**MIS 6334.001**

***Data and Model Analytics using SAS Enterprise Miner***

# **Executive Summary**

For this project, we have created the entire SEMMA process in SAS Enterprise Miner. The dataset used in this project is from Expedia; a travel company that facilitates booking flights, hotels, renting transportation etc. for travelers. The objective of our analysis was to determine whether a user is going to complete the booking process in the remainder of their session on the website, based on a variety of information about user demographics, browsing habits, details of time spent on the site, browsing sessions, previous bookings etc.

While creating the data source, we created a cost-based confusion matrix to be used for evaluating models. We explored the data extensively and performed various pre-processing techniques on it. We then trained various models using the data. We have also tried to enhance model performance by reducing the number of attributes, sampling the data and using Bagging and Ensemble techniques. We have then compared all the models and scored a new data using the champion models.

We have followed the entire SEMMA process in the project:-

* *Sample* - we defined the data source, assigned roles and measurements, and partitioned the data into training data and validation data. We have also sampled the data set for the 2nd process flow to balance out the number of 1 and 0 values of the Target Variable. We used File Import node to convert xls file to sas file for X12 pre-processing.
* *Explore* – we generated descriptive statistics of the data using the StatExplore node and also used the Variable Selection node for reducing the number of input variables
* *Modify* – we used the Impute node for handling missing values, Drop node to drop attributes which may not be significant for further analysis, Transform Variables node for transforming variables to improve the fit of the models to the data and Principal Component node to do PCA.
* *Model* – we have used the analytical tools to train a statistical or machine learning model to reliably predict a desired outcome. For this, we have used logistic regression, decision trees, ensemble and neural networks nodes.
* *Assess* – We have evaluated the usefulness and reliability of the findings from the data mining process by comparing models using Model Compare node and performed score code management using the Score node.

Under *Utility*, we have used Start Groups and End Groups nodes for running Bagging and SAS code node to run a SAS code for displaying the predicted target values of the score data.

In this project, we have created 2 process flows; one for the entire data set and the other for the sampled data set. In the report, we have presented a detailed snapshot of each of the 2 process flows along with explanation and reasons behind the various steps taken by us in the flow.

We have also included a section on our learnings and the over-all experience of working on the project, at the end of the report.

**Contents**

[**Executive Summary** 1](#_Toc446959411)

[**Data Description** 3](#_Toc446959412)

[**Setting Up The Project** 4](#_Toc446959413)

[**Part I. Basic Data Preprocessing** 7](#_Toc446959414)

[**Part II. Building Decision Trees** 11](#_Toc446959415)

[**Optimal Decision Tree** 11](#_Toc446959416)

[**X12 Pre-processing** 13](#_Toc446959417)

[**Interactive Decision Tree** 17](#_Toc446959418)

[**Part III. Building Neural Networks and a Regression Model** 21](#_Toc446959419)

[**Impute** 21](#_Toc446959420)

[**Variable Transformations** 23](#_Toc446959421)

[**Logistic Regression** 25](#_Toc446959422)

[**Comparison between Logistic Regression with Transformation and without Transformation:-** 29](#_Toc446959423)

[**Neural Network** 29](#_Toc446959424)

[**Comparison between Neural Network with Transformation and without Transformation:-** 33](#_Toc446959425)

[**Part IV. Model Comparison and Champion Model Evaluation** 34](#_Toc446959426)

[**Model Comparison** 34](#_Toc446959427)

[**Scoring New Data** 36](#_Toc446959428)

[**Part V. Improve Your Model Performance** 38](#_Toc446959429)

[**Imputation** 38](#_Toc446959430)

[**Variable Transformation** 40](#_Toc446959431)

[**Variable Reduction** 41](#_Toc446959432)

[**Comparison between the 3 Variable Reduction Techniques:-** 45](#_Toc446959433)

[**Sampling** 45](#_Toc446959434)

[**Comparison of Model performance with Sampled and Non-sampled data** 60](#_Toc446959435)

[**Ensemble & Bagging** 60](#_Toc446959436)

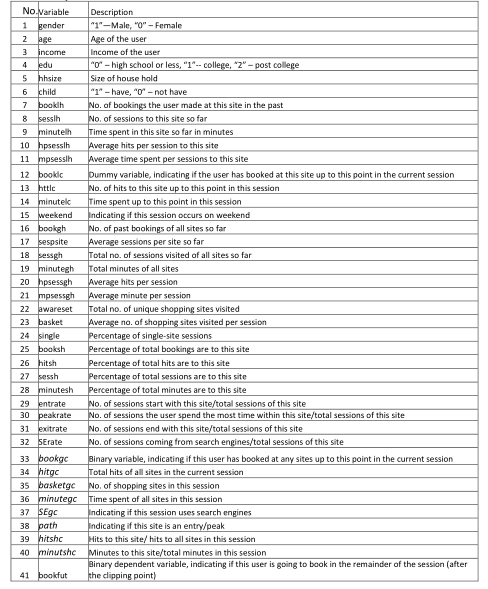
[**Part VI. Summary** 63](#_Toc446959437)

[**Final Diagrams** 63](#_Toc446959438)

[**Learnings** 65](#_Toc446959439)

[**References** 66](#_Toc446959440)

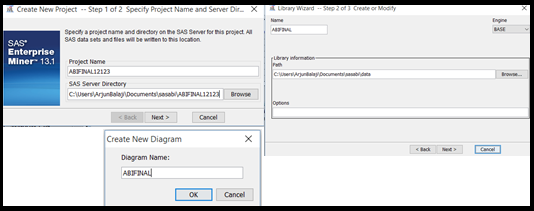
# **Data Description**



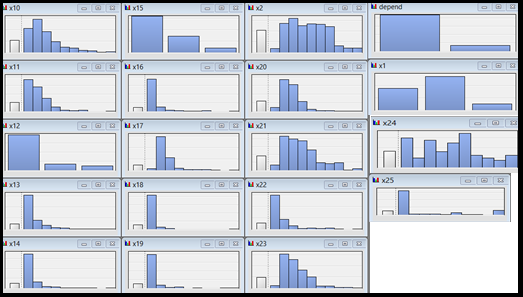
Number of customers represented = 1010

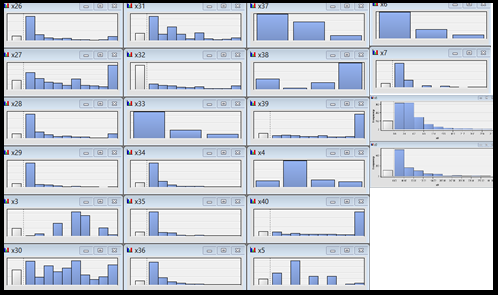
# **Setting Up The Project**

1. We created a new project, a new library where the data sets to be used are stored and a new diagram.

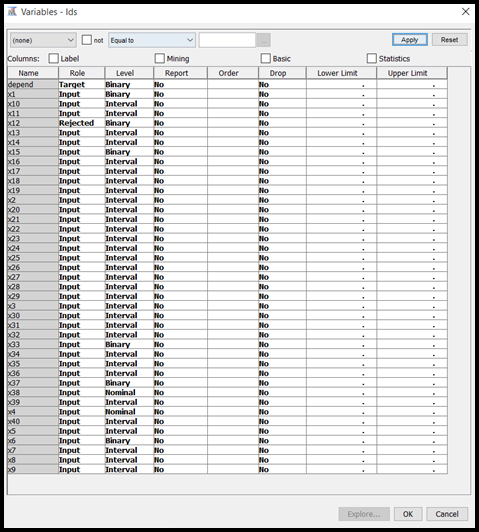


1. For creating the datasourse, we selected the data set Expediatrain from the library and studied the data distribution of each variable to set the their levels to appropriate values.



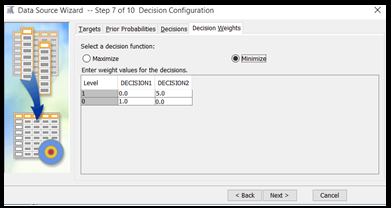


1. We selected the *advanced* option in the *Data Source Wizard - Metadata Advisor options* and changed role of X32, X38 from Rejected to Input and the role of depend to Target. We also changed Level of X3, X35, X5, and X7 from Nominal to Interval.

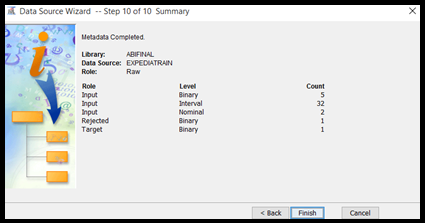


1. We then selected the Yes option button to indicate that we want to *build models* based on the values of decisions.

On the *Decision Weights* tab, we created the cost-based confusion matrix, with the cost misclassifying 1 as 0 being 5 and the cost misclassifying 0 as 1 being 1. We select the *Minimize* button indicating that we want to minimize the loss in the analysis.



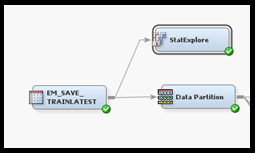
1. In the *Data Source Wizard — Create Sample* window, you selected No for creating sample data set as we wished to use the entire data set. We then set the Role of the data source to raw ad click on Finish.



# **Part I. Basic Data Preprocessing**

1. **Use dataset expediatrain.sas7bdat. Provide a brief summary of the variables in the data set (you can choose to use StatExplore for that purpose).**

We added a *StatExplore* node to the *datasource* – em\_train\_trainlatest.sas7bdat to study the statistical summary of the input data. Stat Explorer node is used to analyze the variable statistics.

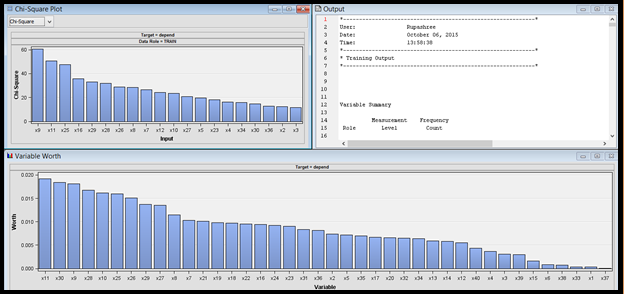


By default, the StatExplore node creates Chi-Square statistics and correlation statistics. A chi square (*X2*) statistic is used to investigate whether distributions of categorical variables differ from one another and thus are used for categorical variables only. To use Chi-Square statistics for the interval variables in addition to the class variables, one needs to bin the interval variables. Hence before running this node we changed the selection for *interval variables* in the *Chi-Square Statistics*properties group in SAS Enterprise Miner to ‘Yes’ in order to distribute interval variables into five (by default) bins and compute chi-square statistics for the binned variables when the node is run.

We executed the StatExplore node.

The Result window displays the following:-

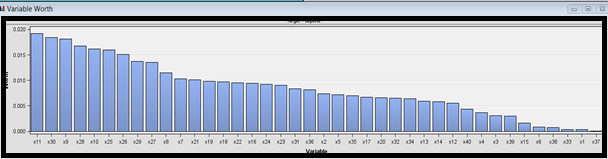
1. The *Variable Worth* plot orders the variables by their worth in predicting the target variable based on the Gini split worth statistic.
2. The SAS output window provides distribution and summary statistics for the class and interval inputs, including summaries that are relative to the target. The *Interval Variables Summary Statistics* section and the *Class Variable Summary Statistics*sections of the output has the Non-Missing column and the Missing column which list the number of observations that have valid values and the number of observations that have missing values for each interval variable respectively.
3. The *Chi-square* plot orders the top 20 variables by their chi-square statistics.



We observed that all the variables have missing values. There are 8 class variables including the target variable. All the remaining variables are interval. We also observed that quite a few variables have a skewed distribution.

1. **Explore the statistical properties of the variables in the input data set. The results that are generated in this step will give you an idea of which variables are most useful in predicting the target response. Unless you see anything interesting, no need to report the details of this step.**

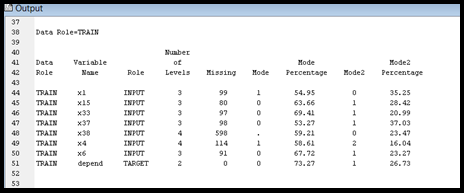
The *Variable Worth* plot gives us a good idea on the relative worth of input variables in predicting the target variables.

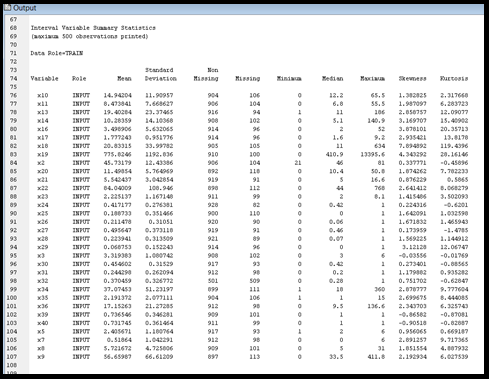


We observe X33 has the highest worth, followed by x11, x30, x9 and so on

1. **Check the Class Variable Summary Statistics and the Interval Variable Summary Statistics sections of the output.** 
   1. **Are there any missing values for any of the variables? Use imputation to fill in all missing data (describe how you did imputation in the report).**

By observing the *Class Variable Summary Statistics* and the *Interval Variable Summary Statistics* sections of the Output, we saw that all Class as well Interval variables have missing values.





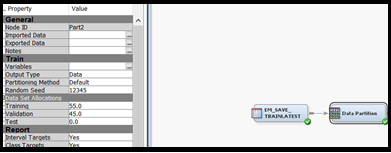
As observed above, all variables have a substantial proportion of missing values. For Decision Trees, missing values do not pose any problems as Surrogate splitting rule enables the node to use the values of other input variables to perform splits with missing values.

However, models like Regression and Neural Network altogether ignore the records that contain missing values, which would substantially reduce the size of the training data set in our case, which in turn would have reduced the predictive power of these models. Hence there is a need to impute missing values before using Regression and Neural Network models.

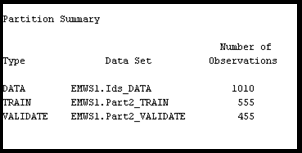
As Decision Tree nodes can handle missing values themselves, we decided to impute the missing values in the data set using the Impute node only before using Logistic Regression and Neural Network models. Imputing the missing values before fitting these models is essential also because we would be comparing these models with Decision Trees. Model comparisons are more appropriate between models that are fit with the same set of observations.

1. **Partition dataset expediatrain.sas7bdat. Use 55% of the data for training and 45% for validation.**

The Data Partition node was added by setting the *training* and *validation* property in *Data Set Allocation* to 0.55 and 0.45 respectively.



The below figure shows the Partition summary of the dataset.



# **Part II. Building Decision Trees**

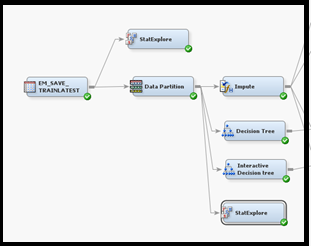
Decision trees are a simple, but powerful form of multiple variable analysis. Decision trees are produced by algorithms that identify various ways of splitting a data set into branch-like segments by breaking down the dataset into smaller and smaller subsets until a termination criteria is reached. These segments form an inverted decision tree that originates with a root node at the top of the tree. The final result is a tree with decision nodes and leaf nodes. Each leaf node represents a class assignment and the decision node represents a test on a particular attribute to further split the tree into branches.

