4 Leaves and Predicting onto the Test Data";
proc hpsplit data=SASGF.claimTrain cvmethod=random(10) seed=456
      cvmodelfit plots;
class Claim_Ind Area Edu_Level Gender Job Revoked State_Code
      Vehicle_Type Status;
model Claim_Ind = Area Edu_Level Gender Job Revoked State_Code
      Vehicle_Type Age Children Child_Driv Clm_Freq Distance
      IM_log_House_val IM_log_Income IM_Vehicle_Age Vehicle_val
      Status IM_YoJ;
grow gini;
prune costcomplexity (leaves=4);
code file='C:\Users\Richard\Documents\Research\Reviews\SAS\SAS Global
Forum 2019\Fournodes.sas';
run;

data claimTestpred(keep=Actual Predicted);
set SASGF.claimTest end=eof;
%include "C:\Users\Richard\Documents\Research\Reviews\SAS\SAS Global
Forum 2019\Fournodes.sas";
Actual = Claim_Ind;
Predicted = (P_Claim_Ind1 >= 0.5);
run;
proc freq data=ClaimTestpred;
tables Actual*Predicted / nocol;
run;

```


In the prune statement the size of the tree to be fitted is specified. The code statement outputs SAS data step code for scoring a new dataset and the data step that follows imports those commands and scores the test dataset. In this example I used a cut-off of 0.5 but it would be very simple to modify to obtain a cutoff of 0.3 if so desired.

Table 4 is a modification of Table 3 to include the cross-validated and test data classification accuracies for classification trees with 4, 14, and 32 leaves.

Table 4. Classification Accuracies for Logistic Regression with and without Variable Selection, on Test Data and Cross-validated on the Training Data

Predictive Method	Crossvalidation Or Test Data	Percent Correct	Specificity	Sensitivity
Logistic Regression No Variable Selection	CV	77.5%	92.0%	38.1%
	Test	77.5%	91.7%	37.8%
Logistic Regression Variable Selection	CV	77.3%	91.8%	38.0%
	Test	77.8%	92.1%	37.2%
Tree with 4 leaves	CV	74.8%	97.2%	14.1%
	Test	75.5%	97.3%	13.1%
Tree with 14 leaves	CV	74.9%	90.0%	34.0%
	Test	75.9%	89.8%	36.0%
Tree with 32 leaves	CV	75.6%	90.0%	36.6%
	Test	76.8%	91.6%	34.3%

The cross-validated accuracies and the predictive accuracies obtained on the test data are very similar with overall percent correctly classified being just a tiny bit higher than the cross-validated estimates. This is unusual, but the differences are so small they may be down to randomness in the cross-validation partitioning of the training data into 10 pieces. The overall predictive accuracy of the classification trees increases very slightly with the number of terminal nodes, from 75.5% for the tree with only four terminal nodes to 76.8% for the tree with 32 nodes. The accuracy for this largest tree is essentially the same as for the logistic regression models. Usually interpretation of small trees is easiest but in this case it is complicated by the fact that the sensitivity for the tree with four leaves is only 13%—14%, which is too poor to be of practical value. For the two larger trees the values of sensitivity are much closer to those for the logistic regression models.

RANDOM FORESTS

In random forests (Breiman 2001) many (typically 100—500) subsets of the data are randomly drawn and decision trees fitted to each random subset of the data. The predictions are then combined to give more accurate predictions. In many applications (see, for example, Cutler et al. 2007) random forests has been found to be among the most accurate classifiers.

In SAS random forests may be fit using the HPFOREST procedure. The default proportion of observations in the original dataset that are selected in each sample is 0.6 and this may be changed using the inbagfraction option. The number of trees to be fit may also be specified with the maxtrees option. The default is 100; I have chosen 200 in the code below; and other versions of random forests use 500 or more trees. Several authors (e.g., Cutler et al. 2007) have found that the accuracy of random forests is

remarkably robust to the number of fitted trees and that as few as 50 trees will yield very accurate predictions. In the algorithm predictions from a given tree are made only for the observations that were not part of the dataset to which that tree was fit. Such observations are said to be out-of-bag (oob) with respect to the tree (and the dataset to which it was fit). The strange looking option `scoreprole=oob` is requesting the accuracy results for predictions generate only through combining the out-of-bag predictions. This is generally equivalent to cross-validation for other procedures.

```

title2 "Random Forests for Claim Indicator";
proc hpforest data=SASGF.claim4 maxtrees=200 scoreprole=oob
    inbagfraction=0.6;
input Area Edu_Level Gender Job Revoked State_Code Vehicle_Type Status
    / level = nominal;
input Age Children Child_Driv Clm_Freq Distance IM_log_House_val
    IM_log_Income IM_Vehicle_Age Vehicle_val IM_YoJ
    / level = interval;
target Claim_Ind / level=nominal;
run;

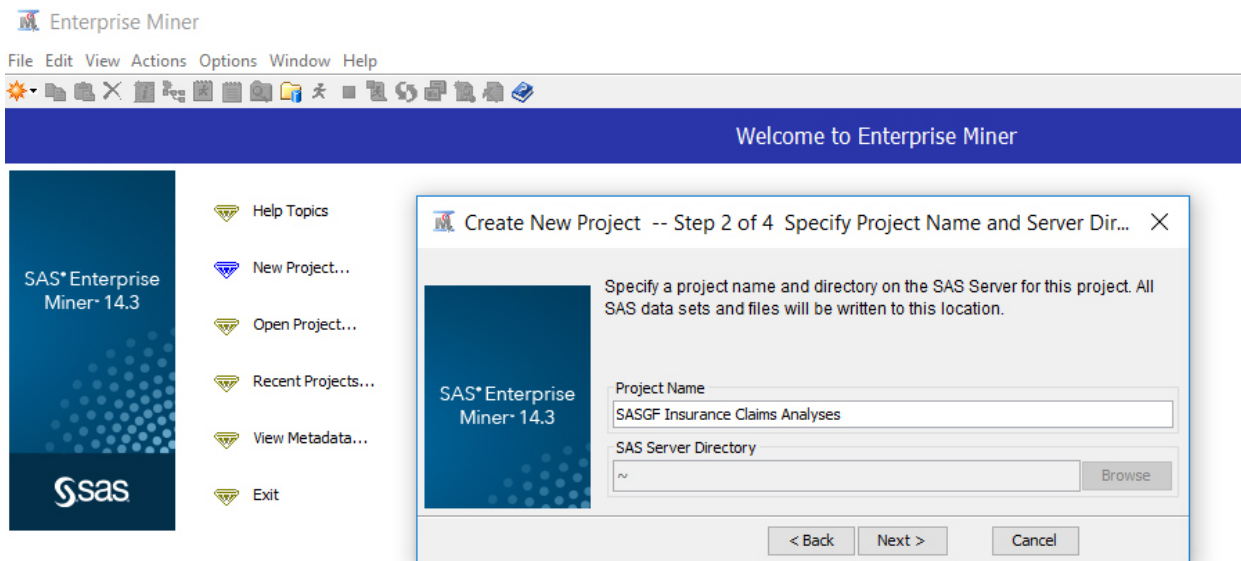
```

The out-of-bag accuracy in this example is 77.0% which is comparable to—but greater than—the accuracies obtained by logistic regression. In this case we would probably prefer logistic regression and classification trees over random forests due to their ease of interpretation.

