An analysis on Expedia dataset for prediction ticket booking using Enterprise Miner 9.4

**Advanced BA with SAS**

*Project Purpose*

***An analysis on Expedia dataset using SAS Enterprise Miner 9.4, Base SAS and***

***Advanced MS Excel***

*Business Goal*

***To predict if the customer is going to book the ticket through Expedia website in***

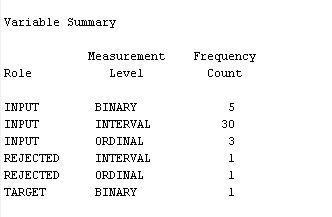
***the other half of the session***

# Part I. Basic Data Preprocessing

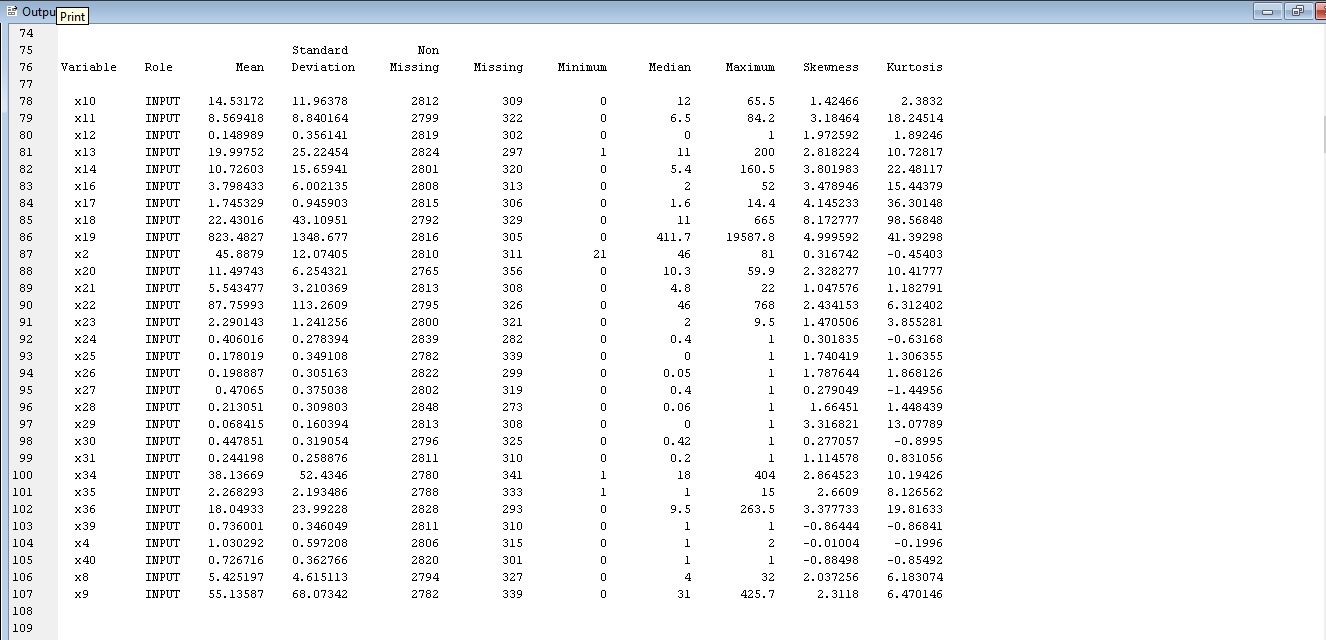
## Data Summary

Our observation is as below:

* We rejected x32(SErate) and x38 (path) since they had missing values > 50%
* All other variables have ~9-10% of missing values
* For some of the variables skewness was there

****

***Figure1: Variable summary***

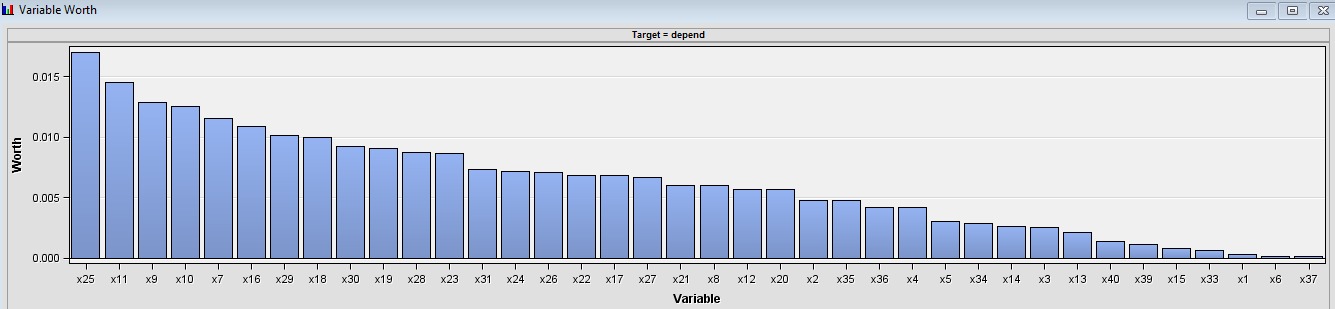
****

***Figure 2: Statistical description of variables***

## Statistical Data Exploration

Using Stat Explorer, the top 4 useful variables are:

* x25 (booksh)
* x11 (mpsesslh)
* x9 (minutelh)
* x10 (hpsesslh)