## **Optimal Decision Tree**

1. **Enable SAS Enterprise Miner to automatically train a full decision tree and to automatically prune the tree to an optimal size. When training the tree, you select split rules at each step to maximize the split decision logworth. Split decision logworth is a statistic that measures the effectiveness of a particular split decision at differentiating values of the target variable. For more information about logworth, see SAS Enterprise Miner Help. Report the results.**

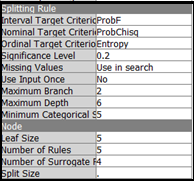
SAS automatically trains a full decision tree and prunes the tree to an optimal size. When training the tree, SAS selects logworth criteria to split rules at each step to maximize the split decision logworth.

After verifying the input data, we followed the following steps to model the input data using nonparametric decision trees. After Data Partition we added the *Decision Tree* node which enables us to perform multi-way splitting of the database, based on nominal, ordinal, and continuous variables. The SAS implementation of decision trees represents a hybrid of the best of CHAID, CART, and C4.5 algorithms. When the Tree node is run in the Automatic mode, SAS automatically ranks the input variables by the strength of their contribution to the tree. This ranking can be used to select variables for use in subsequent modeling.



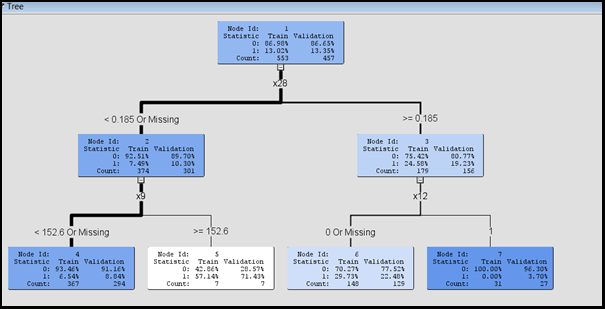
In the Decision Tree node, in the Properties Panel, under Train properties, we made the below settings:

1. *Maximum Depth splitting* rule property was set to 6. This specification enables SAS Enterprise Miner to train a tree that includes up to six generations of the root node.
2. *Leaf Size* node property was set to 5. This specification constrains the minimum number of training observations in any leaf to five.
3. *Number of Surrogate Rules* node property was set to 4. This specification enables SAS Enterprise Miner to use up to four surrogate rules in each non-leaf node if the main splitting rule relies on an input whose value is missing.



After making these changes we executed the decision tree node. The results of the tree are displayed below:

The optimal tree created is shown below:



The above Decision Tree shows some unusual behavior of the variable X12, which provides a pure split when its value is 1. On further studying the data set, we observed that whenever the value of X12 is 1, the value of the target variable is always 0.

The variable X12 indicates whether a user has booked at this site up to this point in the current session. If a user has already booked at this site up to this point in the current session i.e. when X12 is 1, he/she is definitely not going to book in the remainder of the session, i.e. the Target variable would be 0.

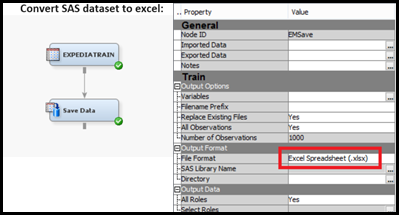
Hence we decided to do some pre-processing for X12.

### **X12 Pre-processing**

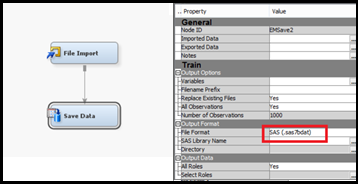
Due to this anomalous behavior of X12, we decided to drop it from the analysis. However before dropping it, we needed to change the value of target variable from 0 to 1 when X12 is 1. This is because X12=1 indicates that the user has booked in this session which can be equated to the Target variable=1 i.e. user is going to book in the current session. Hence all factors that influence the Target Variable to be equal to 1 would be influencing X12 to be 1 as well.

To do the above mentioned pre-processing we followed the following steps:-

1. We exported the SAS Data source to an excel file using Save Data Node.



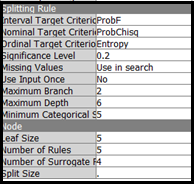
1. In Excel sheet we altered the target variable values from 0 to 1 when X12 is set to 1. The proportion of 0 and 1 in the dataset changed from 877:133 to 740:270.
2. After altering the Target variable, we converted excel back to a SAS dataset using the File Import and Save Data node and used it for our analysis.



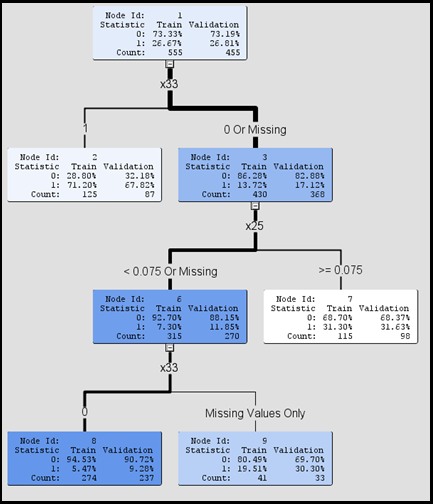
1. We set up the project using this new data set with the above mentioned settings:-
   * 1. We set the role of x12 from Input to Reject, x32 and x38 from Rejected to Input and the role of depend to Target. We also changed the levels of x3, x5, x7 and x35 from Nominal to Interval.
     2. On the *Decision Weights* tab, we created the cost-based confusion matrix, with the cost misclassifying 1 as 0 being 5 and the cost misclassifying 0 as 1 being 1. We select the *Minimize* button indicating that we want to minimize the loss in the analysis.
2. The Data Partition node was added by setting the *training* and *validation* property in *Data Set Allocation* to 0.55 and 0.45 respectively.

A Decision Tree node was added, with the following settings in the Properties Panel, under Train properties:

1. *Maximum Depth splitting* rule property was set to 6. This specification enables SAS Enterprise Miner to train a tree that includes up to six generations of the root node.
2. *Leaf Size* node property was set to 5. This specification constrains the minimum number of training observations in any leaf to five.
3. *Number of Surrogate Rules* node property was set to 4. This specification enables SAS Enterprise Miner to use up to four surrogate rules in each non-leaf node if the main splitting rule relies on an input whose value is missing.



After making these changes we executed the decision tree node. The results of the tree are displayed below:

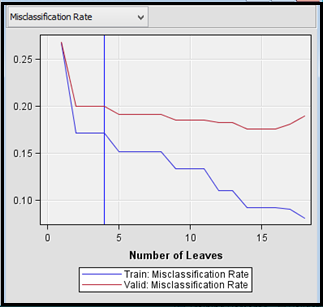


The optimal tree that is formed, initially split on the attribute *bookgc* with highest logworth. After that, the second node has split on the attribute *booksh* and the last split happened on the node 5 with the attribute *booksc*. Split decision logworth is a statistic that measures the effectiveness of a particular split decision at differentiating values of the target variable. The leaf nodes thus formed have purer and purer subsets as we go down the various levels. The object of analysis, “*bookfut*” is reflected in the root node as a simple, one-dimensional display in the decision tree.

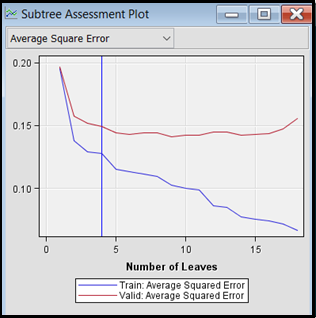
Each leaf node contains the following information:

* node number
* number of training observations in the node
* percentage of training observations in the node with *bookfut* =1 (user will book), adjusted for prior probabilities percentage of training observations in the node with *bookfut* =0 (user will not book), adjusted for prior probabilities

The *subtree assessment* plot for the *Misclassification rate* showed that the error for both the training and validation data set displays a downward trend initially. After the initial reduction it stabilized after the 3rd node in both the data set. After the 4th leaf node, Misclassification rate plummets in the training set. However the rate of decrease of Misclassification rate is much more gradual for the Validation set.



The *subtree assessment* plot for the *average square error* rate shows that the error displays a steady downward trend in the training dataset. However, in the Validation set, after an initial reduction the average square error increases, indicating over-fitting of data with increase in the number of leaves.

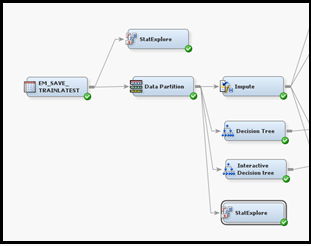


## **Interactive Decision Tree**

1. **Then, interactively train a decision tree. At each step, you select from a list of candidate rules to define the split rule that you deem to be the best. Report the results.**

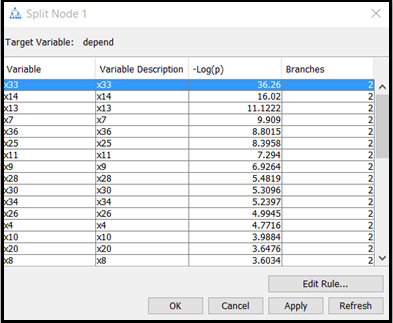
In this step, we created the Decision Tree interactively at each step by selecting from a list of candidate rules to define the split rule that seemed be the best, based on –Log (p).

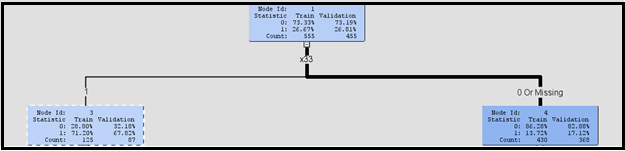
We thus added another decision tree node for - interactive decision tree making - to our workflow. Using interactive training, we can override any automatic step by defining a splitting rule or by pruning a node or subtree.



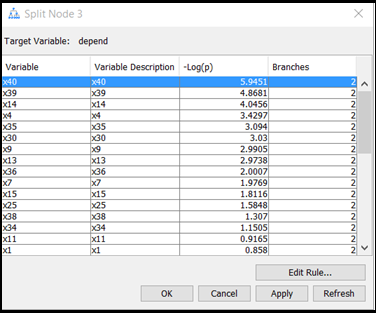
We added split points and edited the splitting condition as needed. The tree was split as the different conditions were entered.

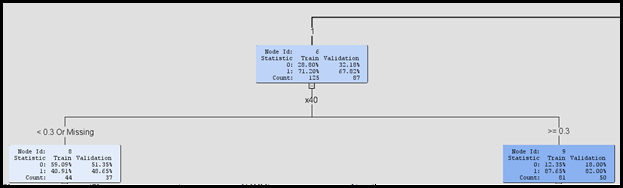
We selected the *Interactive ellipsis* from the Decision tree properties panel and in the interactive decision tree window - split the root node into further sub nodes. After the first split, the tree now had two additional nodes.



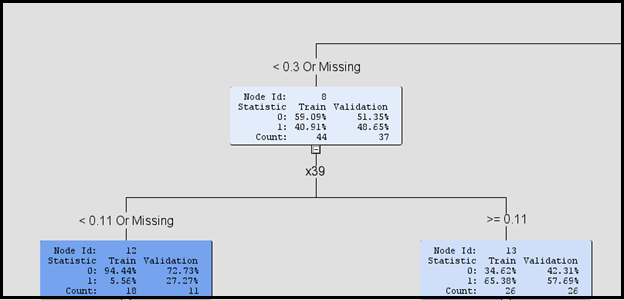


After selecting the lower left node (*bookgc*) in a new *Split Node*, we selected *minutshc*, which ranked second in logworth.

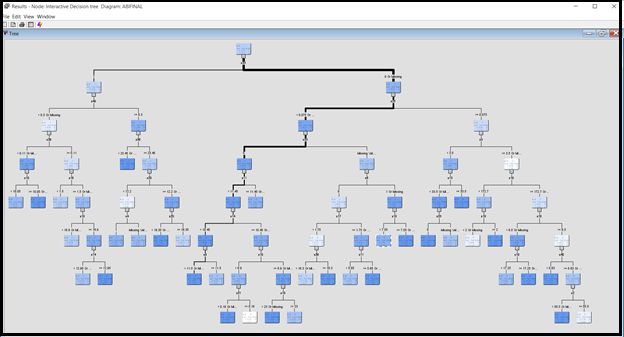




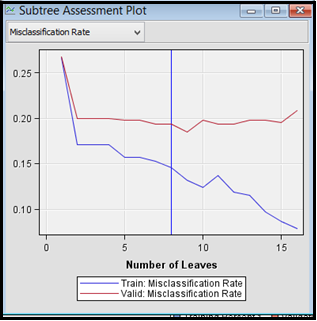
After selecting the lower right node (*minutesh*) in a new *Split Node*, we selected *hitshc*, manually split the node for this rule. We observed from the screenshot below, that we were able to create a pure subset (Node id 8) based on this attribute.



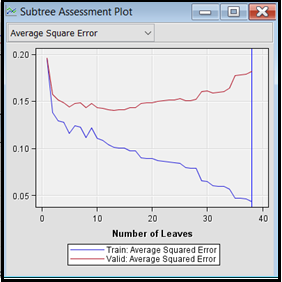
After further continuing this method to split a few nodes, we trained the tree using *Train Node* option and the tree formed is shown below:



We observed that the *Misclassification Rate* is steadily decreasing for both Training and Validation data till leaf 3. For the subsequent leaves the Misclassification Rate decreases at a much higher rate for Training Data compared to the Validation Data, indicating a better performance of the interactive tree model with respect to the training data compared to the validation data.



We observed that the average square error for the Training dataset has a downward trend. However, after the 10th leaf, the average square error for the Validation dataset increases, indicating over-fitting of data after the 10th leaf.

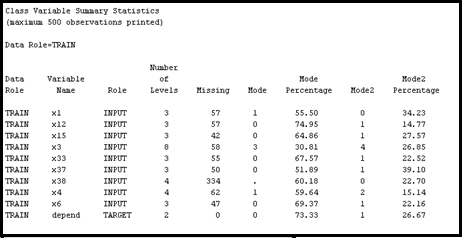


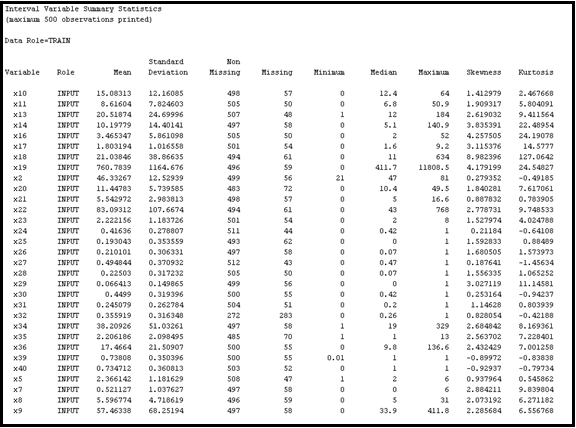
Due to this over fitting issue, the predictions results might not be generalize to new data outside the analysis dataset we have.