USING SAS ENTERPRISE MINER TO ANALYZE THE INSURANCE CLAIMS DATA

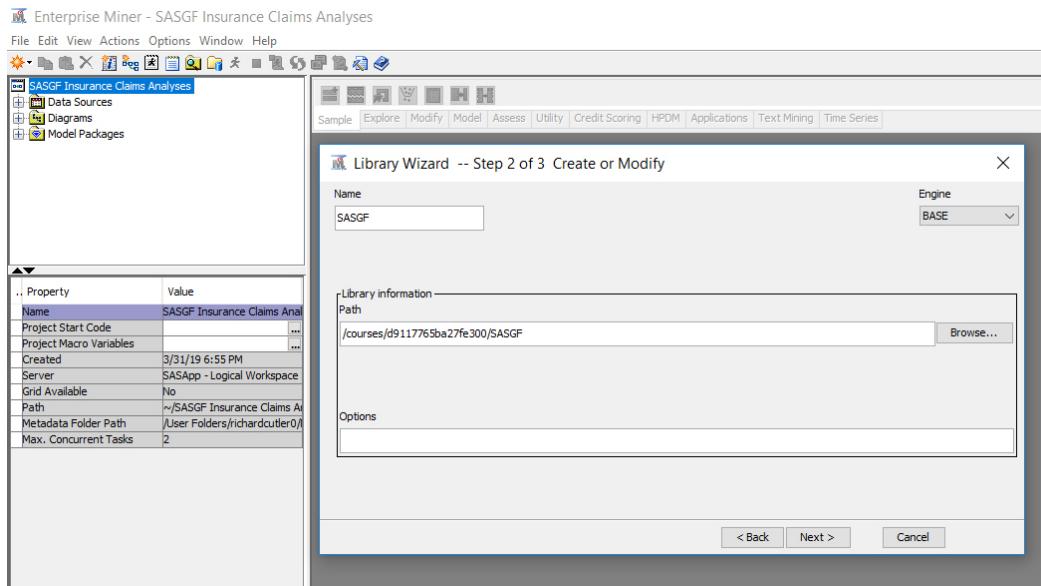
In this section I show how all the preceding analyses and so much more may be carried out in SAS Enterprise Miner. A first step is to create a new project in SAS Enterprise Miner. Clicking on the New Project option in the left hand menu brings up a *New Project Wizard* (Display 1). I have given the name *SASGF Insurance Claims Analyses* to the new project.

Display 1. Creating a New Project in SAS Enterprise Miner



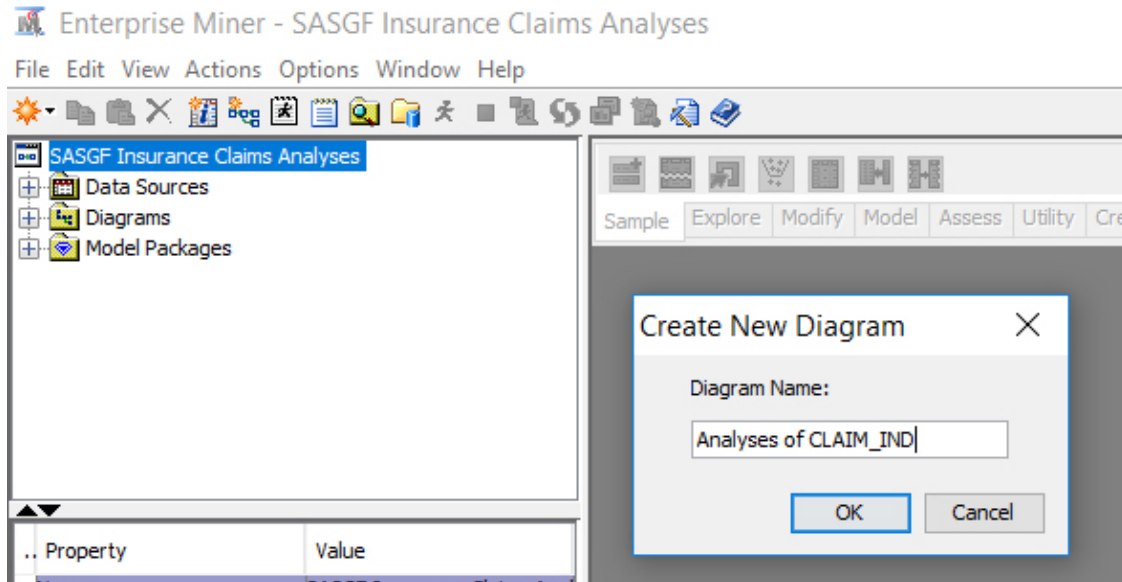
Next, from within the project I created a library for the data and other items relating to these analyses. Clicking on *File ... New ... Library* brings up the *Library Wizard* (see Display 2). Within the box shown in Display 2 I specify a path to a directory I have previously set up in my SAS Studio account within SAS OnDemand.

Display 2. Creating a New Library for the Project



The final piece of the setup in SAS Enterprise Miner is to create a diagram, the graphical workspace onto which we are going to place analysis *nodes* of various kinds. This, too, is accomplished by clicking on *File ... New ... Diagram*, which pops up a *New Diagram Wizard*. See Display 3. I have called this diagram *Analyses of CLAIM_IND*.