***Figure 3: Variable Histogram***

## Data Imputation

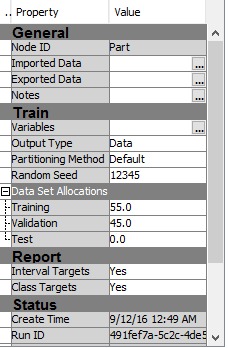
As we can see from Figure 2, all the variables have ~10% missing values, so we imputed the variables using:

* Median for interval variables
* Count for class variables

## Data Partition

Train:0.55

Validaion:0.45

****

***Figure 4: Dataset Partition***

# Part II. Building Decision Trees

We used 2 decision trees - first target being ProbChiSquare and the latter being Entropy

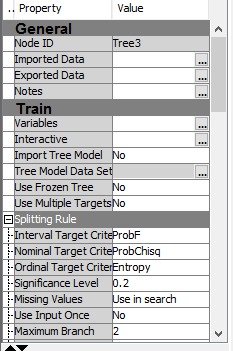
Decision Tree Using ProbChiSquare

Why ChiSquare? – It measures how well the observed distribution of data fits with the distribution that is expected if the variables are independent

Misclassification Rate:

Train: 0.102041

Validation: 0.116643



***Figure 5: DecisionTree1\_ChiSq***

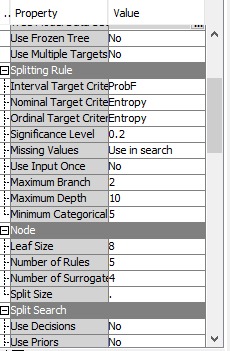
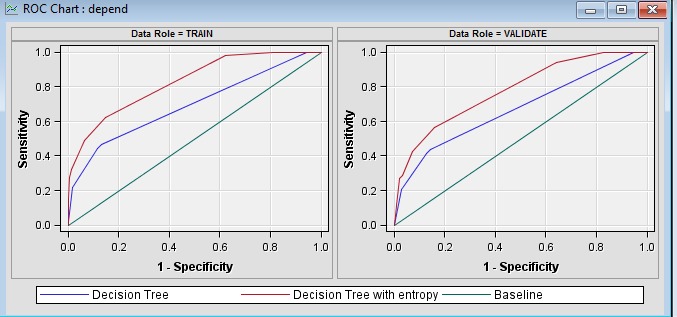
## Decision Tree Using Entropy

Why Entropy? – Entropy gives measure of impurity in a node

Misclassification Rate:

Train: 0.119534

Validation: 0.130868

***Figure 6: DecisionTree2\_Entropy (ROC side-by-side comparison)***

***\*IMP\* We did not use the Impute node before Decision Tree node because DT takes care of missing values automatically***

## Interactive Decision Tree with Explicit Pruning

With logworth decision values, we *explicitly pruned the trees* in Interactive DT1

Misclassification Rate:

Train: 0.102041

Validation: 0.116643

## Interactive Decision Tree with Manual Split Node Points

In IDT2, apart from pruning trees, we manually changed the split point of nodes according to median of particular node

Misclassification Rate:

Train: 0.119534

Validation: 0.130868

# Part III. Building Neural Networks and a Regression Model

## Data Transformation

In some of the interval variables like AwareSet, booksh, exitRate, hpsesslh, mpsesslh which were right skewed, we used the **log transformation** to reduce the skewness. Then, latter, we transformed few other interval variables like httlc, minutegc, peakrate and minutelc using the **exponential method**

## Logistic Regression

Technique used: Logistic Regression

Link Func: Logit

Model Selection: Stepwise

Misclassification Rate:

Train: 0.125364

Validation: 0.128023

## Neural Network

For Neural network, we developed two-layer direct architecture with 100 hidden neurons

Number of Tries: 6

Maximum Iterations: 100

Maximum number of links: 1000

Misclassification Rate:

Train: 0.123615

Validation: 0.130868

## HP-Neural Network

For HP-Neural network, we developed two-layer direct architecture 2 layers with 6 hidden neurons

Number of Tries: 7

Maximum Iterations: 80

Maximum number of links: 1000

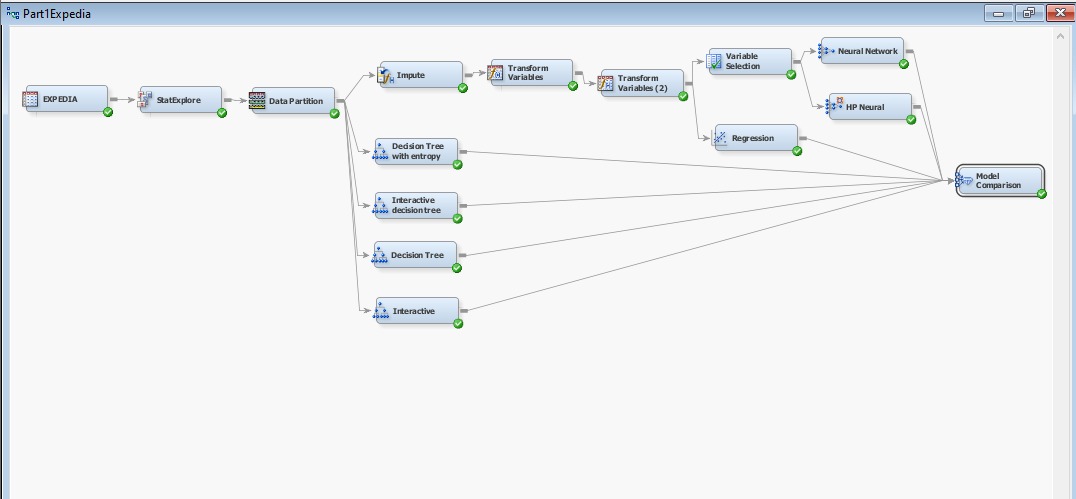
Misclassification Rate:

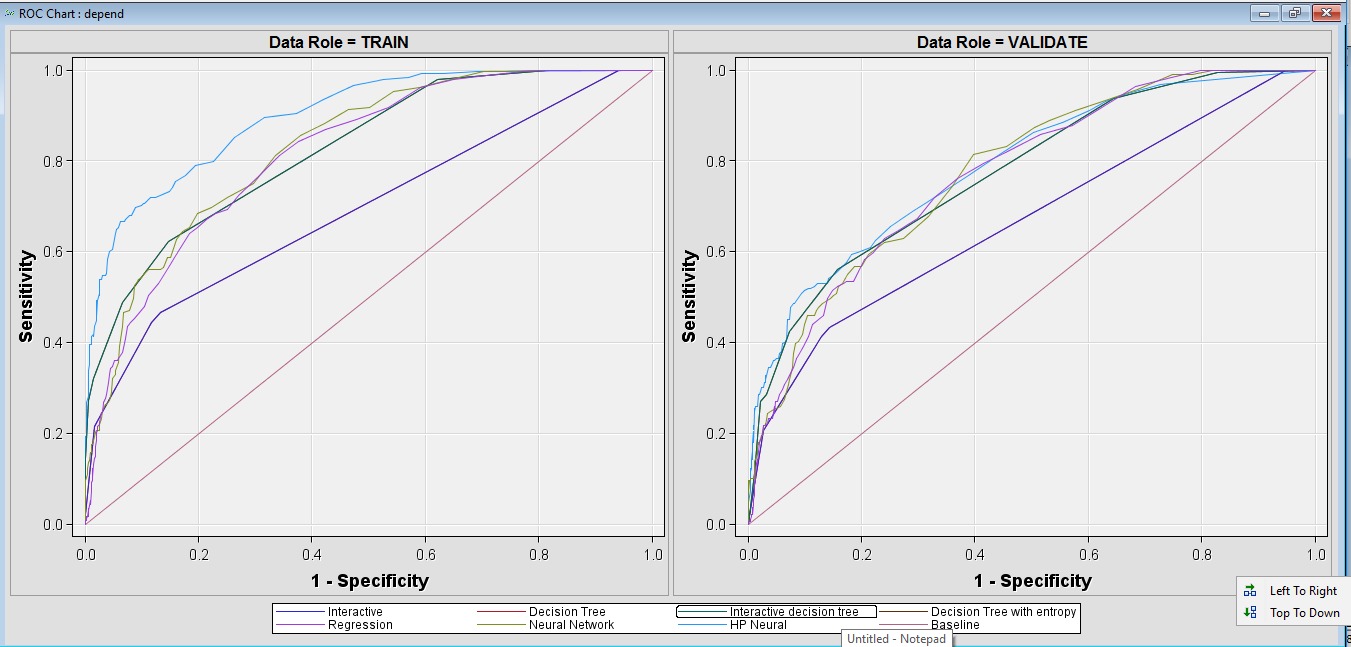
Train: 0.088047

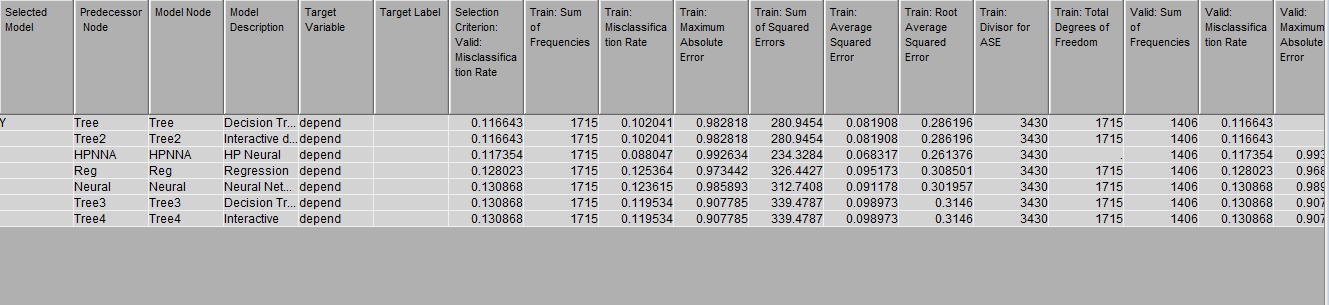
Validation: 0.117354

# Part IV. Model Comparison and Champion Model Selection

Decision Tree was our champion model

****

****

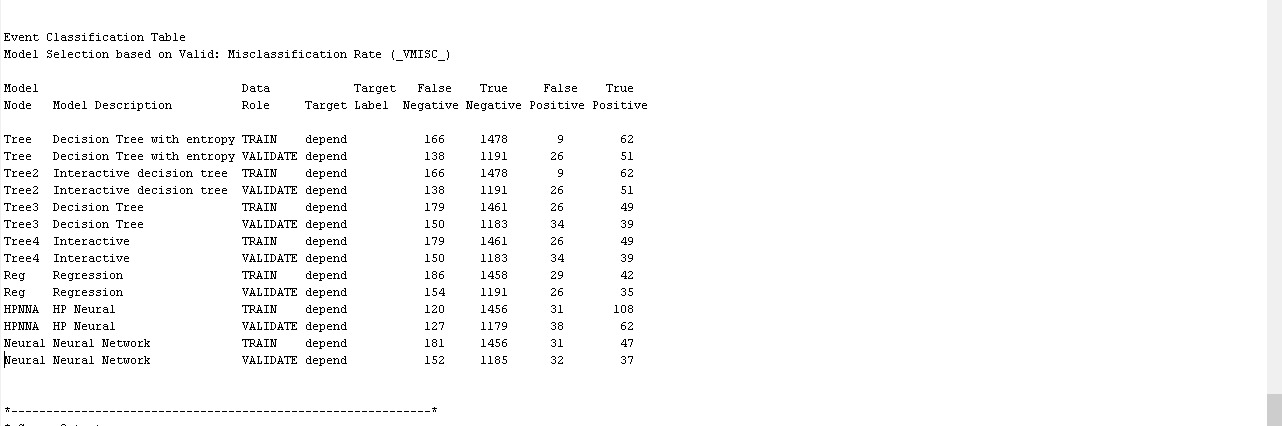
****

***Figure 7: Champion model before improvement, ROC and the overall Misclassification Rate***

## Confusion Matrix

We focus more on users who book a trip because that is how we can estimate the accuracy of booking. Based on Decision Tree, the confusion matrix is as follows:

|  |  |  |
| --- | --- | --- |
|  | Booking  (Predicted Yes) | No Booking  (Predicted No) |
| 1  (Actual Yes) | 39 (TP) | 150 (FN) |
| 0  (Actual NO) | 34 (FP) | 1183 (TN) |



***Figure 8: Confusion Matrix with Event Classification Table***

Cost 5 for misclassifying 1 as 0 => 150\*5 = 750

Cost 1 for misclassifying 0 as 1 => 1183\*1 = 1183

Correct Predictions = 39+1183 = 1222

Incorrect Predictions = 150+34 = 184

Total Scored Cases = 39+150+34+1183 = 1406

Error Rate = 184/1406 = 0.1309