# **Part III. Building Neural Networks and a Regression Model**

## **Impute**

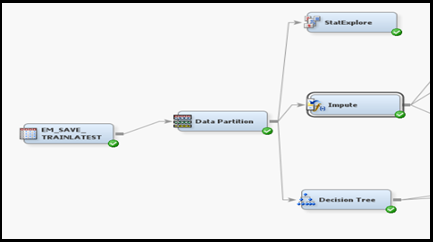
Using the stat explore node, we observed that many of the input variables have missing values.

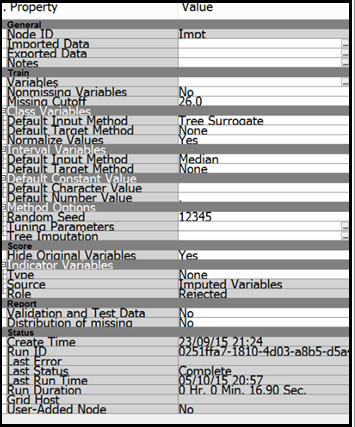




Regression and Neural Networks ignore records containing missing values. Therefore, it is recommended to impute the missing values before performing Regression or Neural Network.

The missing cut-off value for the Impute node is set to 26%, which rejects x32, and x38. For class variables, Tree Surrogate is used as the Default Input Method so that Enterprise Miner predicts the missing value by building a decision tree with that (missing) variable as the target and the other input (non-missing) variables as predictors. For interval variables, Median is selected as the Default Input Method so that the missing values are replaced by the median of the non-missing values.

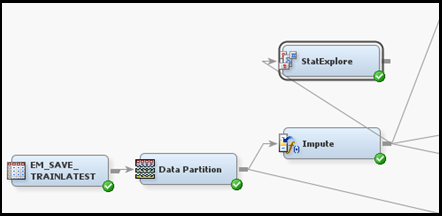




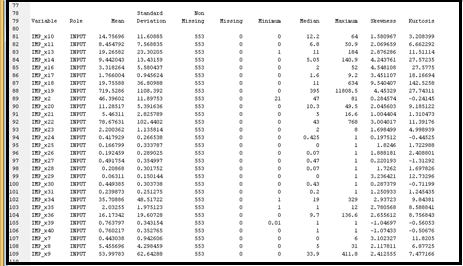
## **Variable Transformations**

1. **Transform input variables to make the usual assumptions of regression more appropriate for the input data. Explain the transformations you did.**

We added a StatExplore node (with the previously mentioned settings) to explore the imputed data.

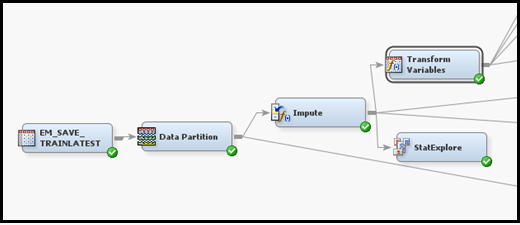


From the Interval Variable Summary Statistics in the Results - Output, we calculated the Coefficient of Variation and considered all variables with coefficient of Variation of more than 0.85 as having high variation.



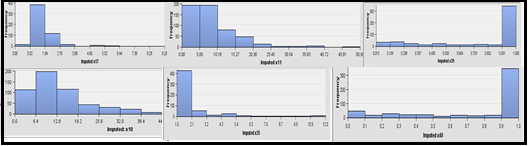
We found that x7, x9, x13, x14, x16, x18, x19, x22, x25, x26, x28, x29, x31, x34 and x36 have high variance or standard deviation. As standard deviation is directly proportional to Skewness, one way to reduce variance would be to reduce the Skewness of the above mentioned variables.

We thus added a Transform Variables node:-



We clicked on the Formulas ellipses in the Train properties of the node to observe the data distribution of the input variables.

As expected, we observed that the above mentioned variables have a highly skewed distribution. We also observed that the distribution graph of few other variables are also heavily skewed – x10, x11, x17, x35, x39 and x40.



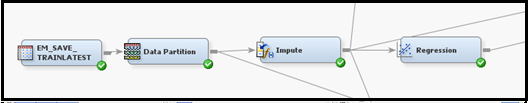
As Regression and Neural network does not work well with skewed data, we selected Log10 as the transformation Method in the Variables – Trans window for all the variables, in order to improve the fit of the model to the data. Log transformations are used to control Skewness.

## **Logistic Regression**

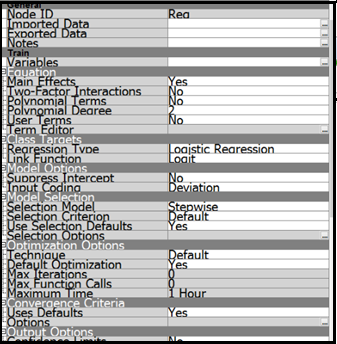
**2. Model the input data using logistic regression. Report the results.**

1. ***Logistic Regression without Transformation:-***

We added a Regression node to the Impute Node:-



The Regression node automatically performed logistic regression as the target variable in this data set is a binary variable. We selected *Stepwise* as the *Selection Model* property which begins regression with all candidate effects in the model and removes effects until the Stay Significance Level or the Stop Criterion is met and also removes effects already present in the model.



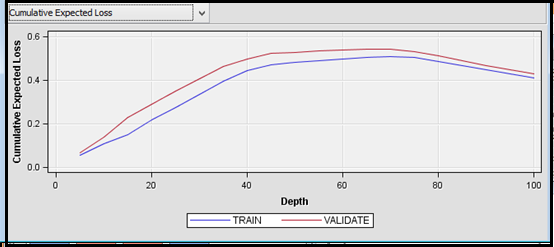
The node is executed.

We observed the output window:-



In the Results – Output, we observed the Odds ratio estimate associated with the input variables. From the Lift graph we also observed the effectiveness of the Logistic Regression model with respect to the Training and Validation data set and found the effectiveness to be comparable.

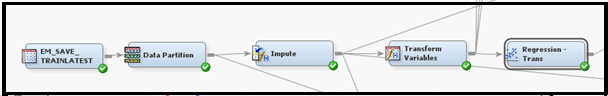
We observed the Mean Expected Loss in more detail:-



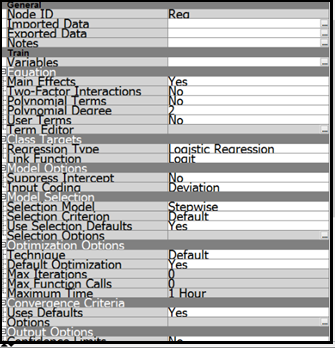
For this data set, we had defined the cost matrix. Thus the expected loss is a function of both the probability of a user booking or not booking in the given site and the estimated cost associated with each corresponding outcome. A value is computed for each decision by multiplying estimated loss values from the decision matrix with the classification probabilities. The decision with the lowest value is selected, and the value of that selected decision for each observation is used to compute the loss measures.

1. ***Logistic Regression with Transformation:-***

We added a Regression node to the Transform Variables Node:-

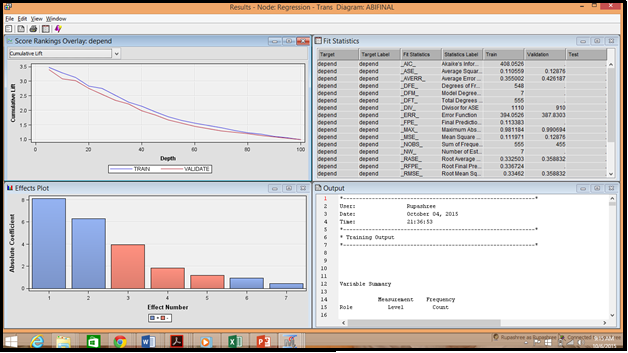


The Regression node automatically performed logistic regression as the target variable in this data set is a binary variable. As in the above case, we selected *Stepwise* as the *Selection Model* property.



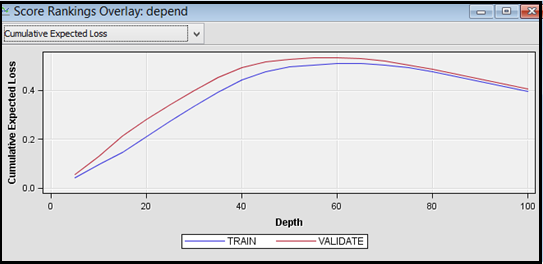
The node is executed.

We observed the output window:-



In the *Results – Output*, we observed the Odds ratio estimate associated with the input variables. From this *Lift* graph too, we can deduce that the effectiveness of the Logistic Regression model with respect to the Training and Validation data set to be comparable.

We observed the Cumulative Expected Loss in more detail:-



As explained above, the expected loss is a function of both the probability of a user booking or not booking in the given site and the estimated cost associated with each corresponding outcome. A value is computed for each decision by multiplying estimated loss values from the decision matrix with the classification probabilities. The decision with the lowest value is selected, and the value of that selected decision for each observation is used to compute the loss measures.

### **Comparison between Logistic Regression with Transformation and without Transformation:-**

From the Cumulative Expected Loss graphs, we observed that the performance of the model has increased after transforming the data. The cumulative loss for both Training and Validation data is lesser for Logistic Regression model with Transformation compared to the one without Transformation.

## **Neural Network**

**3. Model the input data using neural networks, which are more flexible than logistic regression (and more complicated). Report the results.**

Neural networks are a class of parametric models that can accommodate a wider variety of nonlinear relationships between a set of predictors and a target variable than can logistic regression.

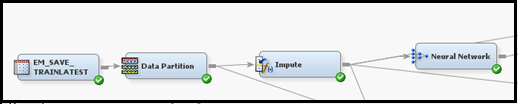
Building a neural network model involves two main phases.

• Definition of the network configuration

• Iteratively training the model

1. ***Neural Network without Transformation:-***

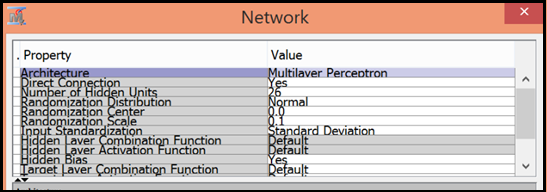
We added the Neural Network node to the Impute Node:-



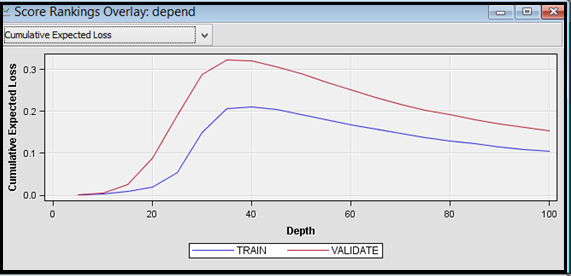
We set the *Hidden units* property to yes to create hidden units variables in our scoring data. We also set the *Standardization* property to Yes create standardization variables in our scoring data.

In the Neural Network node, in the Properties Panel, under the Train properties, we selected the ellipses that represent the value of *Network*:-

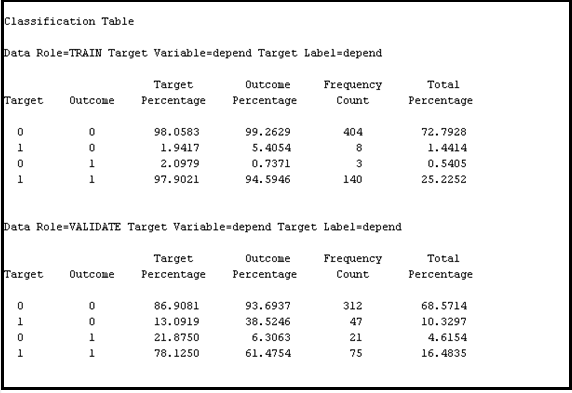
* *Direct Connection* is set to Yes. This allows network to have connections directly between the inputs and the outputs in addition to connections through the hidden units.
* The *Number of Hidden Units* is set to 26. Thus, it trains a multilayer perceptron neural network with 26 units on the hidden layer.



After executing the Neural Network Node, in the result window, under the Score Rankings Overlay window. Cumulative Expected Loss graph shows a decrease in the loss after 35% depth in both the datasets.



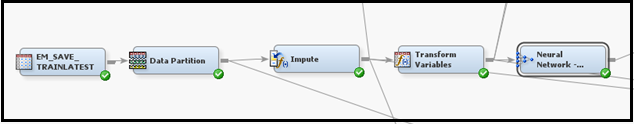
For the Neural Networks model too, the expected loss is a function of both the probability of a user booking or not booking in the given site and the estimated cost associated with each corresponding outcome as defined in the cost matrix.



The classification table shows the frequency of the misclassification errors for both Training and Validation data.

1. ***Neural Network with Transformation:-***

We added the Neural Network node to the Transform Variables Node:-

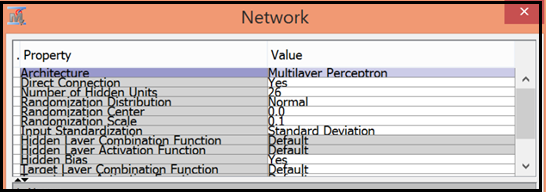


We set the *Hidden units* property to yes to create hidden units variables in our scoring data. We also set the *Standardization* property to YES create standardization variables in our scoring data.

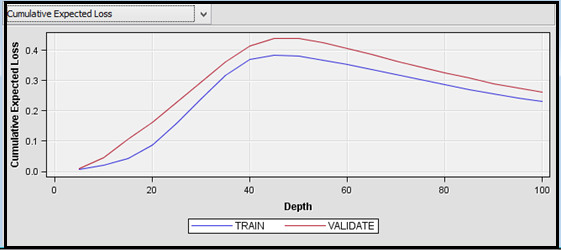
In the Neural Network node, in the Properties Panel, under the Train properties, we selected the ellipses that represent the value of *Network*:-

• *Direct Connection* is set to YES. This allows network to have connections directly between the inputs and the outputs in addition to connections through the hidden units.

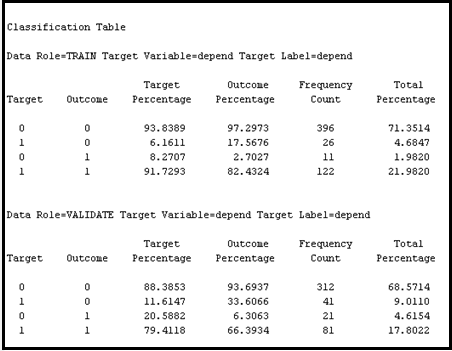
• The Number of *Hidden Units* is set to 26. Thus it trains a multilayer perceptron neural network with 26 units on the hidden layer.



After executing the Neural Network Node, in the result window, under the Score Rankings Overlay window. Cumulative Expected Loss graph shows a decrease in the loss after the 45% depth in both the datasets.



As explained above, the expected loss is a function of both the probability of a user booking or not booking in the given site and the estimated cost associated with each corresponding outcome as defined in the cost matrix.



The classification table shows the frequency of the misclassification errors for both Training and Validation data.