Display 3. Creating a New Diagram for the Project



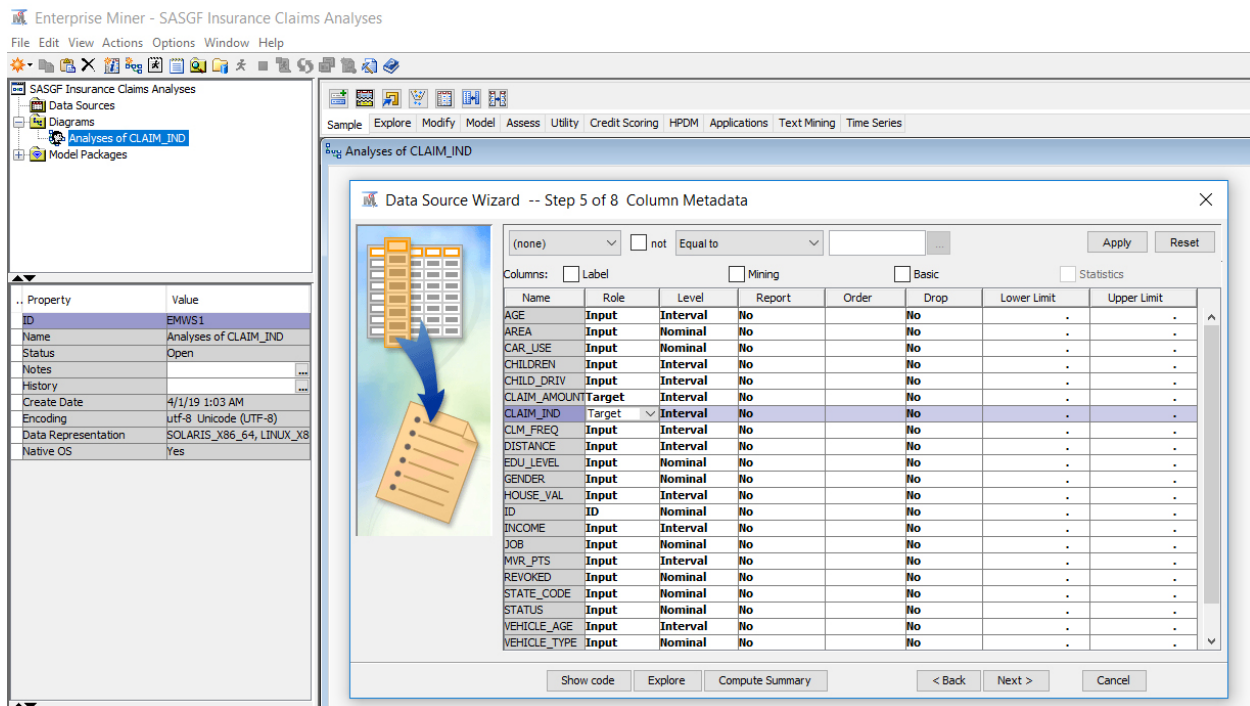
PRELIMINARY DATA MANIPULATIONS AND SAMPLING

I previously placed the insurance claims dataset in the directory for these analyses in SAS Enterprise Miner. To use it I just and dropped it into the diagram (workspace) and that brought up the *Data Source Wizard*. One of the most important steps in the wizard is step 5, which is where one can specify formats and roles for variables. In the column labelled **Role** one variable is labelled as an *ID* variable and is not used in subsequent analyses, and all the remaining variables are labelled *Input*. To specify that CLAIM_IND and CLAIM_AMOUNT should be regarded as response variables, I change their roles to *Target*.

In any given analyses one can typically only have a single response/target variable so you may have to specify which variable to use as the response for the specific analysis. I found that CLAIM_IND was the default response for my analyses, and that's what I wanted.

The second change I made was in the column labelled **Level**. Variables that are obviously categorical (because they contain letters) have level *Nominal*; all others have level *Interval*. The variable CLAIM_IND is coded as 0 and 1 so by default it is level *Interval*. In subsequent analyses regressions would be carried out, when we really want binary classifications. To remedy this situation I changed the level of CLAIM_IND from *Interval* to *Binary*.

Display 4. Identifying a Data Source for the Project

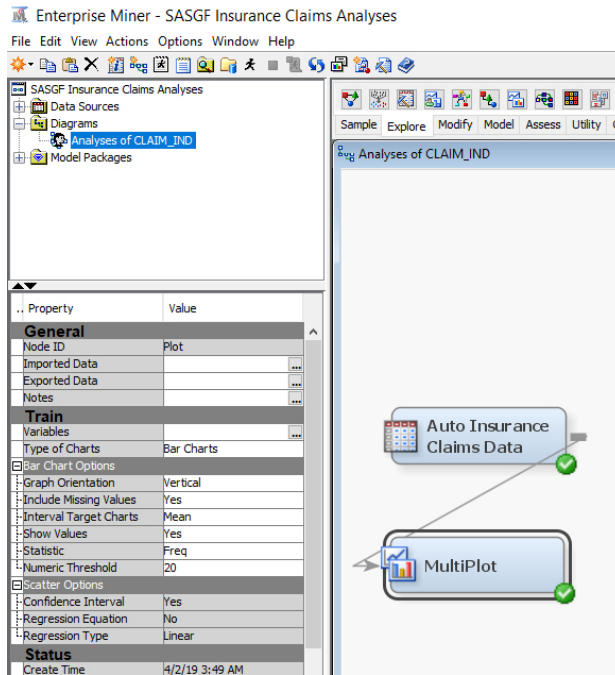


The screenshot shows the SAS Enterprise Miner interface. The main window is titled "Analyses of CLAIM_IND" and displays the "Data Source Wizard -- Step 5 of 8 Column Metadata". The wizard is showing a list of variables with their roles and levels. The variable CLAIM_IND is highlighted, and its role is set to "Target" and its level is set to "Interval".

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
AGE	Input	Interval	No		No	.	.
AREA	Input	Nominal	No		No	.	.
CAR_USE	Input	Nominal	No		No	.	.
CHILDREN	Input	Interval	No		No	.	.
CHILD_DRIV	Input	Interval	No		No	.	.
CLAIM_AMOUNT	Target	Interval	No		No	.	.
CLAIM_IND	Target	Interval	No		No	.	.
CLM_FREQ	Input	Interval	No		No	.	.
DISTANCE	Input	Interval	No		No	.	.
EDU_LEVEL	Input	Nominal	No		No	.	.
GENDER	Input	Nominal	No		No	.	.
HOUSE_VAL	Input	Interval	No		No	.	.
ID	ID	Nominal	No		No	.	.
INCOME	Input	Interval	No		No	.	.
JOB	Input	Nominal	No		No	.	.
MVR_PTS	Input	Interval	No		No	.	.
REVOKED	Input	Nominal	No		No	.	.
STATE_CODE	Input	Nominal	No		No	.	.
STATUS	Input	Nominal	No		No	.	.
VEHICLE_AGE	Input	Interval	No		No	.	.
VEHICLE_TYPE	Input	Nominal	No		No	.	.