Overall Accuracy Rate = 1222/1406 = 0.8691

# Part V. Model Performance Improvement

After we had a half-way check with professor and showed our champion model from Step4, we were convinced that the model can still improve. Also, there was a dependency between x12 (Dummy) and x41 (Depend – bookfut) which we had missed. So, we created a new variable (newVar) based on these 2 variables:

LIBNAME myproj 'C:\SASFiles';

**DATA** expedia;

SET 'C:\SASFiles\expedia.sas7bdat';

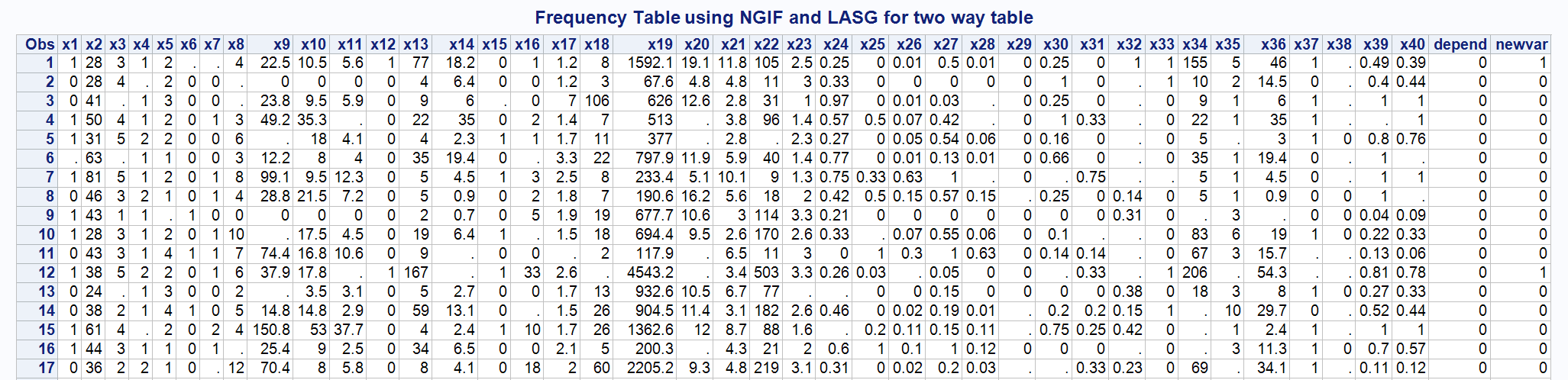
IF x12=**1** THEN newvar=**1**;

ELSE newvar= depend;

**RUN**;

**PROC** **PRINT** DATA=expedia;

**RUN**;



***Figure 9: New Table***

We changed these variables accordingly:

Binary: x1-gender, x6-child, x15-weekend, x37-SEgc, x33-bookgc, x41-depend

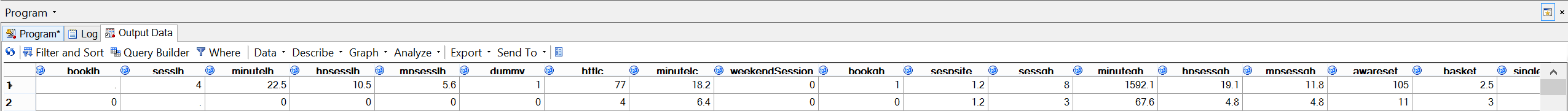
Ordinal: x3-income, x5-hhsize, x7-booklh

Rejected: x32-SERate, x38-path, x12-dummy

Target: bookprob

## Data Preprocessing

Changed variable names accordingly:



***Figure 10: New variable***

LIBNAME Expedia 'C:\ABA with sas';