### **Comparison between Neural Network with Transformation and without Transformation:-**

From the Classification Table, we observed that the performance of the model has increased after transforming the data. The number of errors misclassifying 1 as 0 has decreased from 47 to 41 in the Validation data when we transformed the data. Consequently the number of records predicting 1 as 1 increases from 75 to 81 when we transformed the data.

# **Part IV. Model Comparison and Champion Model Evaluation**

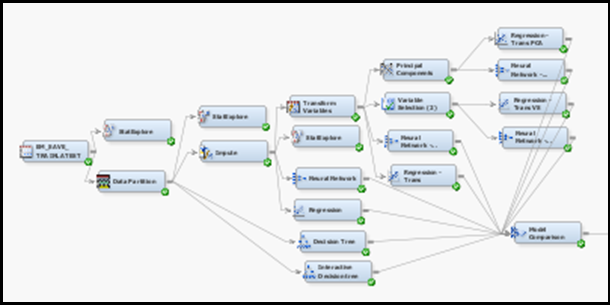
## **Model Comparison**

1. **Compare the above four models you tried, and select a champion model. When evaluating the model performance, try to use confusion matrix as the main evaluation criterion. And let’s use a cost 5 for misclassifying 1 as 0, and a cost of 1 for misclassifying 0 as 1.**

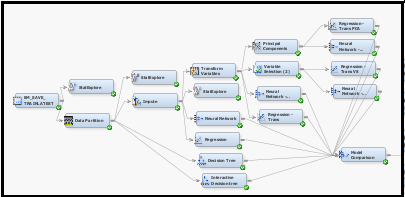
Since we now have many candidate models used for predicting whether the individual will book or not in the remainder of his current session, these models can be compared to determine a champion model that will be used to score new data. The Model Comparison node is used to compare the models that we have built so far.

We have compared models separately for sampled and unsampled data sets. After adding the Model Comparison Node, the updated workflow looks like the below screenshot:

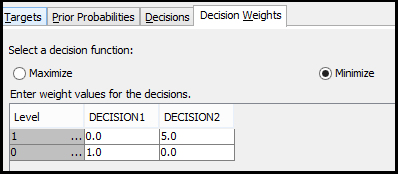
Non-sampled workflow:-



Sampled workflow:-

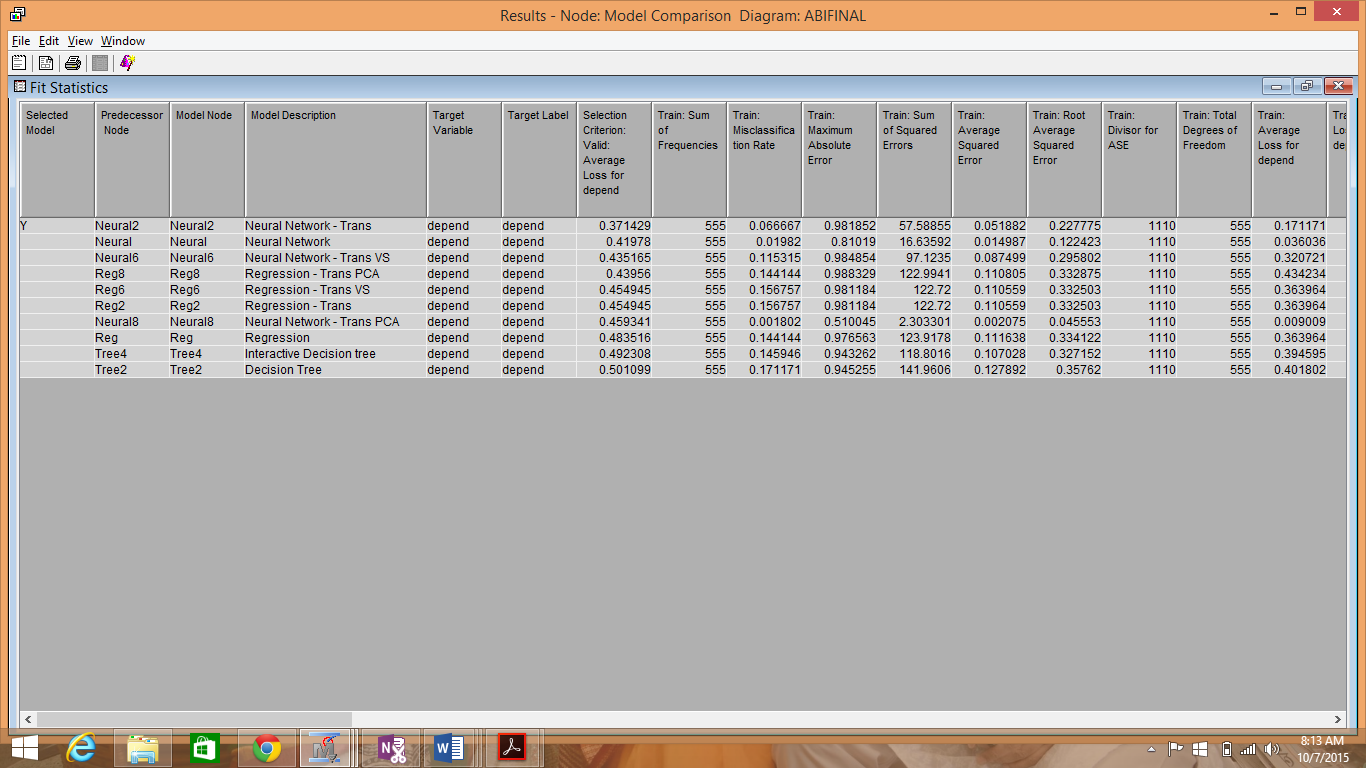


We had created a cost-based confusion matrix while creating the datasource:

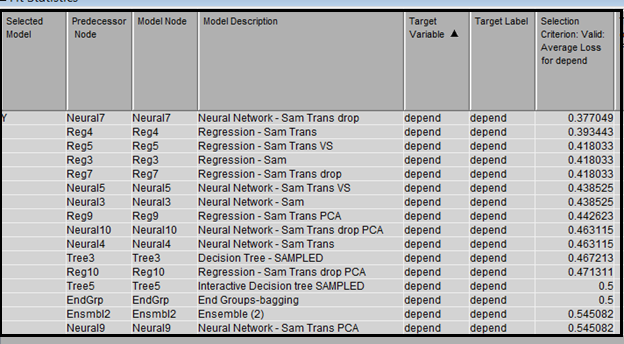


Since the confusion matrix is used by the Model Compare node for comparing the models, the *Selection criteria* is automatically set to *Average Loss for depend for the Validation data set*.

After the Node was executed, the fit statistics window shows that the champion model selected by the Model Comparison node using non-sampled data is the ‘Neural Network after transforming skewed variables’.



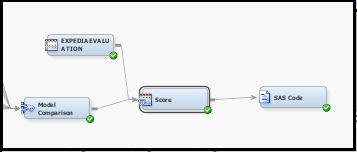
The champion model selected by the model comparison node after sampling data is ‘Neural Network after transforming skewed variables and applying subjective variable selection’.



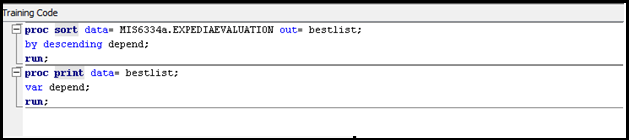
## **Scoring New Data**

1. **Score the new evaluation dataset -- expediaevaluation.sas7bdat -- using the champion model.**

A new ExpediaEvaluation dataset is created. While creating data source, the role of the Expediatest data set is set to “*score*”. We connected the model comparison node and the score data source to the Score node in the workflow and executed the score node for scoring the new test data using the selected champion model. We then used SAS Code to print the predicted depend for ExpediaEvaluation dataset. The part of the workflow is given below:

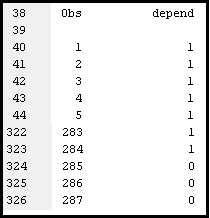


We used the following SAS Code to print the predicted depend for ExpediaEvaluation dataset:



The ExpediaEvaluation dataset has 2111 records. For both the sampled and unsampled data, the Champion model has predicted 284 1’s and rest as 0’s.

A sample screenshot of the display is given below:-



# **Part V. Improve Your Model Performance**

## **Imputation**

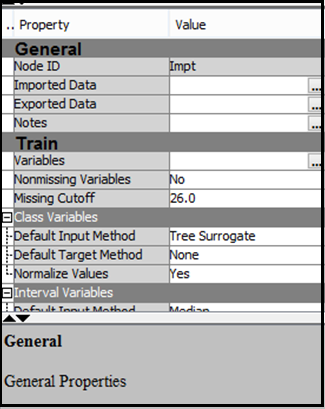
1. **Should we do imputation on all variables with missing data? And use imputed data for all classifiers?**

Data mining databases often contain observations that have missing values for one or more variables. Missing values can result from data collection errors, incomplete customer responses, actual system and measurement failures, or from a revision of the data collection scope over time, such as tracking new variables that were not included in the previous data collection schema. If an observation contains a missing value, then by default that observation is not used for modeling by nodes such as Neural Network, or Regression. However, rejecting all incomplete observations may ignore useful or important information which is still contained in the non-missing variables. Rejecting all incomplete observations may also bias the sample, since observations that missing values may have other things in common as well.

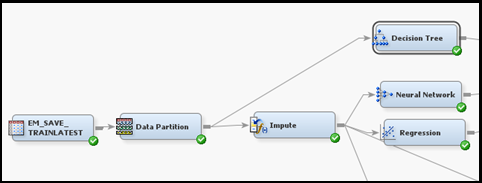
The Impute node is used to replace missing values in data sets that are used for data mining. The median is less sensitive to extreme values than the mean or midrange. Therefore, we have used the Impute node to replace Interval missing values with the Median value which is more suitable for our dataset which has a number of skewed variables. We have used the Tree Surrogate rule for replacing the missing class values.

While imputing missing values ensures that useful information is not lost, replacing missing values can greatly affect a variable's sample distribution. Imputing missing values with mean, median or other specified values may often lead to creation of a data distribution that does not accurately represent the original data distribution of the given variable, especially if the proportion of the missing values in a variable is substantial. Thus, it is often appropriate to reject variables having missing values over a threshold.

We decided on a threshold of 26% or 263 missing values (out of 1010 total observations of a variable) for a variable to be rejected. Thus on that basis, we set the value of the Missing Cutoff of the Impute node to be 26.0. This setting rejects the variables which have the missing values that are greater than 263. We observed that x38 out of the class variables and x32 out of the interval variables are rejected.



Decision Trees as can handle missing values themselves through various mechanism. SAS EM gives us the option of specifying the maximum number of *surrogate rules* that the Decision Tree node seeks in each non-leaf node. The first surrogate rule is used when the main splitting rule relies on an input whose value is missing. We have specified the *Number of Surrogate Rules* node property to be 4. This specification enables SAS Enterprise Miner to use up to four surrogate rules in each non-leaf node if the main splitting rule relies on an input whose value is missing. The Interactive decision tree assigns missing values to the branch that maximizes purity or logworth. Hence we do not need to impute or replace missing values for Decision Trees. However as mentioned above, models like Regression and Neural Networks need imputation of missing values. Hence we have added the Impute node only for the Regression and Neural Network model and not for the Decision Tree in our process flow.



## **Variable Transformation**

1. **Skewed data? Data with high variance?**

A dataset for modelling is perfectly balanced when the percentage of occurrence of each class is 100/n, where n is the number of classes. If one or more classes differ significantly from the others, this dataset is called skewed or unbalanced. Classification methods are generally not designed to cope with skewed data, thus, various action have to be taken when dealing with imbalanced data sets. The skewed data thus needs to be transformations in order to improve the fit of the model.

We however had decided to check for skewness and variance only after imputing the missing values as imputing the missing values with some other values like median etc. may change the data distribution of variables. We thus first imputed the missing values, checked for skewness and variance, transform the variables as needed and then used the Regression and Neural Networks models. As Decision Trees are insensitive to Skewness etc., no transformation is needed to be done on the data before applying this model.

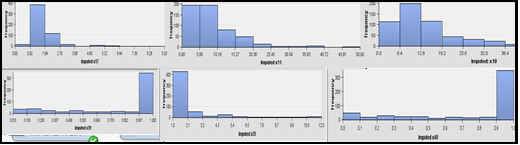
As it is difficult to conclude whether a given variance is large or not, we considered Coefficient of

Variation to measure variance. We calculated the Coefficient of Variation by dividing Standard Deviation with Mean; the values of which we got from the *Interval Variable Summary Statistics* sections of the Output of StatExplore. We considered a threshold of 0.85, i.e. variables having Coefficient of Variation 0.85 and above have high variance and variables having Coefficient of Variation lesser than 0.85 have low variance.

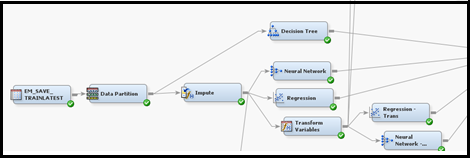
We found that x7, x9, x13, x14, x16, x18, x19, x22, x25, x26, x28, x29, x31, x34 and x36 have high variance or standard deviation. As standard deviation is directly proportional to skewness, one way to reduce variance would be to reduce the skewness of the above mentioned variables by applying the transformation Log10. We clicked on the *Formulas ellipses* in the *Train* properties of the node to observe the data distribution of the input variables.

As expected, we observed that the above mentioned variables have a highly skewed distribution and thus we selected Log10 as the transformation Method in the Variables – Trans window for these variables. Log transformations are used to control skewness.

We also observed that the distribution graph of few other variables are heavily skewed – x10, x11, x17, x35, x39 and x40.



We thus applied Log10 transformations on the above mentioned variables as well.



We observed performance enhancement in both Logistic Regression and Neural Network models after applying Log10 transformation to the data. The average loss for the validation data for neural networks and logistic regression is $0.41978 and 0.483516 respectively. However, the average loss for neural networks and logistic regression after transforming the data, goes down to $0.371429 and $0.454945 respectively. Hence, the performance of both models get enhanced.

## **Variable Reduction**

1. **Do we need all 40 attributes for prediction? If no, how about removing some variables?**

We considered that all the 40 attributes are not needed for prediction and some need to be dropped due to the following reasons:-

• *Curse of Dimensionality* - Convergence of any estimator to the true value is very slow in a high dimensional space

• *Redundant data* – A number of variables in the data set provide very similar information or are deduced from each other thus giving no new information.

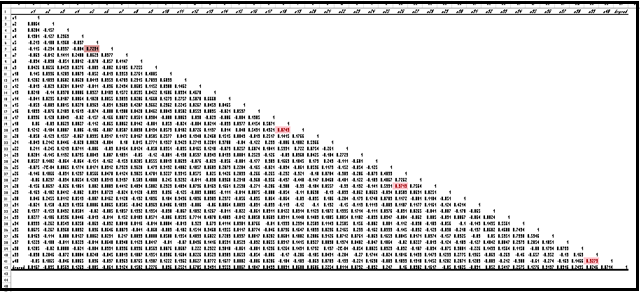
We have reduced the attributes using the following techniques:-

1. ***Qualitative & Correlation analysis on the input variables:-***
2. Variables X5 and X6 represent ‘household size’ and ‘whether the user has children or not’ respectively. In the age of nuclear families, we felt that household size and having children are highly correlated and we can thus reject x6, as x6 to some extent can be inferred from x5. We also observed that the correlation between X5 and X6 is 0.7291 which is quite high, thus indicating that one of the variables can be dropped from the analysis.