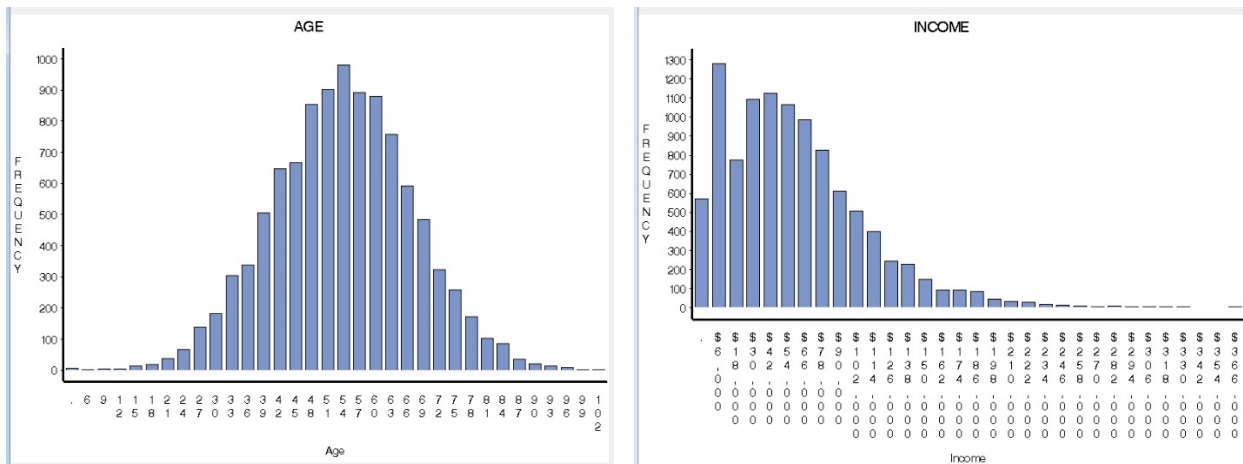
My usual first step in my analyses, before sampling or partitioning data, is to summarize the distributions of variables. In SAS Enterprise Miner I pull down the *Multiplot* node from the *Explore* menu, connect it to the data source and then *right click ... Run*. See Display 5.

Display 5. Summarizing Interval Valued Variable Using the Multiplot Node.



Histograms for two of the interval values variables are shown in Figure 4. The histogram for AGE looks remarkably like the normal density histogram while the histogram for INCOME shows the right skewness of this variable that we saw in the output from PROC UNIVARIATE in the first set of analyses.

Figure 4. Histograms of AGE and INCOME



Transforming variables is done with the *Transform Variables* node in the *Modify* menu. In the left hand menu in Display 6 I have set all the default imputation setting to none. Then I clicked on the Variables part of the menu which opened the menu box on the right of the display. There I manually selected log transformations for three of the variables. The transformation is applied with an offset of 1. That is,

$$\text{LOG_INCOME} = \log(1 + \text{INCOME}).$$

Display 6. Transforming Variables Using the Transform Variables Node.

The screenshot shows the SAS Enterprise Miner interface for 'Analyses of CLAIM_IND'. The 'Transform Variables' node is selected, and its configuration dialog is open. The dialog shows a list of variables with their methods and roles. The 'Transform Variables' node is connected to 'Auto Insurance Claims Data' and 'MultiPlot'.

Name	Method	Number of Bins	Role	Level
AGE	Default	4	Input	Interval
AREA	Default	4	Input	Nominal
CAR_USE	Default	4	Input	Nominal
CHILDREN	Default	4	Input	Interval
CHILD_DRIV	Default	4	Input	Interval
CLAIM_AMOUNT	Log	4	Target	Interval
CLAIM_IND	Default	4	Target	Interval
CLM_FREQ	Default	4	Input	Interval
DISTANCE	Default	4	Input	Interval
EDU_LEVEL	Default	4	Input	Nominal
GENDER	Default	4	Input	Nominal
HOUSE_VAL	Log	4	Input	Interval
INCOME	Log	4	Input	Interval
JOB	Default	4	Input	Nominal
MVR_PTS	Default	4	Input	Interval
REVOKED	Default	4	Input	Nominal
STATE_CODE	Default	4	Input	Nominal
STATUS	Default	4	Input	Nominal
VEHICLE_AGE	Default	4	Input	Interval
VEHICLE_TYPE	Default	4	Input	Nominal
VEHICLE_VAL	Default	4	Input	Interval
YOJ	Default	4	Input	Interval

Imputation of missing values is accomplished using the Impute node, also in the Modify menu. I switched off the default imputation and the specified using the mean value to impute LOG_INCOME, LOG_HOUSE_VAL, VEHICLE_AGE, and YOJ.

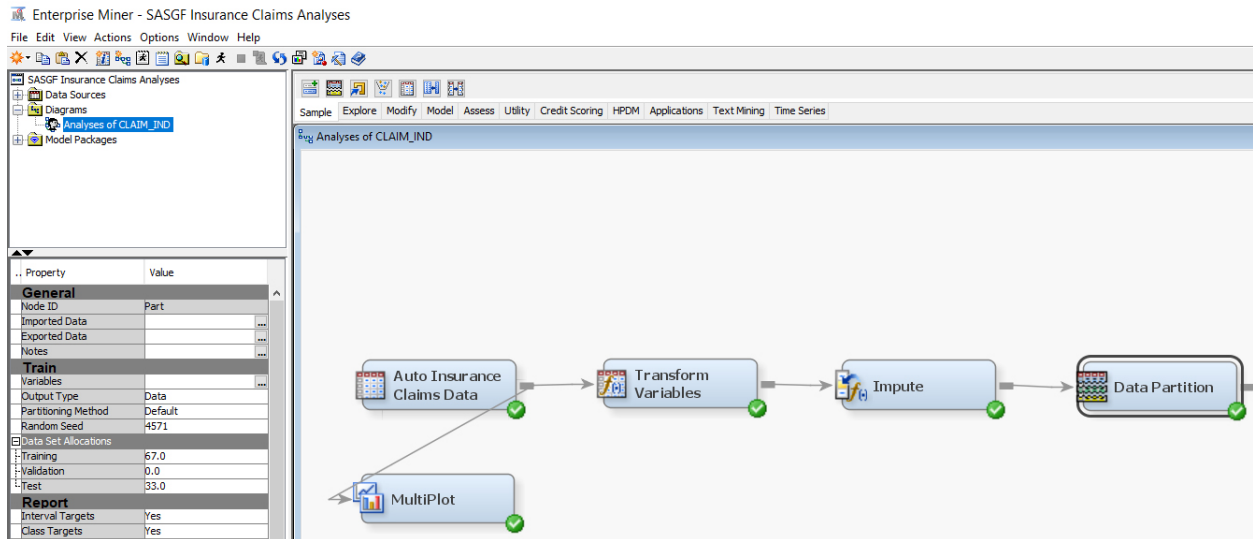
Display 7. Imputation of Missing Values Using the Impute Node.