**DATA** Expedia.expedia;

SET 'C:\ABA with sas\expedia';

rename x1=Gender x2=Age x3=Income x4=Education x5=HouseSize x6=Child x7=booklh x8=sesslh x9=minutelh x10=hpsesslh;

rename x11=mpsesslh x12=dummy x13=httlc x14=minutelc x15=weekendSession x16=bookgh x17=sespsite x18=sessgh x19=minutegh x20=hpsessgh;

rename x21=mpsessgh x22=awareset x23=basket x24=singleSiteSess x25=booksh x26=hitsh x27=sessh x28=minutesh x29=entrate x30=peakrate;

rename x31=exitrate x32=SErate x33=bookgc x34=hitgc x35=basketgc x36=minutegc x37=SEgc x38=path x39=hitshc x40=minutshc depend=bookfut newvar=bookprob;

**RUN**;

**PROC** **PRINT** DATA= Expedia.expedia;

**RUN**;

### Data Cleansing and Data Imputation

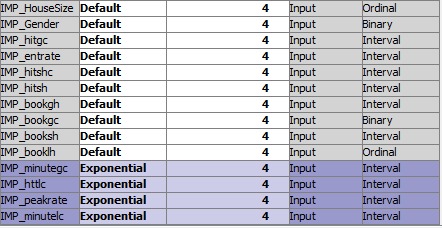
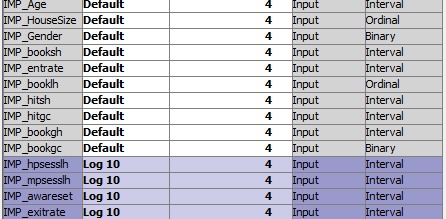
* We did Data Cleaning on noisy and inconsistent data
* We need to do Data Imputation to fill in missing data. Yes, we used imputed data for all classifiers except Decision Tree and Random forest for better results and model comparisons
* Since replacing all the missing values would become biased if there are many missing values, the Impute node comes into play when the missing values are randomly scattered throughout the input data and consist of ~10% of the total number of values for any one input variable

### Data Integration

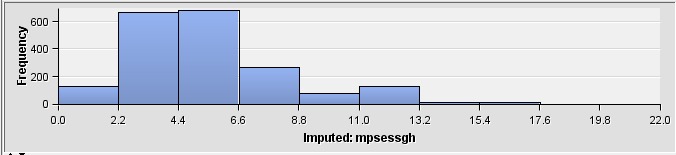
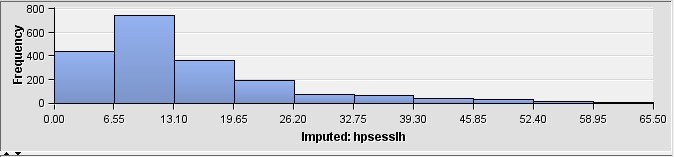
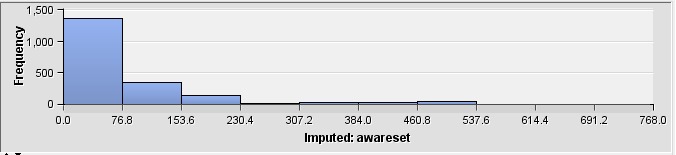
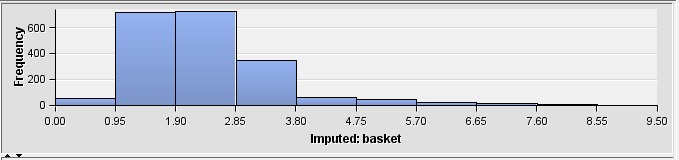
* Data integration from multiple sources: x12 (Dummy) had a high variable importance while x41 (depend-bookfut) was the target variable, which when compared had discrepancies in data which might have affected our final model analysis
* We addressed it by *combining x12 and x41 to form a new variable* as shown above in part V
* There were many missing values (> 50%) in x32 and x38, thus we rejected them

### Data Transformation

* Data transformation: we normalized data: used log and exponential to change the skewed data
* The skewness was reduced by using different base for log and exponential and also on different variables based on trial-and-error on various combinations

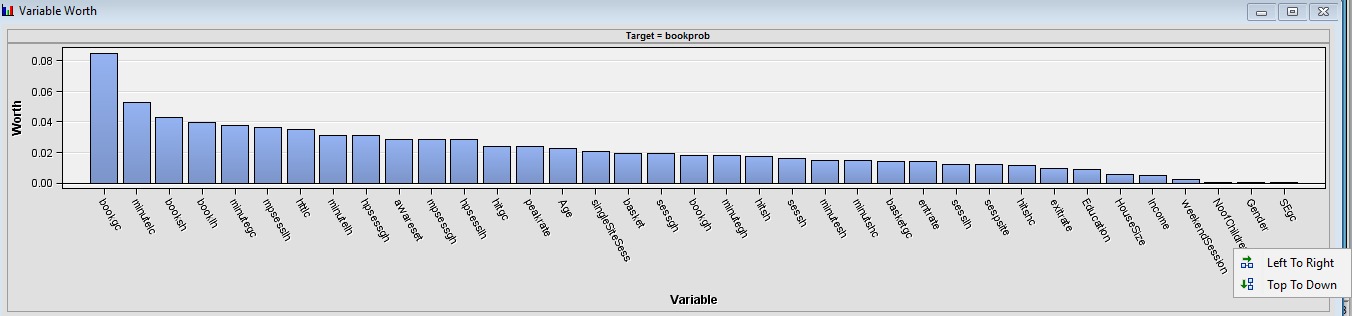
***Figure 11: log Figure 12: Exponential***