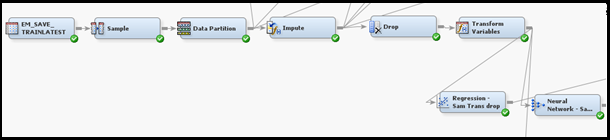
b) Variables X18 and X19 represent ‘Total no. of sessions visited of all sites so far’ and ‘Total minutes of all sites’ and are thus highly correlated with each other with the correlation value of 0.8749. As we observed that the correlation between X19 and the target variable is higher than the correlation between X18 and the target variable, we decided to retain X19 and drop X18 from the analysis.

1. Variables X26 and X29 representing ‘Percentage of total hits are to this site’ and ‘No. of sessions start with this site/total sessions of this site’ are highly correlated with each other with the correlation value of 0.9713. The correlation between x29 and the target variable is higher than the correlation between X26 and the target variable. We thus decided to retain X29 and drop X26 from the analysis.
2. Similarly, variables X39 and X40 represent ‘Hits to this site/ hits to all sites in this session’ and ‘Minutes to this site/total minutes in this session’ and are thus highly correlated with each other with the correlation value of 0.927879. The correlation between x40 and the target variable is higher than the correlation between X39 and the target variable. . Hence we decided to retain X40 and drop X39.

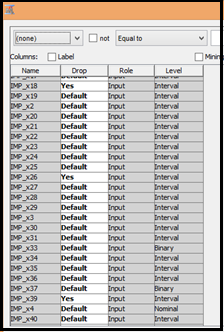
The below table shows the correlation of each variable with every other variable in the data set. We have highlighted the correlation values that we have considered as high.



We thus added a Drop node after the Impute Node in Sampled workflow:-



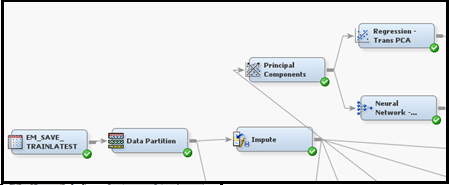
We dropped variables X5, X18, X26 and X39.



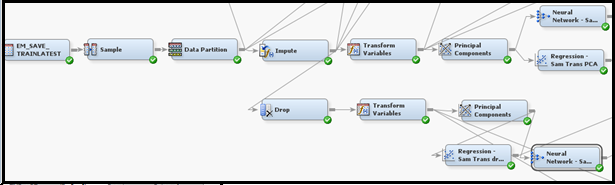
After dropping these variables, the average loss for Neural Networks (in the sampled workflow) decreases from $0.463115 to $0.377049 and it became the champion model.

***ii) Principal Component Analysis (PCA)*** – is a feature reduction technique where the transformed features are the linear combinations of the original features.

Applying PCA in the non-sampled workflow:-



Applying PCA in the sampled workflow, both before and after dropping variables based on qualitative analysis:-



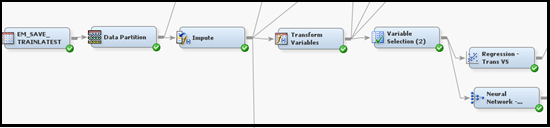
However, applying PCA did not bring about the expected performance enhancements of our models. In the non-sampled workflow, PCA improved the performance of only the Logistic Regression model, whereas it deteriorated the performance of all models in the sampled workflow.

As Principal components are uncorrelated linear combinations of the original input variables; and depend on the covariance matrix or the correlation matrix of the original input variables, PCA technique is suitable for reducing the number of interval variables. As our data is a mix of interval and class variables, we think that PCA technique is not suitable for the data, which is why the performance enhancement was not as expected.

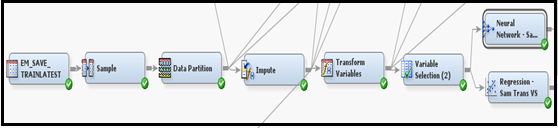
1. ***Using the Variable Selection node***

Variable Selection node reduces the number of inputs by setting the status of the input variables that are not related to the target as rejected. Although rejected variables are passed to subsequent nodes in the process flow, these variables are not used as model inputs by a successor modeling node.

Applying Variable Selection node in the non-sampled workflow:-



Applying Variable Selection node in the sampled workflow:-



In both the sampled and non-sampled work flow, 21 variables are rejected by the Variable Selection node. In non-sampled workflow, the performance of Logistic regression model increases and performance of neural network model decreases, after applying the Variable Selection node. In the sampled workflow, the performance of the Neural Network increases and the performance of Logistic Regression decreases, after applying the Variable Selection node. Hence, we did not observe any clear trend with respect to model performance on applying the Variable Selection node.

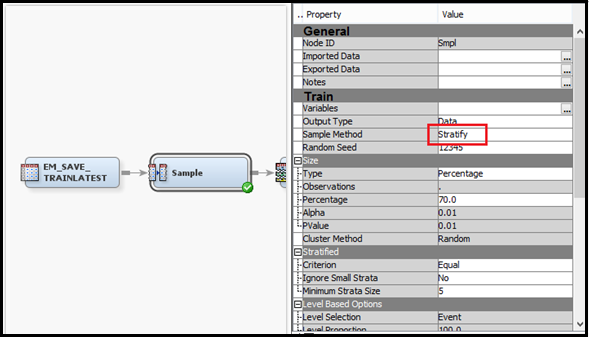
### **Comparison between the 3 Variable Reduction Techniques:-**

We observed that out of the 3 variable reduction techniques used, variables dropped on the basis of ‘Qualitative & Correlation analysis on the input variables’ resulted in best performance of the models.

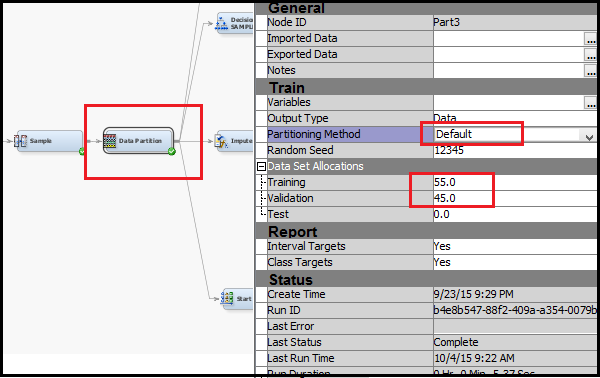
## **Sampling**

1. **Is the dataset too large in terms of number of records? If you think so, how about sampling for a smaller size? Will this actually help with the prediction performance?**

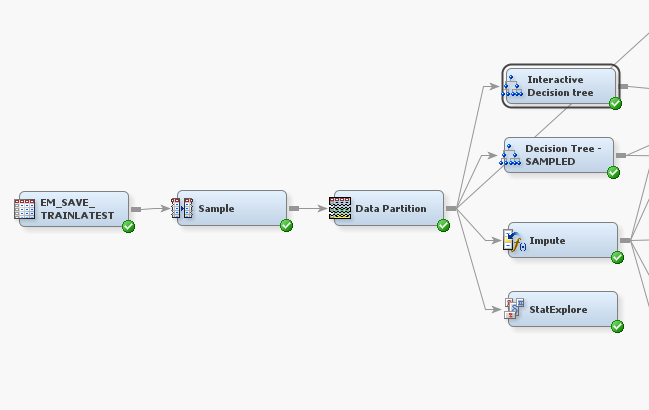
The size of the dataset is 1010 which is not considered a very large data set. However we considered sampling the data to mitigate the effect of an unbalanced data. A training dataset can consist of disproportionately high number of one value than the other and our dataset is an example of such an unbalanced dataset with the proportion of one’s being 13% and proportion of 0’s being 87%. Such unbalanced dataset when used to create a classifier often produces biased models. The prediction of such biased models have overestimated accuracy which needs to be fixed. There are several ways to fix this data imbalance and the most common method is *sampling*. Here we have under-sample the most frequent value- in our dataset which is 0’s and over-sampled the less frequent 1’s. With this balanced dataset, the models would be able to predict with better accuracy. Sampling also significantly decreases model training time and if it sufficiently represents the source dataset then the prediction models which are created on the sample dataset can be applied on the complete data set as well.

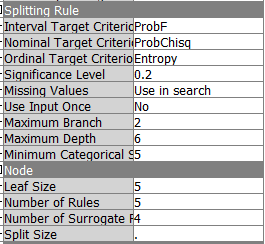


1. We have added a Sample node to the dataset. Here we used *stratified random sampling* method, where the population consists of N elements and is divided into H groups called strata. Each element of the population are assigned to only one stratum and we obtain probability sample from each of these stratum. Stratified sampling method provides greater accuracy and required smaller sample thus saving money. It also reduces selection bias and ensures each subgroup within the population receives proper representation within the sample. We have set *Stratified Criteria* to ‘*Equal*’ in order to balance the 0s and 1s in the dataset.
2. The Data Partition node was added by setting the training and validation property in Data Set Allocation to 0.55 and 0.45 respectively.

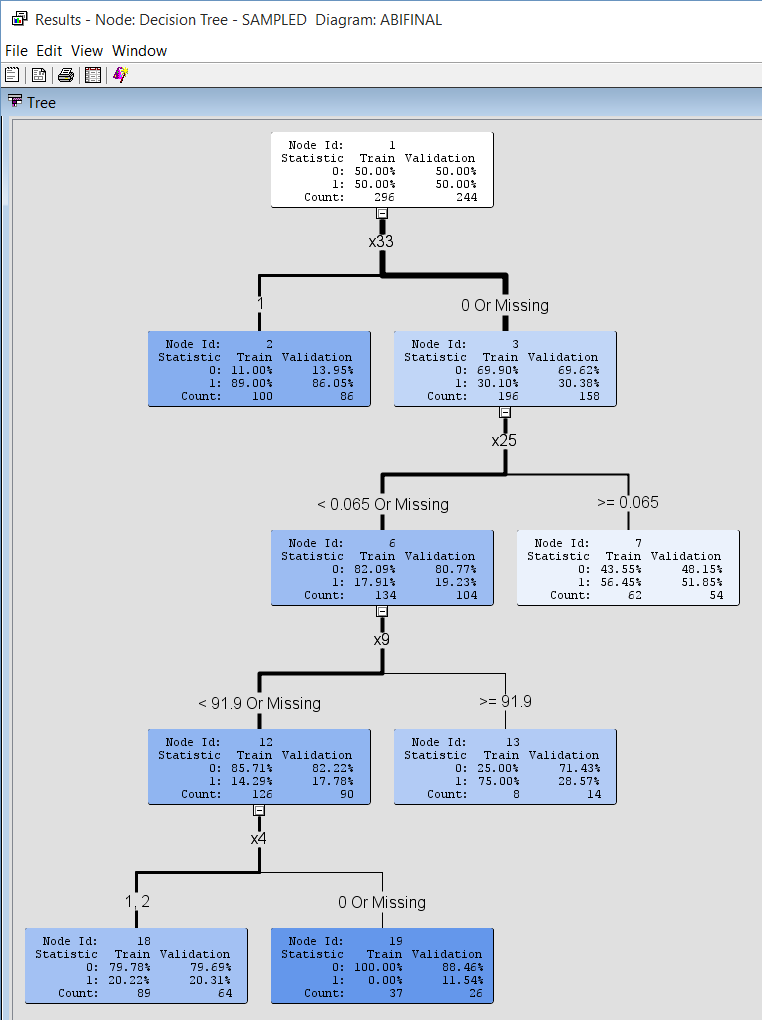


1. We added a ***Decision Tree*** node to the Data Partition node with the settings similar to the ones used in the Decision Tree used on un-sampled dataset.

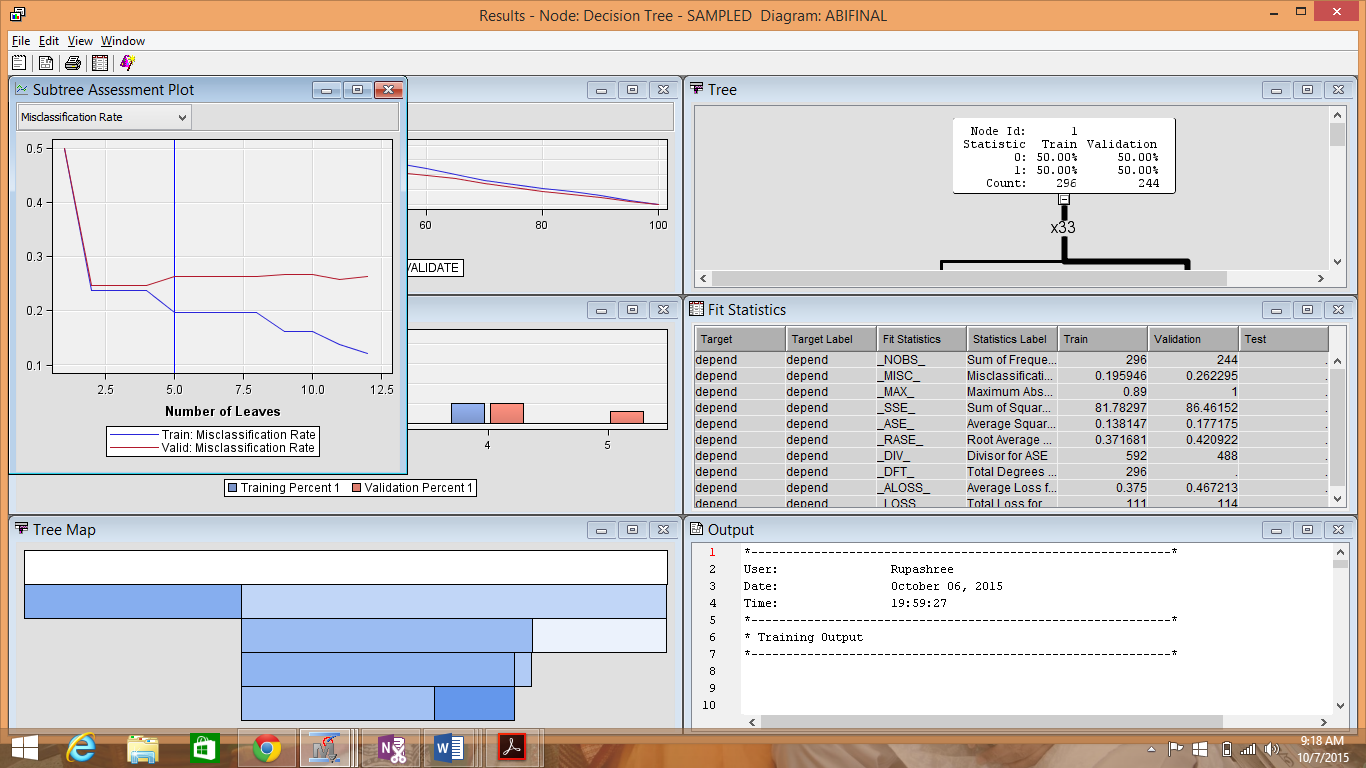




When we executed the decision tree with the above properties we found the following results:-

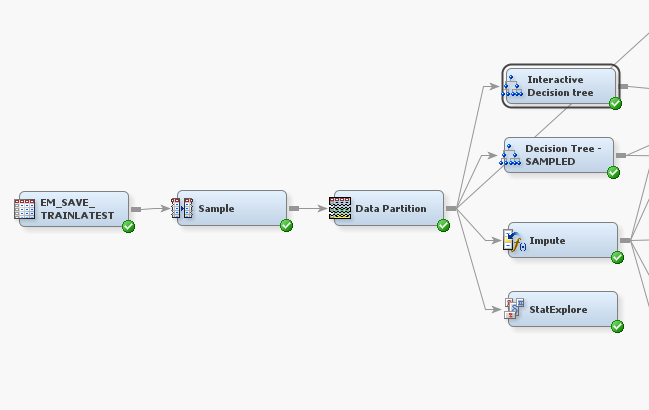


The optimal tree that is formed, initially split on the attribute bookgc with highest logworth. After that, the second node has split on the attribute booksh and the third split happenned on the node 6 with the attribute minutehl. The final split happened at the attribute edu.