The screenshot shows the SAS Enterprise Miner interface for 'Analyses of CLAIM_IND'. The 'Impute' node is selected, and its configuration dialog is open. The dialog shows a list of variables with their methods and roles. The 'Impute' node is connected to 'Auto Insurance Claims Data', 'Transform Variables', and 'MultiPlot'.

Name	Use	Method	Use Tree	Role	Level
AGE	Default	Default	Default	Input	Interval
AREA	Default	Default	Default	Input	Nominal
CAR_USE	Default	Default	Default	Input	Nominal
CHILDREN	Default	Default	Default	Input	Interval
CHILD_DRIV	Default	Default	Default	Input	Interval
CLAIM_IND	Default	Default	Default	Target	Interval
CLM_FREQ	Default	Default	Default	Input	Interval
DISTANCE	Default	Default	Default	Input	Interval
EDU_LEVEL	Default	Default	Default	Input	Nominal
GENDER	Default	Default	Default	Input	Nominal
JOB	Default	Default	Default	Input	Nominal
LOG_CLAIM_AM	Default	Default	Default	Target	Interval
LOG_HOUSE_VAL	Yes	Mean	Default	Input	Interval
LOG_INCOME	Yes	Mean	Default	Input	Interval
MVR_PTS	Default	Default	Default	Input	Interval
REVOKED	Default	Default	Default	Input	Nominal
STATE_CODE	Default	Default	Default	Input	Nominal
STATUS	Default	Default	Default	Input	Nominal
VEHICLE_AGE	Yes	Mean	Default	Input	Interval
VEHICLE_TYPE	Default	Default	Default	Input	Nominal
VEHICLE_VAL	Default	Default	Default	Input	Interval
YOJ	Yes	Default	Default	Input	Interval

To this point, all the data manipulations and adjustments have been identical to those carried out in the usual SAS programming language in the first part of the paper. The final step of data manipulation is to (randomly) partition the insurance claims dataset into training and test pieces. The *Partition* node in the *Sample* menu allows one to do this quickly and easily. In Display 8 I have selected 67% ($n = 6901$) of the data for the training dataset and the remaining 33% ($n = 3399$) for the test data.

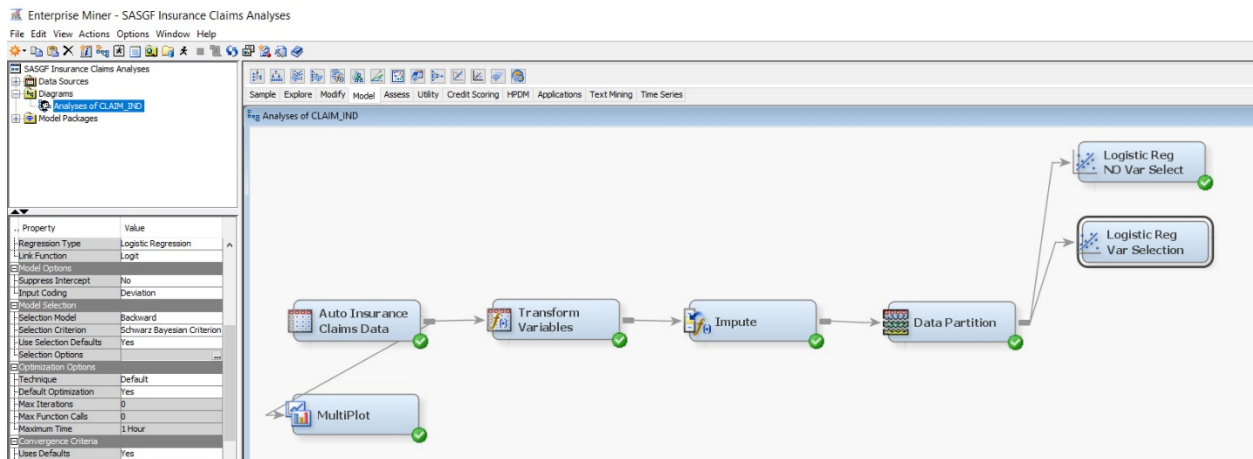
Display 8. Partitioning the Insurance Claims Data into Training and Test Datasets.



LOGISTIC REGRESSION ANALYSIS

Using the *Regression* node in the Model menu I fit logistic regression models for CLAIM_IND with and without variable selection. I renamed the nodes to reflect the analyses carried out in each node. Variable selection is automatic so I had to switch it off for the higher node. For the lower node I selected backward elimination as the method. Using P-values as a stopping criterion is not available in this node so, instead, I used Schwarz's criterion to select the model.

Display 9. Logistic Regression with and without Variable Selection.



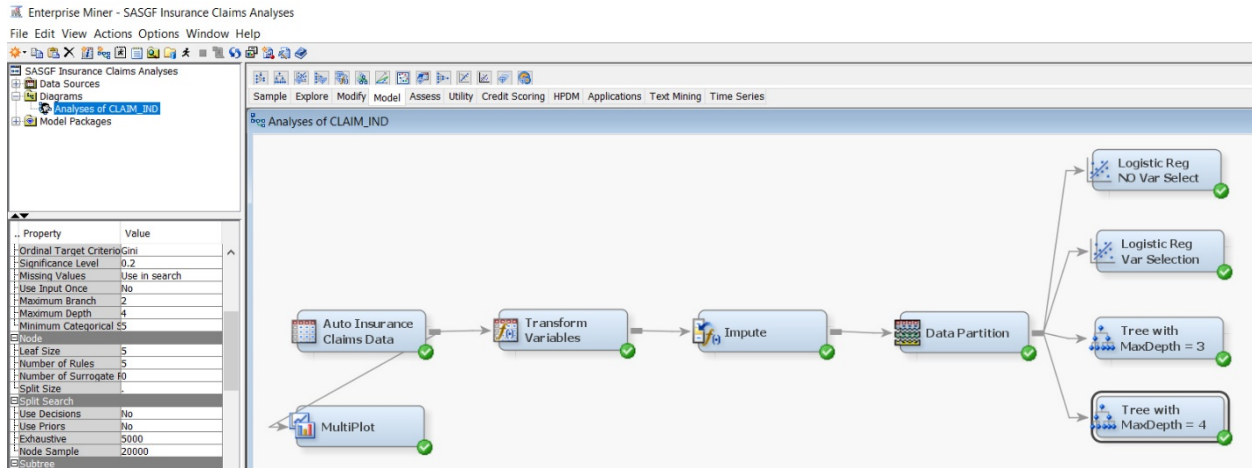
Discussion of the results of these analyses are postponed until later in the paper.

CLASSIFICATION TREES

Classification trees for a binary, interval-valued, and multicategory response may be fit using the *Decision Tree* node in the Model menu (Display 10). For controlling the size of the fitted tree the item in the left menu

labelled Depth is used. I fit trees of depths three and four (which had between 8 and 16 leaves). Discussion of the fitted trees and their accuracies is postponed until later.