***Figure 13: Skewness measure***

### Data Reduction

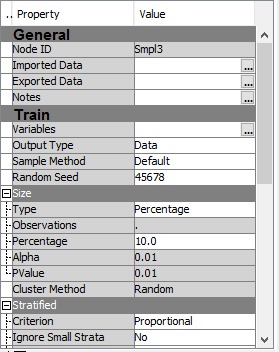
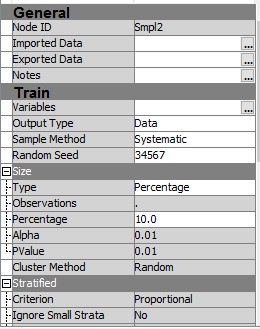
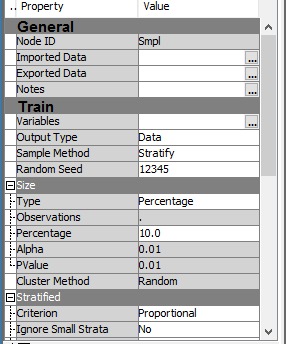
* Data reduction: We did remove variables as mentioned above (x32, x38)
* We dropped x12 and depend (which were highly correlated) as we created a new variable with their combination which became our new target
* These are the other variables which help us come to new conclusion on the number of bookings



***Figure 14: Variable worth***

Frankly, we felt the data set is not that large

* But since Sampling is an effective way determine the stability of an obtained sample value which represents the population from which it is drawn, we did try sampling
* We reduced data volume yet (mostly) and preserved patterns by taking 3 samples



***Figure 15: Sampling for Decision Trees (Bootstrap Aggregating)***

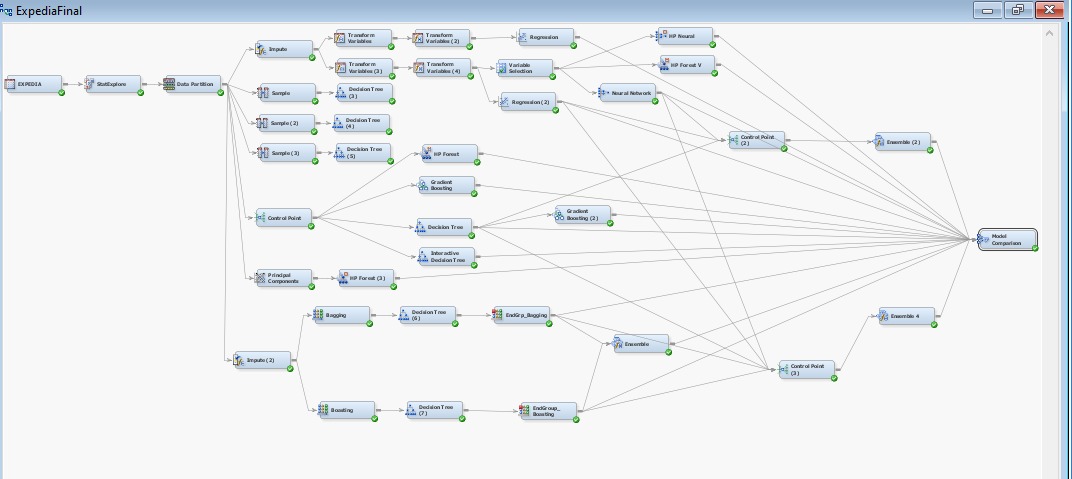
* Ensemble node can be used to average the predicted probabilities of all connected models. An advantage of this averaging ensemble method is that it smooths the linear cut points, making your model more robust in handling new data
* We can use it for boosting, bagging or model-averaging by using Decision Tree, Random Forests or Bagging-Boosting via Start and End groups to understand the variable importance

## Data Improvement Techniques

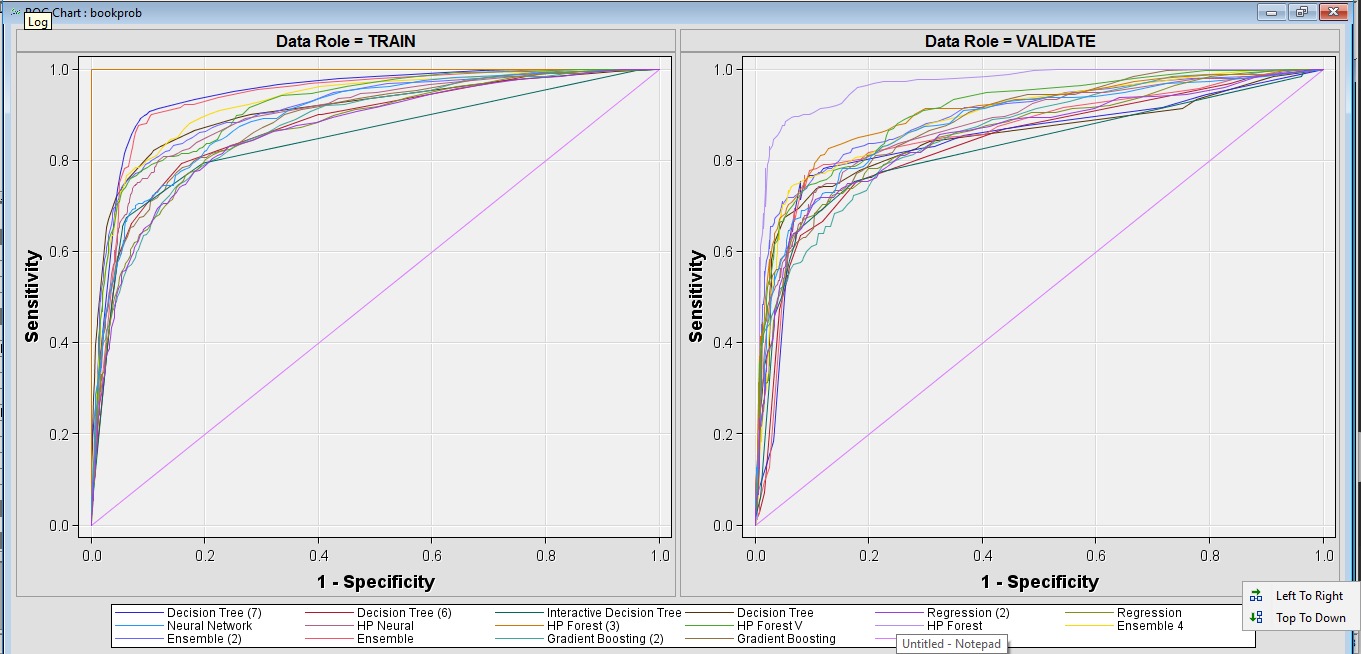
* We partitioned the dataset using 70% of the data for training and 30% for validation and saw performance improvement in many models
* Compared to our previous model, we implemented few new nodes here. We saw mixed results: few improved, few models did not perform at their best and few remained same