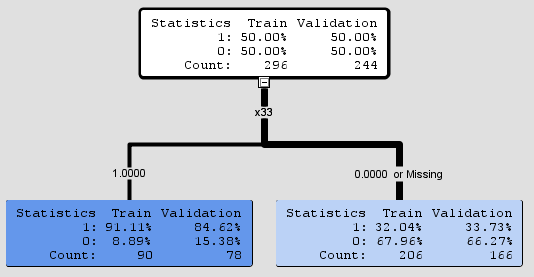
The subtree assessment plot for the Misclassification Rate showed that the error rate for both Training and Validation data decreases till 2nd node. It then stabilizes for both the datasets till 4th node before decreasing it further for training dataset and increasing for validation dataset. This indicating that overfitting after the 5th node.

1. We then added an ***Interactive Decision Tree*** to the Data Partition node

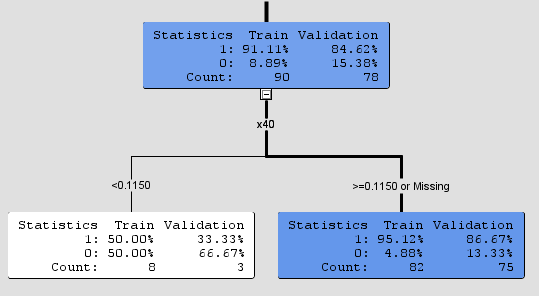


The results are shown below:-

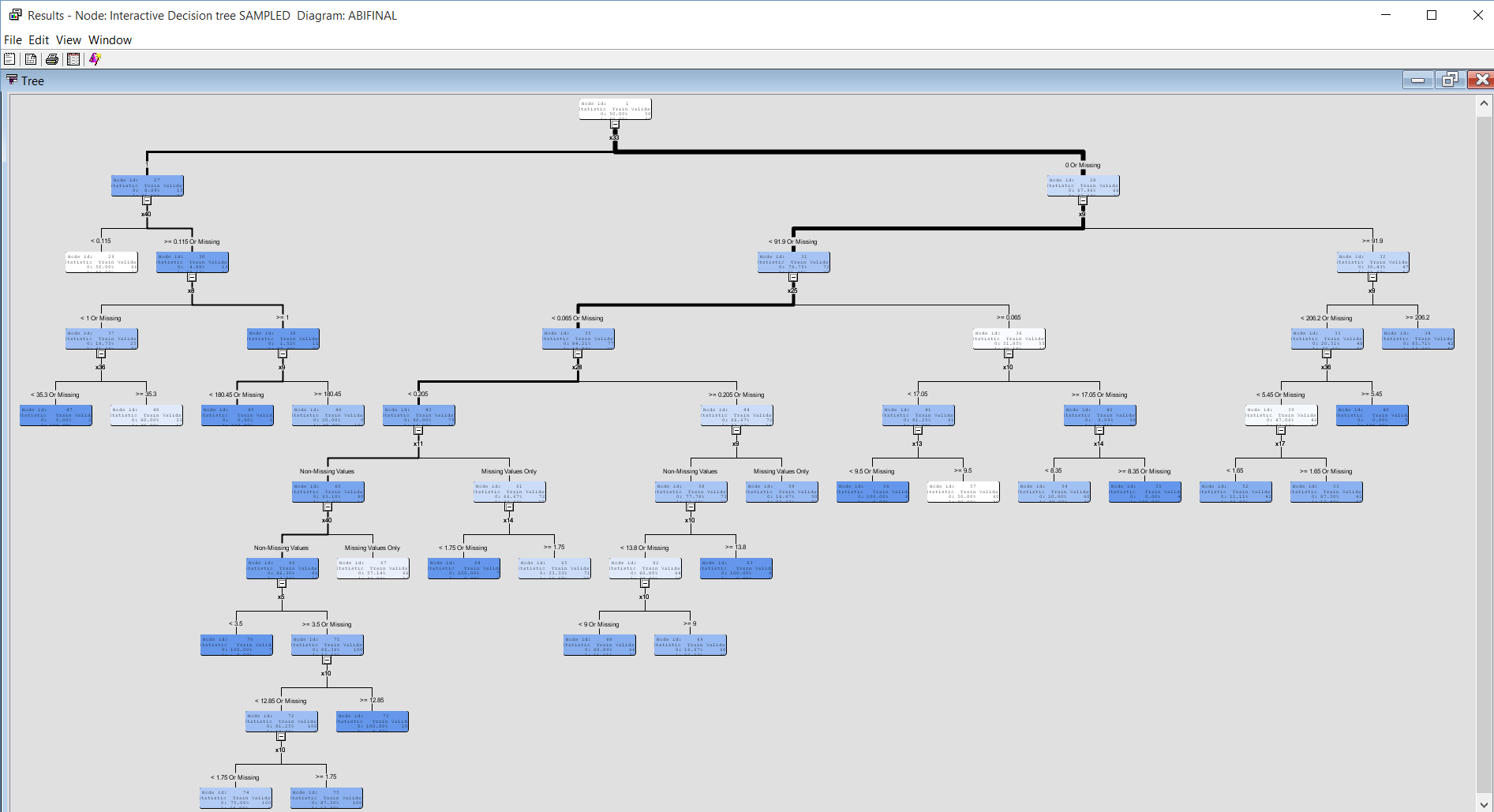
On the first manual split the logworth value was highest for the bookgc variable.



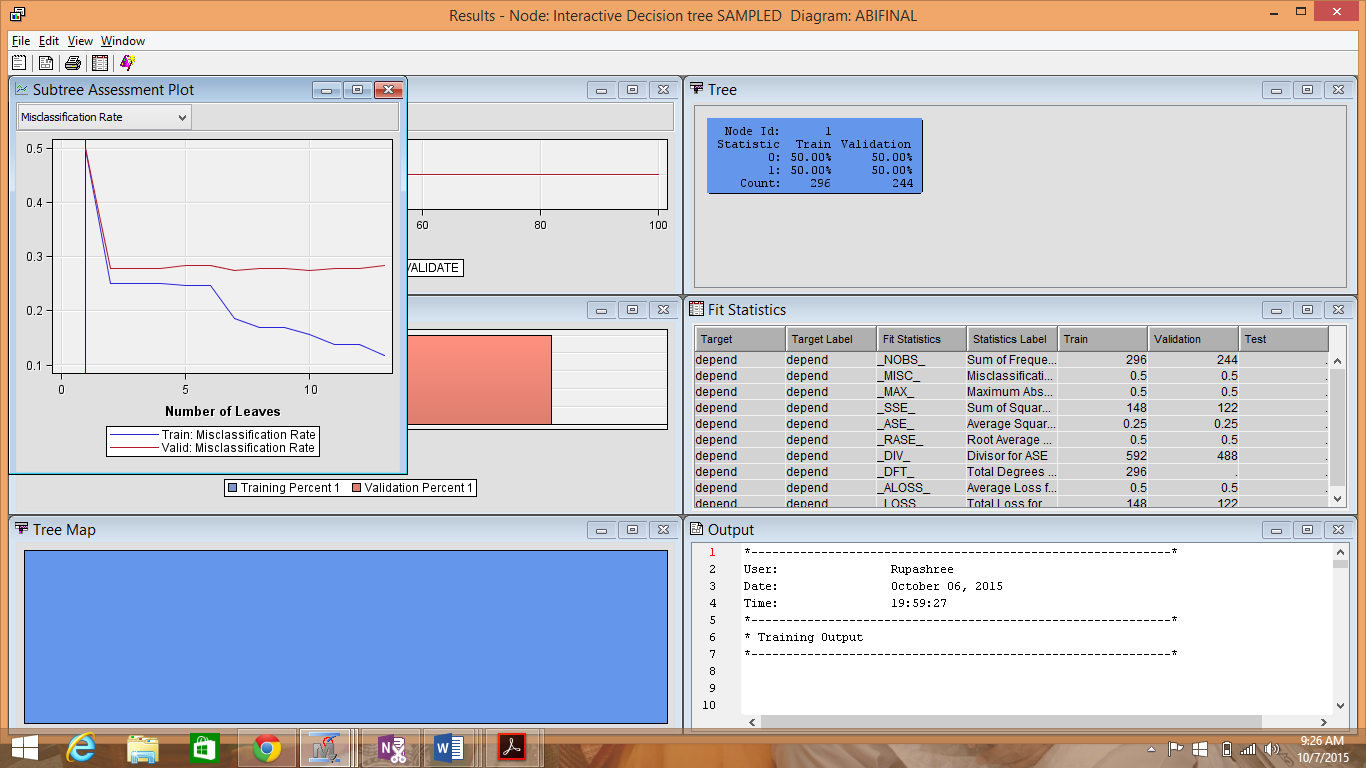
The lower left node was selected for the second manual split, as the logworth value for minutsch was found to be second highest.



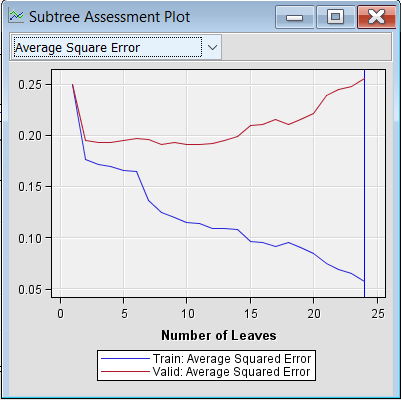
After further continuing this method to split a few nodes, we trained the tree using Train Node option and the tree formed is shown below:



We observed that the Misclassification Rate decreases at a high rate till the 3rd node for both training and validation datasets, after which the error rate decreases further for training dataset and increases slightly for validation dataset. This indicates overfitting of data after the 3rd node.



We observed that the average square error has a downward trend. After the 3rd leaf, the average square error for the training dataset has reduced drastically. And the Average square error in the validation set increases indicating over-fitting of data after the 3rd leaf.

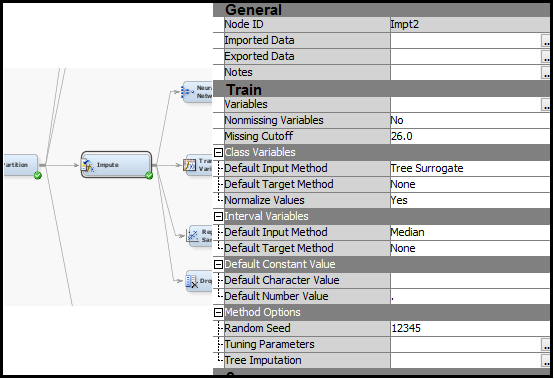


1. We added the Impute node with the following settings:-

For class variables, we selected *Tree Surrogate* as the *Default Input Method*, enabling SAS Enterprise Miner to build a decision tree with that variable as the target and the other input variables as predictors.

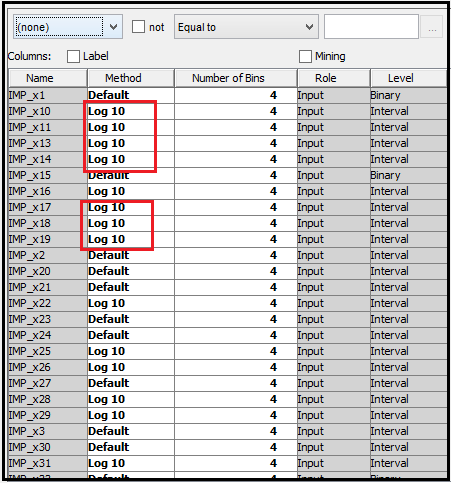
For interval variables, we selected *Median* as the *Default Input Method*, so that the values of missing interval variables are replaced by the median of the non-missing values.

We set the missing cutoff as 26% which means any variable having missing values more than 26% will be dropped.

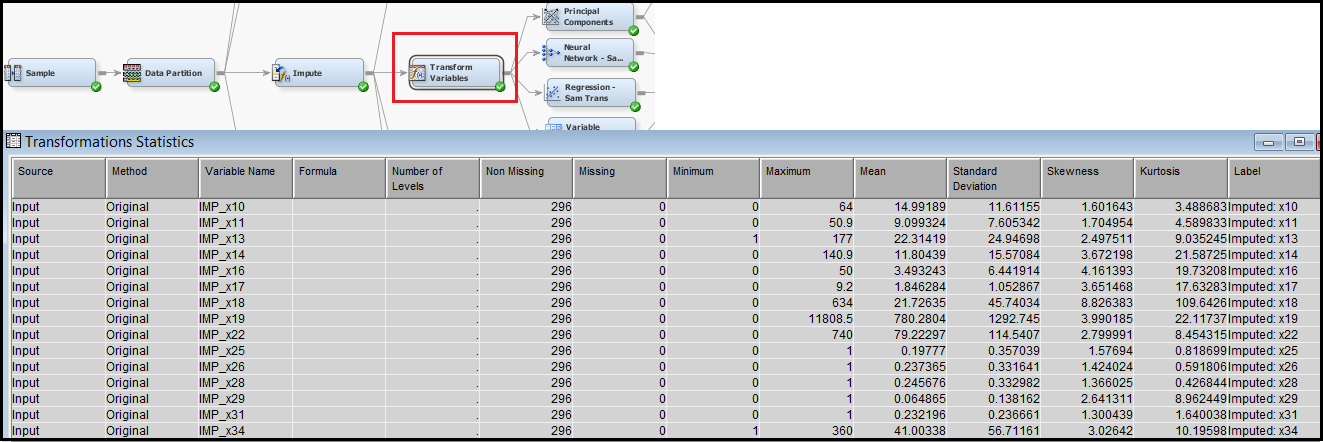


1. Variable transformations are used to stabilize variance, remove nonlinearity, improve additivity, and counter non-normality. For our dataset, we used transform variable node to remove Skewness by using log10 transformation.