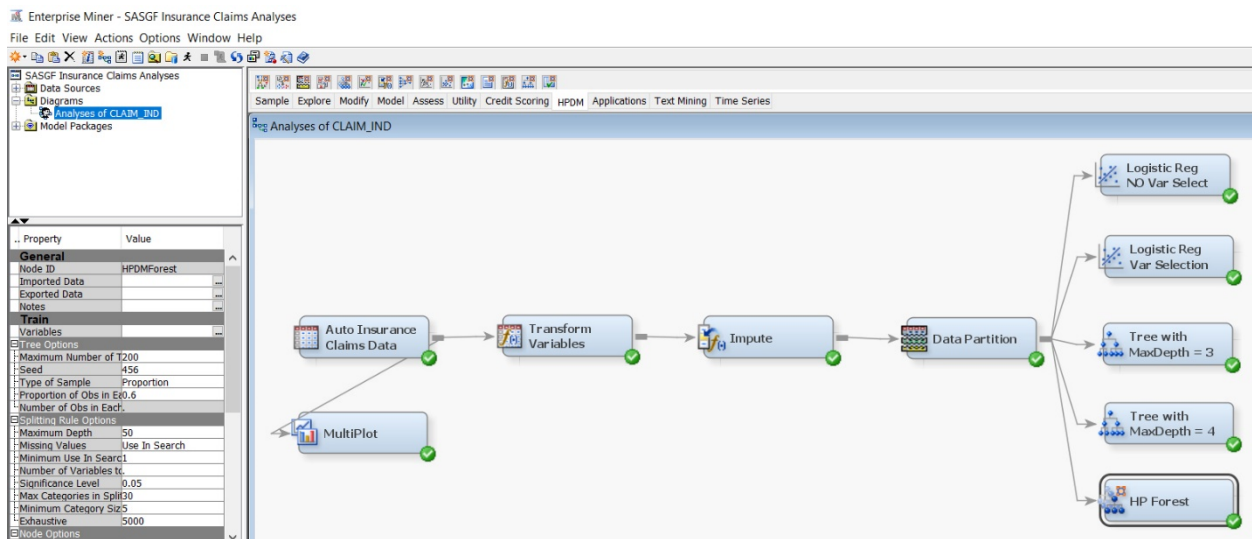
Display 10. Fitting Classification Trees of Depths Three and Four.



RANDOM FORESTS

Within SAS Enterprise Miner random forests may be found in the *HPForest* node of the HPDM menu. By design random forests has few parameters that need to be set. I increased the maximum number of trees to be fit from the default value of 100 to 200. The proportion of observations in the original dataset to have in each sampled dataset has a default value of 0.6 and I have yet to encounter a problem in which choosing a different value of this parameter yields a significantly more accurate set of predictions. As with the previous predictive analytic methods applied in SAS Enterprise Miner to the insurance claim data, a discussion of the results is postponed until later in this paper.

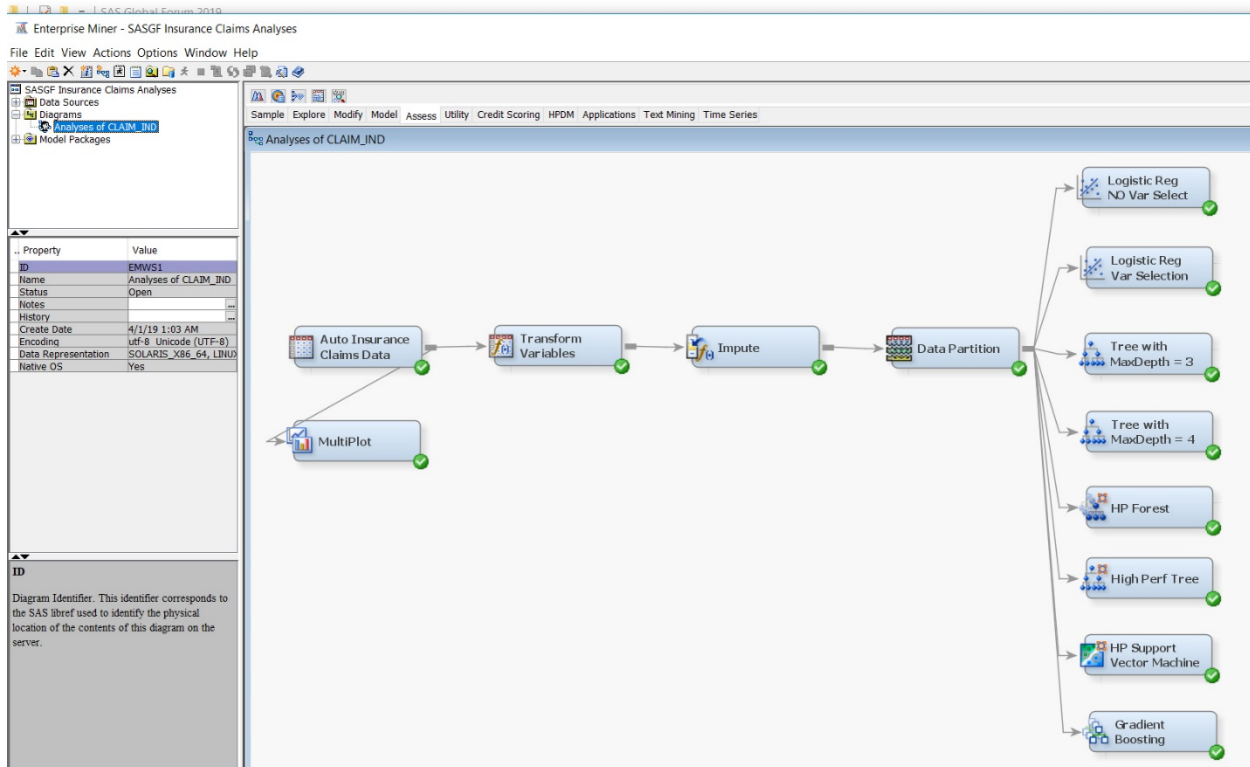
Display 11. Fitting Random Forests to the Insurance Claims Data.



GRADIENT BOOSTING MACHINES, SUPPORT VECTOR MACHINES AND MORE TREES

Several other cutting edge classification procedures may be applied to data in SAS Enterprise Miner. The *Gradient Boosting* node in the Model menu fits gradient boosting machines (Friedman 2001). Within the HPDM menu the HP SVM node may be used to fit support vector machines and the HP Tree node is a slightly different implementation of decision trees. Because it is so easy to do so I added all three methods to my analysis: see Display 12.