|  |  |
| --- | --- |
| **Bagging** Iterations: 20  Misclassification Rate:  Train: 0.142857  Validation: 0.15688 | **Boosting** Iterations: 20  Misclassification Rate:  Train: 0.09111  Validation: 0.12486 |
| **Ensemble (With Decision Tree, Neural and Regression)** Posterior Probability: Average  Misclassification Rate:  Train: 0.112637  Validation: 0.11526 | **Ensemble With 5 Nodes (Decision Tree, Neural Network, Regression, Bagging and Boosting)** Posterior Probability: Voting  Misclassification Rate:  Train: 0.113553  Validation: 0.11739 |
| **HP Forest with PCA** Misclassification Rate:  Train: 0.01  Validation: 0.13127 | **HP Forest with Variable Selection** Misclassification Rate:  Train: 0.118132  Validation: 0.124867 |
| **HP Forest**  To avoid overfitting, tree depth is chosen as 10 and Maximum trees to be 20  Misclassification Rate:  Train: 0.0  Validation: 0.07897 | **Gradient Boosting** Misclassification Rate:  Train: 0.146062  Validation: 0.144077 |

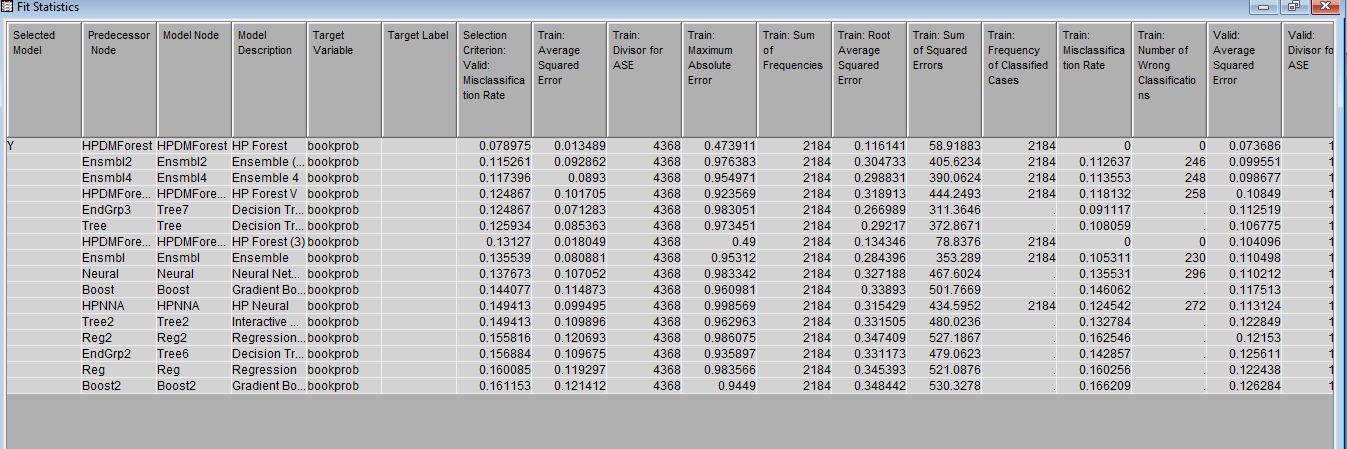
# Part VI. Summary



***Figure 16: Final Model***



***Figure 18: Final ROC Curve***



***Figure 17: Fit Statistics***

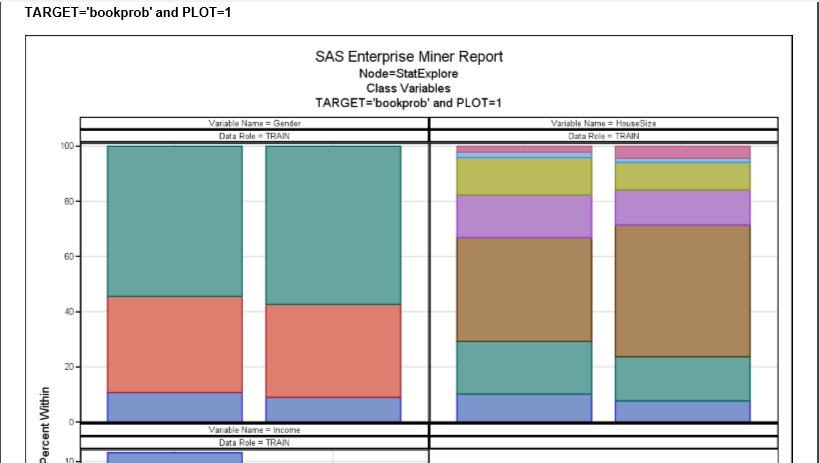
## Learnings

|  |  |  |
| --- | --- | --- |
| MODEL | DATA | Misclassification Rate |
| Bagging | Train | 0.125 |
| Boosting | Train | 0.157 |
| HPForest | **Train** | **0.079** |
| Gradient Boosting | Train | 0.144 |
| Ensemble | Train | 0.115 |