We found that x7, x9, x13, x14, x16, x18, x19, x22, x25, x26, x28, x29, x31, x34, x36, x10, x11, x17, x35, x39 and x40 are skewed. We applied log10 transformations on all the variables from variable tab like in the diagram below.



The variable transformation node and its output screen are shown below:

****

We have added Neural Network and Regression nodes with and without transformations.

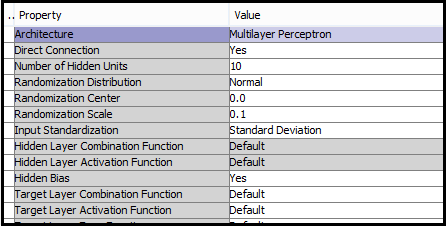
1. ***Neural networks without transformation.***

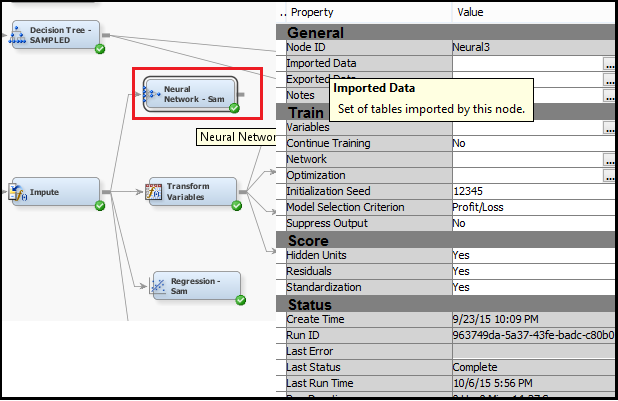
We added a Neural Network node to the Impute node and set the *Hidden units* property to yes to create hidden units variables in our scoring data. We also set the *Standardization* property to yes create standardization variables in our scoring data.

In the Neural Network node, in the Properties Panel, under the Train properties, we selected the ellipses that represent the value of *Network*:-

• *Direct Connection* is set to yes. This allows network to have connections directly between the inputs and the outputs in addition to connections through the hidden units.

• *The Number of Hidden Units* is set to 10. Thus it trains a multilayer perceptron neural network with 10 units on the hidden layer.

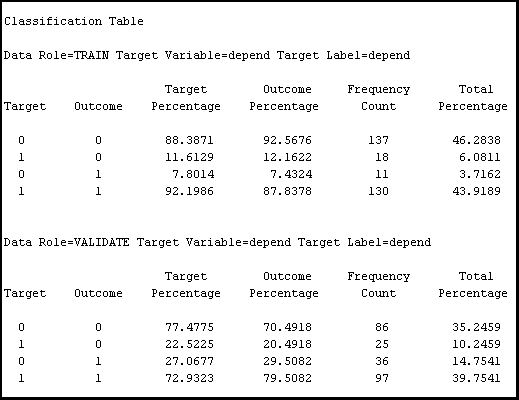
****

****

We observed the cumulative expected loss for neural networks to be a function of both the probability of a user booking or not booking in the given site and the estimated cost associated with each corresponding outcome as defined in the cost matrix.



The classification table shows the frequency of the misclassification errors for both Training and Validation data.



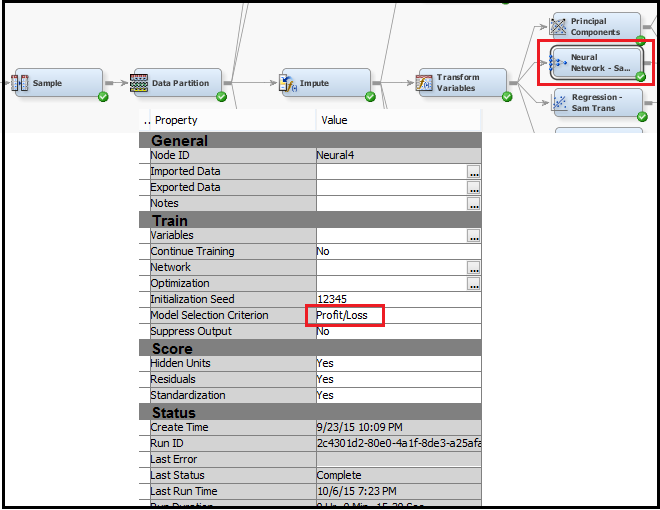
1. ***Neural networks with transformation***

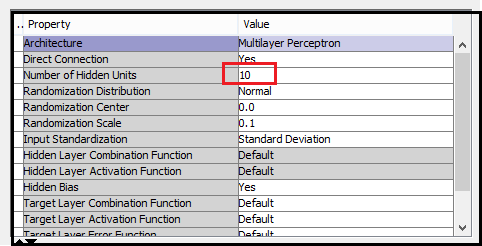
We added a Neural Network node to the Transform Variables node and set the *Hidden units* property to yes to create hidden units variables in our scoring data. We also set the *Standardization* property to YES create standardization variables in our scoring data.

In the Neural Network node, in the Properties Panel, under the Train properties, we selected the ellipses that represent the value of *Network*:-

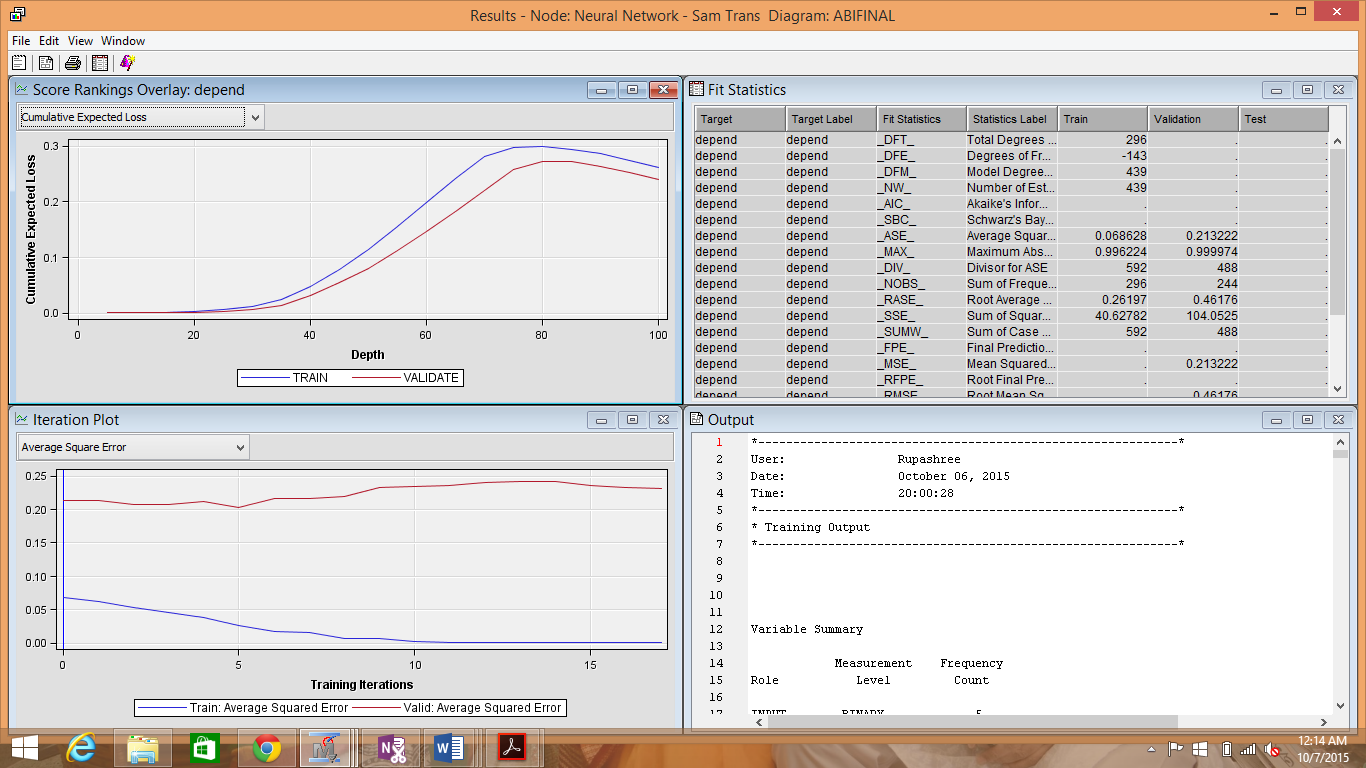
• *Direct Connection* is set to YES. This allows network to have connections directly between the inputs and the outputs in addition to connections through the hidden units.

• The *Number of Hidden Units* is set to 10. Thus it trains a multilayer perceptron neural network with 10 units on the hidden layer.

****

****

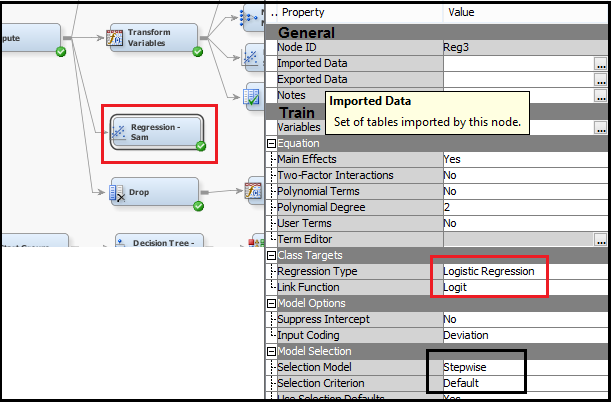
The output of the node is as follows;-



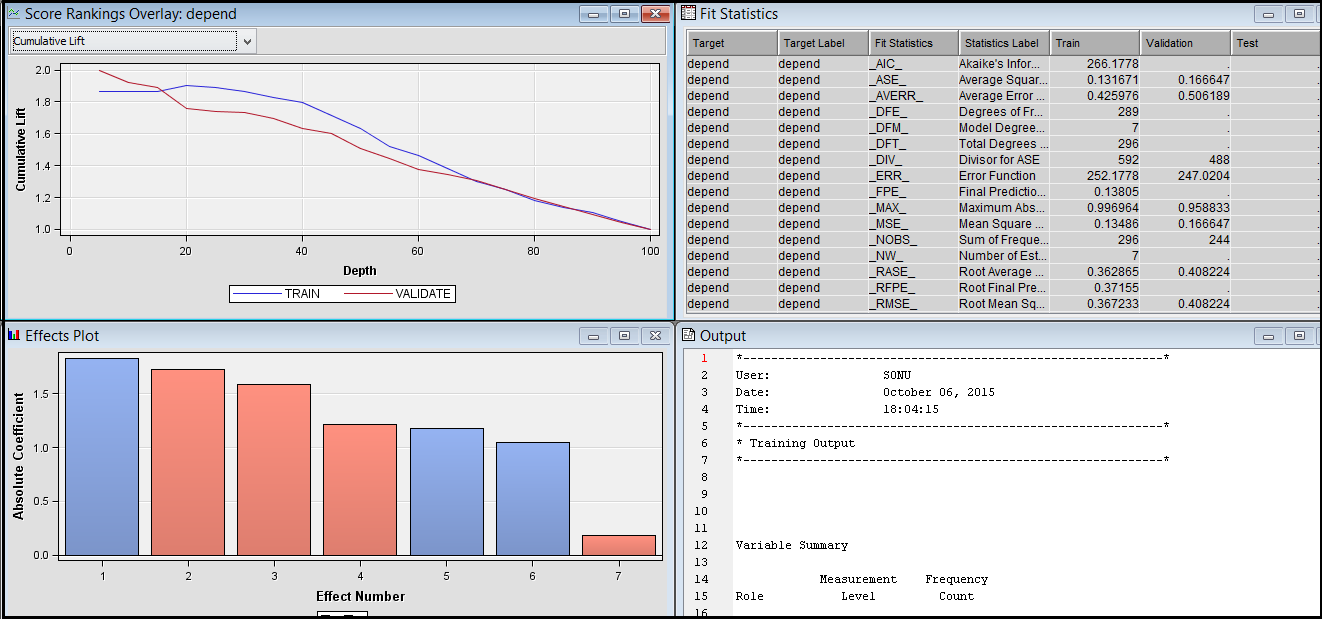
We observe that the cumulative expected loss for neural network with transformation is comparable to that of the neural network without transformation.

1. ***Logistic Regression without transformation***

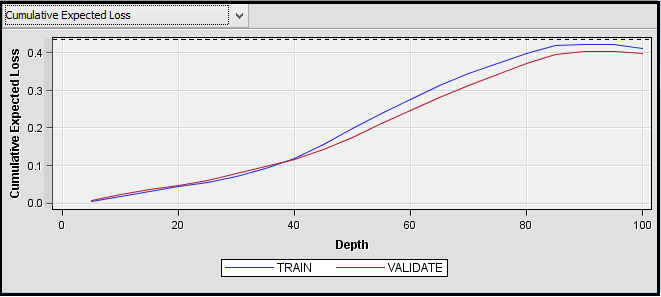
We added a Regression node after the Impute node. The node automatically performed logistic regression as the target variable in this data set is a binary variable. We selected *Stepwise* as the *Selection Model* property which begins regression with all candidate effects in the model and removes effects until the Stay Significance Level or the Stop Criterion is met and also removes effects already present in the model.

****

The output of the sampled regression node.

****

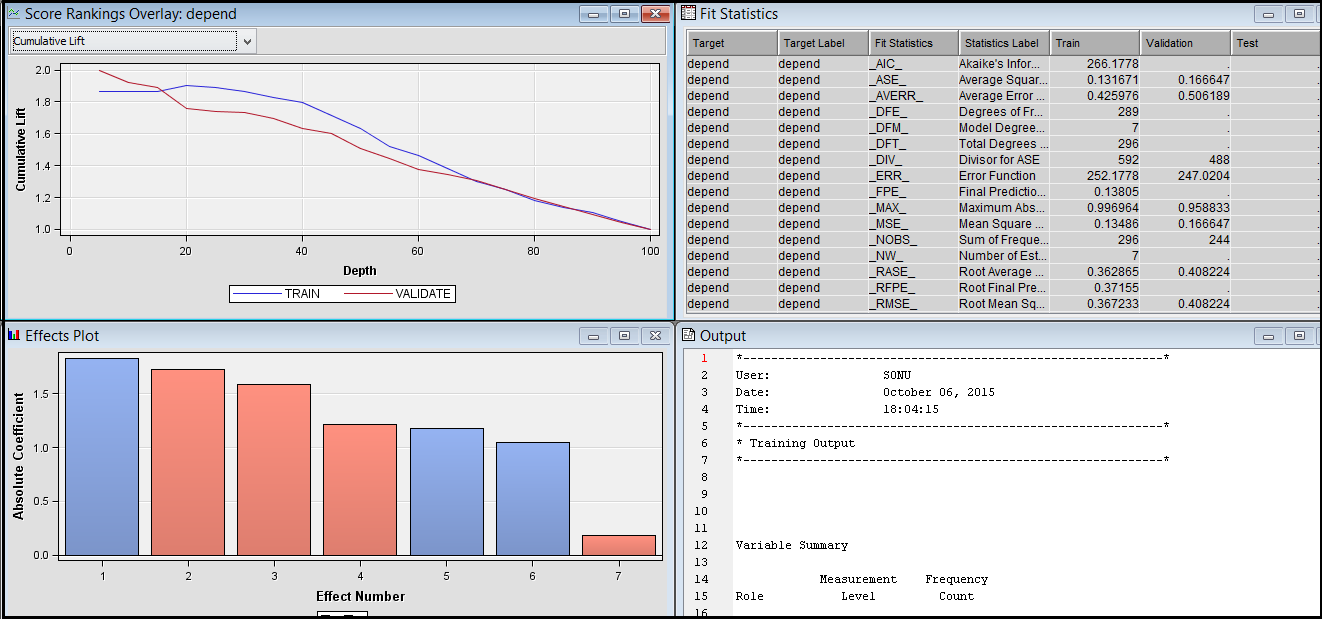
The cumulative expected loss for regression on sampled data is lesser than the regression on non-sampled as per the graph down below.