Display 12. Fitting Gradient Boosting Machines, Support Vector Machines and a Tree.



MODEL COMPARISON

In preceding steps we have fit many predictive methods but not discussed the results or compared the methods. One of the most remarkable features of SAS Enterprise Miner is the *Model Comparison* node in the Assess menu. This allows us to compare the predictive accuracies of many methods applied to the same data in one foul swoop. There is almost nothing to do here other than to join all the predictive analytics nodes to the *Model Comparison* node and clicking run. See Display 13 for the final diagram for these analyses. From the output of the *Model Comparison* node Table 5 was constructed. It contains the Percent Correct (PCC) and the AUC values for each method on both the training and test data. There is actually more information available in the output to the *Model Comparison* node: for the training data the full confusion matrix is available and hence so are the sensitivity and specificity.

The differences are very small for these data but the two logistic regression models—with and without variable selection—have slightly higher accuracies than the other methods while the three classification trees and support vector machines have slightly lower accuracies.

Display 13. The Model Comparison Node to Compare the Accuracies of Predictive Methods

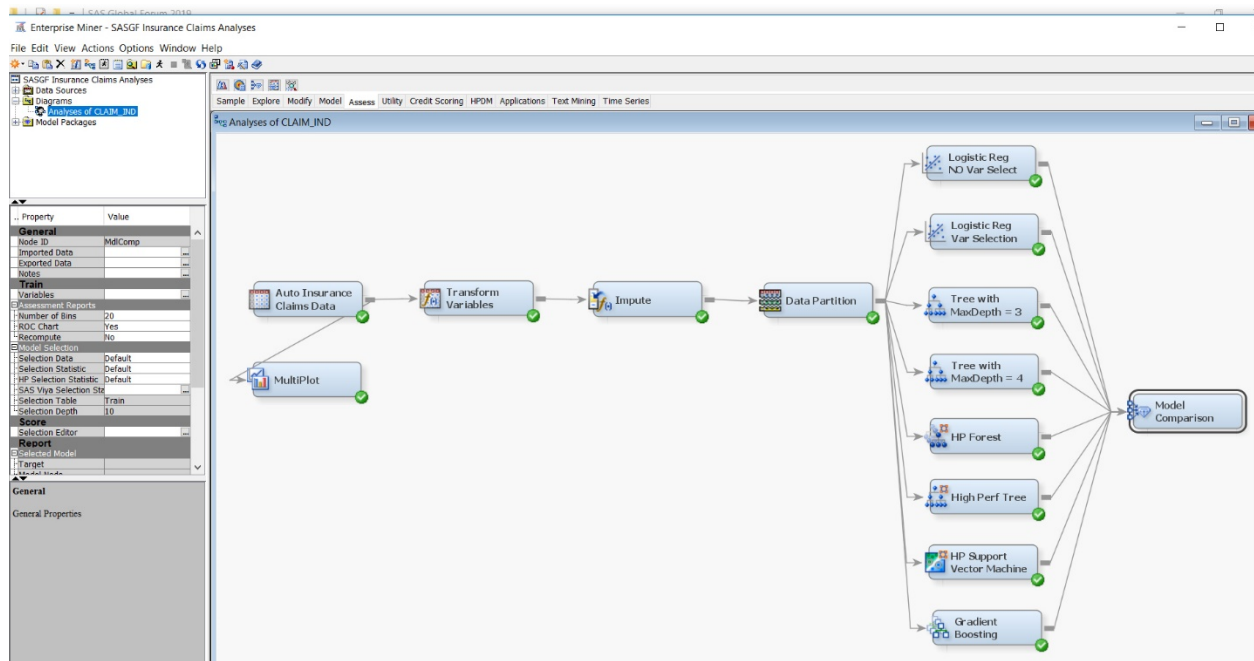
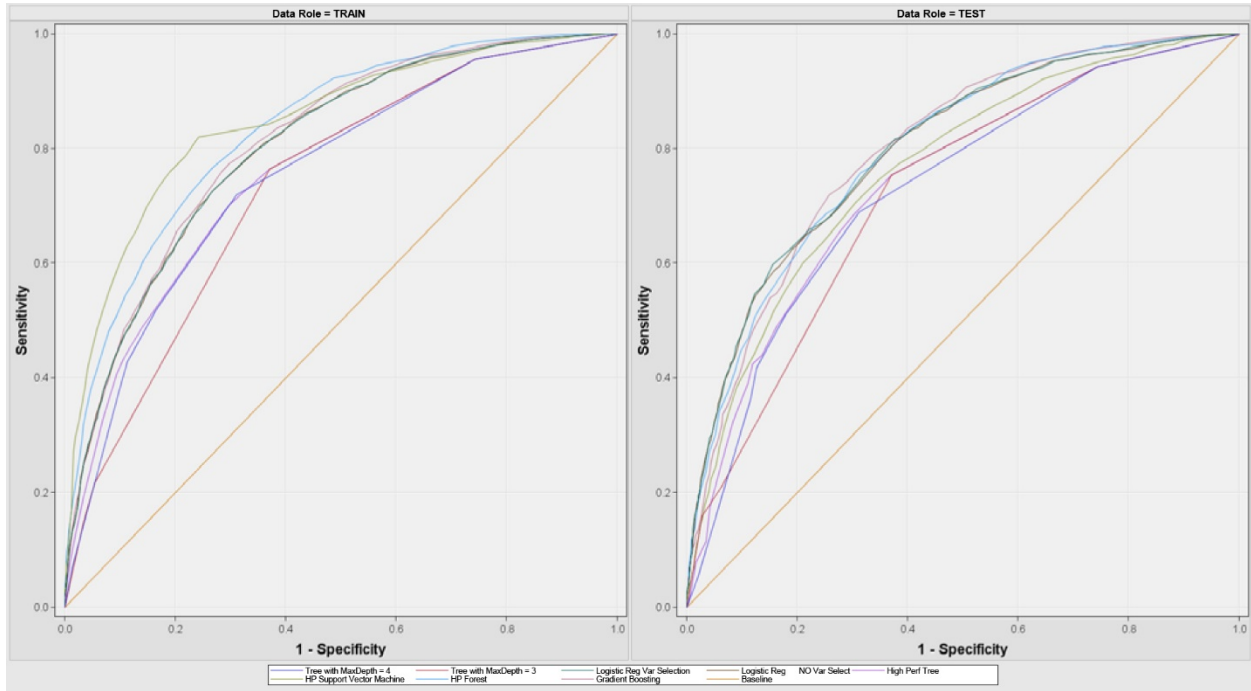


Table 5. Predictive Accuracies for Logistic Regression with and without Variable Selection, Classification Trees, Random Forests, Gradient Boosting Machines and Support Vector Machines on Training and Test Data

Method	Accuracy Metric			
	Percent Correctly Classified		AUC	
	Training Data	Test Data	Training Data	Test Data
Logistic Regression Full Model	78%	79%	0.80	0.80
Logistic Regression Variable Selection	78%	79%	0.80	0.80
Tree Depth = 3	75%	74%	0.73	0.72
Tree Depth = 4	76%	75%	0.75	0.73
Random Forests	77%	76%	0.83	0.80
Gradient Boosting	78%	77%	0.84	0.80
Support Vector Machines	82%	76%	0.85	0.76
High Performance Tree	77%	76%	0.76	0.74

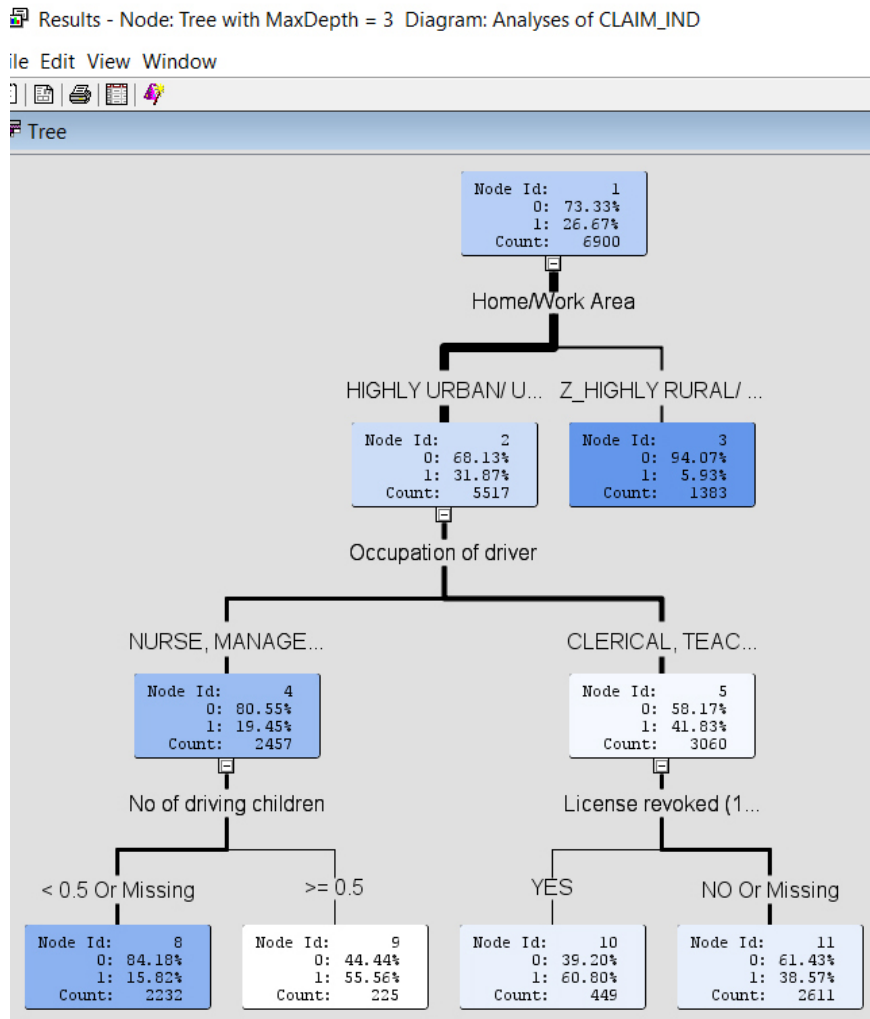
Figure 5 shows ROC curves for all the methods overlaid on the same plot for the training and test data separately. The extent to which the curves overlap is another indicator of the modest differences in the predictive accuracies among the different methods.

Figure 5. ROC Curves for Predictive Methods for Training and Test Data.