* Certain insights about variable importance can only be obtained after few model implementations.
* Data is not always what it looks like: Considering data analysis before preparing the model helps to explain variable dependencies over the target; Stat Explorer contributed to look for missing values, data skewness, outliers and noisy/inconsistent data. Few variables tend to be of importance, but if they are direct implications with no logic, then there is no use of it.
* It’s better to use Replacement technique to keep the limits of data in 3 sigma bound so as to get rid of outliers and noisy data
* Data imputation with tree surrogate method and transforming data to reduce skewness using Log transformations is a good idea to progress with before applying logistic regression
* HP Forest works good for this dataset. However, main key is to try various scenarios like: apply variable selection or principal component analysis prior to achieve better results
* Working with variable selection after imputing data following random forest model helps us to achieve less misclassification rate
* We can always manipulate number of trees with random forest to save the computation time and cost. Also, to prevent overfitting we can decide on some pruning parameters to present more reliable model

## Business Inference

* Men booked flight tickets more than women
* Income category 2 made the highest number of booking



***Figure 18: Score Card Inference***

# References

[*https://support.sas.com/resources/papers/proceedings14/SAS133-2014.pdf*](https://support.sas.com/resources/papers/proceedings14/SAS133-2014.pdf)

[*https://www.salford-systems.com/resources/webinars-tutorials/tips-and-tricks/using-surrogates-to-improve-datasets-with-missing-values*](https://www.salford-systems.com/resources/webinars-tutorials/tips-and-tricks/using-surrogates-to-improve-datasets-with-missing-values)

# Appendix

Data from “An Empirical Analysis of the Value of Complete Information for eCRM Models”, Padmanabhan, B., Z. Zheng, and S. Kimbrough. MIS Quarterly, 30(2), 2006. Variables 1-15 are site-centric variables; 16-40 are additional user-centric variables and the last is the dependent variable. At the end of some variables, “g” means all sites (global) and “l” means only this site (local); “c” means only the current session and “h” means all past sessions.

|  |  |  |
| --- | --- | --- |
| **No.** | **Variable** | **Description** |
| 1 | gender | “1”—Male, “0” – Female |
| 2 | age | Age of the user |
| 3 | income | Income of the user |
| 4 | edu | “0” – high school or less, “1”-- college, “2” – post college |
| 5 | hhsize | Size of house hold |
| 6 | child | “1” – have, “0” – not have |
| 7 | booklh | No. of bookings the user made at this site in the past |
| 8 | sesslh | No. of sessions to this site so far |
| 9 | minutelh | Time spent in this site so far in minutes |
| 10 | hpsesslh | Average hits per session to this site |
| 11 | mpsesslh | Average time spent per sessions to this site |
| 12 | booklc | Dummy variable, indicating if the user has booked at this site up to this point in the current session |
| 13 | httlc | No. of hits to this site up to this point in this session |
| 14 | minutelc | Time spent up to this point in this session |
| 15 | weekend | Indicating if this session occurs on weekend |
| 16 | bookgh | No. of past bookings of all sites so far |
| 17 | sespsite | Average sessions per site so far |
| 18 | sessgh | Total no. of sessions visited of all sites so far |
| 19 | minutegh | Total minutes of all sites |
| 20 | hpsessgh | Average hits per session |
| 21 | mpsessgh | Average minute per session |
| 22 | awareset | Total no. of unique shopping sites visited |
| 23 | basket | Average no. of shopping sites visited per session |
| 24 | single | Percentage of single-site sessions |
| 25 | booksh | Percentage of total bookings are to this site |
| 26 | hitsh | Percentage of total hits are to this site |
| 27 | sessh | Percentage of total sessions are to this site |
| 28 | minutesh | Percentage of total minutes are to this site |
| 29 | entrate | No. of sessions start with this site/total sessions of this site |
| 30 | peakrate | No. of sessions the user spend the most time within this site/total sessions of this site |
| 31 | exitrate | No. of sessions end with this site/total sessions of this site |
| 32 | SErate | No. of sessions coming from search engines/total sessions of this site |
| 33 | *bookgc* | Binary variable, indicating if this user has booked at any sites up to this point in the current session |
| 34 | *hitgc* | Total hits of all sites in the current session |
| 35 | *basketgc* | No. of shopping sites in this session |
| 36 | *minutegc* | Time spent of all sites in this session |
| 37 | *SEgc* | Indicating if this session uses search engines |
| 38 | *path* | Indicating if this site is an entry/peak |
| 39 | *hitshc* | Hits to this site/ hits to all sites in this session |
| 40 | *minutshc* | Minutes to this site/total minutes in this session |
| 41 | bookfut | Binary dependent variable, indicating if this user is going to book in the remainder of the session (after the clipping point) |