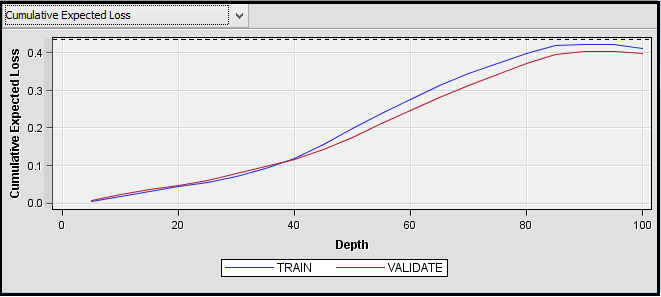
1. ***Logistic Regression with Transformation***

We added a Regression node after the Transform Variables node. The node automatically performed logistic regression as the target variable in this data set is a binary variable. As in the above case, we selected Stepwise as the Selection Model property.

The output result is as follows:-

****

The cumulative expected loss for sampled logistic regression with transformation is lesser than the sampled logistic regression with transformation. Hence the model works better after transformation of the data.



### **Comparison of Model performance with Sampled and Non-sampled data**

We observed that the performance of the Logistic Regression increases on sampling the data. The Logistic Regression on sampled data, both with and without transformation performs better than their counterparts on non-sampled data. However the performance of Neural Network deteriorates with sampled data.

## **Ensemble & Bagging**

1. **Can ensemble help? How will you do it?**

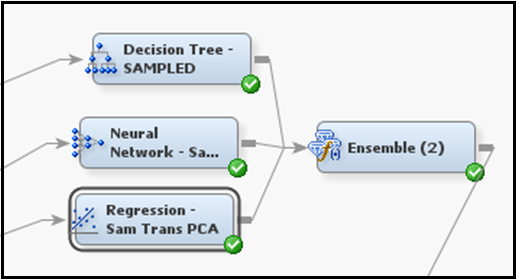
We have tried to do Model Enhancement by

a) ***Ensemble by ‘Voting;-***

The Ensemble node creates new models by combining the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple predecessor models. It thus creates a combined model to improve the stability of disparate nonlinear models, such as those created from the Neural Network and Decision Tree nodes.

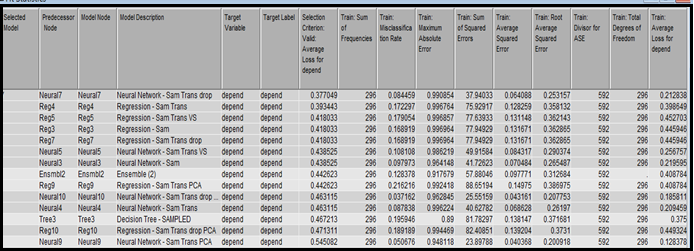
It is important to note that the ensemble model can only be more accurate than the individual models if the individual models disagree with one another. In our sampled workflow, we have used the 3 most weak performing models namely; Decision Tree, Neural Network after transformation and PCA and Logistic Regression after transformation and PCA for ensemble. The component models from the three complementary modeling methods are integrated by the Ensemble node to form the final model solution.

Thus, each of the 3 models used, created a separate model using the same training data. The Ensemble node co-mines the posterior probabilities of the class target through ‘voting’ to create the ensemble model.



We observed that the performance of the Ensemble model is better than all the 3 individual model used:-

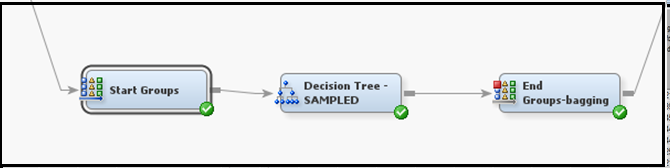
|  |  |
| --- | --- |
| **Model** | **Average Loss for Validation Data** |
| Ensemble | 0.442623 |
| Decision Tree - Sampled | 0.467213 |
| Neural Network – Sam Trans PCA | 0.545082 |
| Regression – Sam Trans PCA | 0.442623 |



1. ***Bagging***

The Bootstrap Aggregation, or Bagging mode in the Start Groups node in the Sampled workflow created unweighted samples of the active training data set for bagging. The Bagging method uses random sampling with replacement to create the n sample replicates. We had set the number of samples as 15.

The Bagging method ran the Decision Tree model 15 times over different training datasets. However the performance of the Decision Tree did not improve through Bagging. A probable reason for this is that, Bagging is based on the assumption of independence. As our data contains a number of attributes that are highly correlated and dependent to each other, Bagging did not cause any performance enhancement.

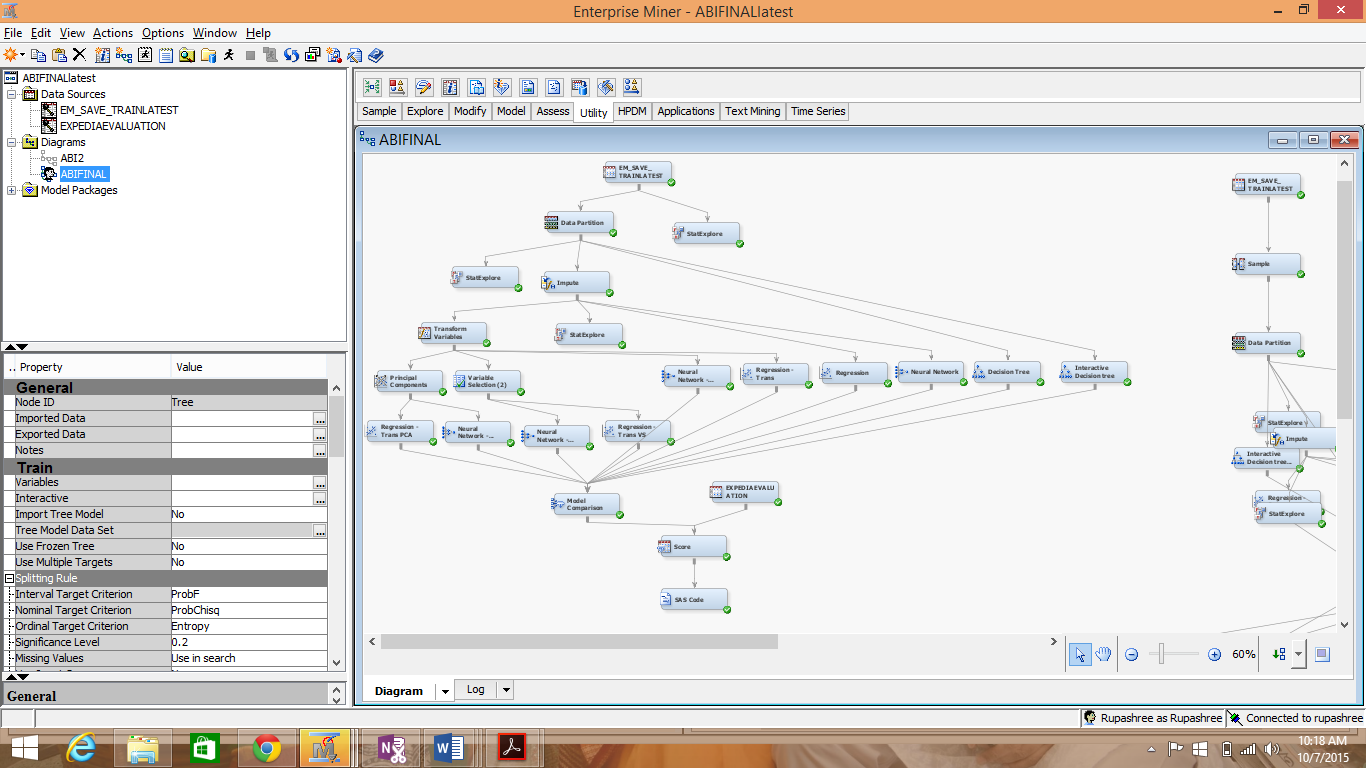


# **Part VI. Summary**

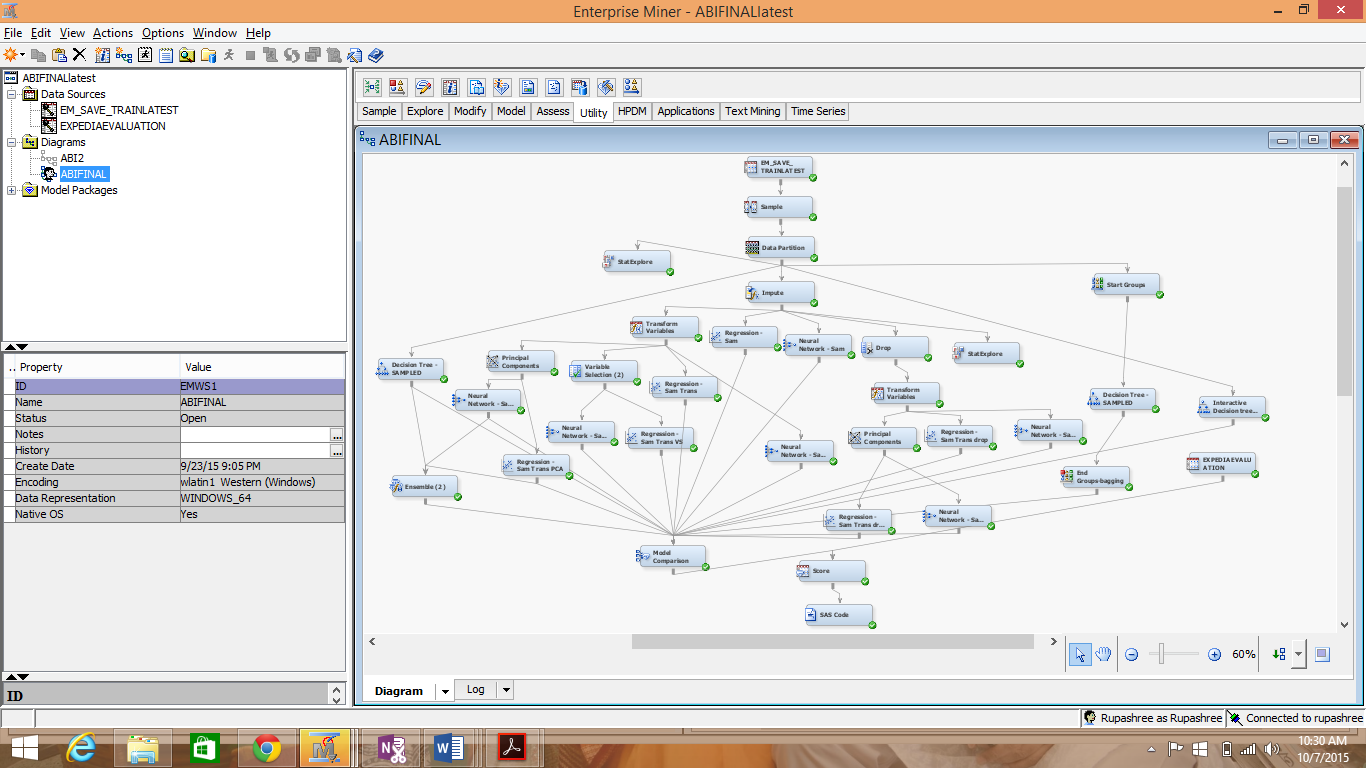
## **Final Diagrams**

1. Include the complete final diagram you get. Make sure it is legible, and if needed use several pages in print.

Workflow 1: Without Sampling



Workflow 2: With Sampling



## **Learnings**

1. **Summarize what you learned from this project. Be concise.**

This project not only provided us with additional exposure to the SEMMA process in SAS Enterprise Miner, but also gave us an opportunity to work on a real-time data set and perform predictive analysis on it from the scratch. The focus of this project was performance enhancements of the models through various techniques like missing value handling, transformations, sampling, variable reduction and ensemble.

We gained immense learnings from working on the project, which can be categorized under the following-

• Data Pre-processing - We explored the data in detail, studied the data distribution and the role of each variable and made the required modifications before using it for the predictive analysis. This step was a good learning experience for us as it gave us an opportunity to analyze the effect of each feature of the data - on the predictive analysis that we wished to do. We understood the effect of conditions like standard deviation, coefficient of variation, Skewness and kurtosis etc. on the fit of the various models to the data. We gained understanding of how sensitive each model is to conditions like Skewness, missing values etc. We brainstormed on various transformation techniques like Maximum normal, Log10 etc. to improve the fit of the models to the data.

• Variable Selection – We got hand-on experience on using PCA technique and Variable Selection node on the data. We deliberated on the suitability of these techniques to the data. We also subjectively dropped attributes by doing in-depth analysis of each variable and brainstorming their significance in determining the target variable and whether they are provide any additional information. This exercise made us appreciate the need to thoroughly understand the data before applying any kind of analysis on it.

• Better understanding of the Models – This project increased our grasp on the various models we used, by giving us an opportunity to understand the significance of each setting and experience how they change the performance of a model. We did trial and testing to determine the optimal setting for each model with respect to model performance. We also learnt to evaluate their performance by studying various graphs like misclassification graph, average square error etc.

• Model Enhancement Techniques – The project gave us an exposure on how to use Ensemble and Bagging techniques for predictive analysis. The exercise improved our understanding of the working of these techniques and scenarios where it is suitable to use these techniques.

Finally, we have compared the performance of the models under various combinations like transformation, sampling, PCA, Model Enhancement Techniques etc which has given us a good understanding on methods to improve model performance and how the effectiveness of model enhancement techniques are often data –dependent.

# **References**

1. Getting Started with SAS® Enterprise Miner™ 13.1
2. SAS Enterprise Miner Help
3. Journal of Targeting, Measurement and Analysis for Marketing (2007), Merits of interactive decision tree building — Part 2: How to do it - Bas van den Berg and Tom Breur,

<http://www.palgrave-journals.com/jt/journal/v15/n4/full/5750054a.html>

1. Variable Selection and Transformation of Variables in SAS® Enterprise Miner ™ 5.2; Kattamuri S. Sarma, Ph.D. Ecostat Research Corp., White Plains NY
2. <http://www.simafore.com/blog/bid/111124/Decision-tree-accuracy-effect-of-unbalanced-data>