There is considerably more information in the output of the individual predictive methods. For example, random forests identifies AREA, JOB, CLAIM_FREQ, CHILD_DRIV (number of children who are driving), REVOKED, EDU_LEVEL, and CHILDREN as the most important variables. The high performance tree has a similar list of important variables. With regard to interpretation of the variables, we may examine one of the classification trees (Figure 6). For people living and working in a rural/highly rural area the accident rate is less than 6% whereas for those living in an urban/highly urban area the accident rate is over 30%. That is a very substantial difference. Among the drivers who live and work in urban and highly urban areas, the next split is on the occupation of the driver with those who are bankers, doctors, and managers having are much lower accident rate (less than 20%) than those in other professions, including clerical and being a student.

Figure 6. Classification Tree of Depth 3.



CONCLUSION

In this article I have shown more or less the same analyses of some insurance claim data using regular SAS programming and in SAS Enterprise Miner. In the latter environment it is very easy to do additional analyses and to compare the predictive accuracies of the different methods. What may be surprising to the reader is how long it takes to do these analyses. Even someone who is experienced in coding in SAS will make errors and have to look up syntax for particular options. In contrast, SAS Enterprise Miner involves selecting options and rerunning an analysis if a mistake is made is almost trivial. And then there is the time spent on the analyses. The SAS program I ran to obtain the output and results presented in the first part of this paper took the better part of half a day while the SAS Enterprise Miner diagram took me only about 45 minutes. My message is that if you have access to SAS Enterprise Miner, it is an alternative to standard SAS programming that is very well worth considering. There is a steep learning curve to get started in SAS Enterprise Miner but once that has been done it is an excellent environment for conducting many statistical analyses quickly and being able to compare the results of many methods.

REFERENCES

- Breiman, Leo. 2001. "Random forests." *Machine Learning* 45(1):5—32.
- Breiman, Leo, Jerome Friedman, Richard Olshen, and Charles Stone. 1984. *Classification and Regression Trees*. Boca Raton, FL: Chapman & Hall.
- Cortes, Corinna and Vladimir Vapnik. 1995. "Support-vector networks." *Machine Learning* 20:273—297.
- Cutler, Richard, Thomas Edwards Jr., Karen Beard, Adele Cutler, Kyle Hess, Jacob Gibson, and Joshua Lawler. 2007. "Random Forests for Classification in Ecology." *Ecology* 88(11):2783—2792.
- Friedman, Jerome. 2001. "Greedy function approximation: The gradient boosting machine." *Annals of Statistics* 29(5):1189—1232.
- Moro, S., R. Laureano and P. Cortez. 2011. "Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology." In P. Novais et al. (Eds.), *Proceedings of the European Simulation and Modelling Conference*, pp. 117—121, Guimarães, Portugal, October, 2011.
- Vapnik, Vladimir. 1995. *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, Inc.