### OPIM 5604- Group Project

### Team 3 – White Paper

Animesh Choudhary

Arjun Periyasamy

Elisa Polci

Subhashree Srinivasan

### Spam Mail Identification

**Executive Summary**

Team 3 conducted a discovery project to build a predictive model that would identify an email as spam or not. Our team set out to build an analytics model that would enable email filtering to reduce the varied costs incurred by businesses due to spam. The data analysis, model development, and assessment activities resulted in both a high performing predictive model and discovery of opportunities to enhance the model for current day application. This white paper is intended to provide an overview of the analytics processes, tools and techniques applied. Additionally, it summarizes the key findings, business implications and recommendations for future works.

**Analytics Process, Tools and Techniques**

The analytics activities followed the SEMMA methodology- sample, explore, modify, model, assess. Information on the source data set is outlined along with tools and techniques utilized to support data pre-processing activities. In turn, iterative model development was conducted in both JMP Pro and R. Through model evaluation and comparison, a final model was selected and further assessed.

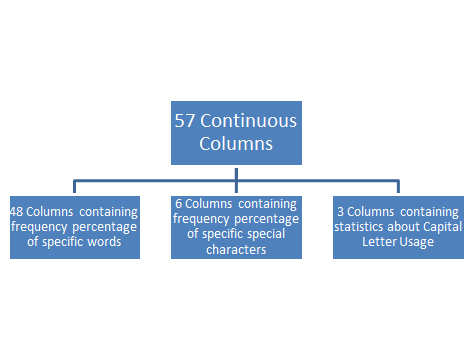
### Sample

The source sample data was obtained from the University of California Irvine’s Machine Learning Repository. The source data set was created by Hewlett Packard in 1999.

Source Data Set (Lichman,M)

* + Spambase (1999)
  + Sample of 4601 observations of spam/ non-spam classifications
  + 57 continuous variables, 1 nominal class variable

*\*Appendix A - link to data set documentation provided by authors*



As the source data set was compiled using emails from 1999 the team explored opportunities to enhance the analytics using more recent email data. The team decided to create the like data set using sample spam and non-spam emails obtained from an open-source filtering tool.

Assessment Data Set:

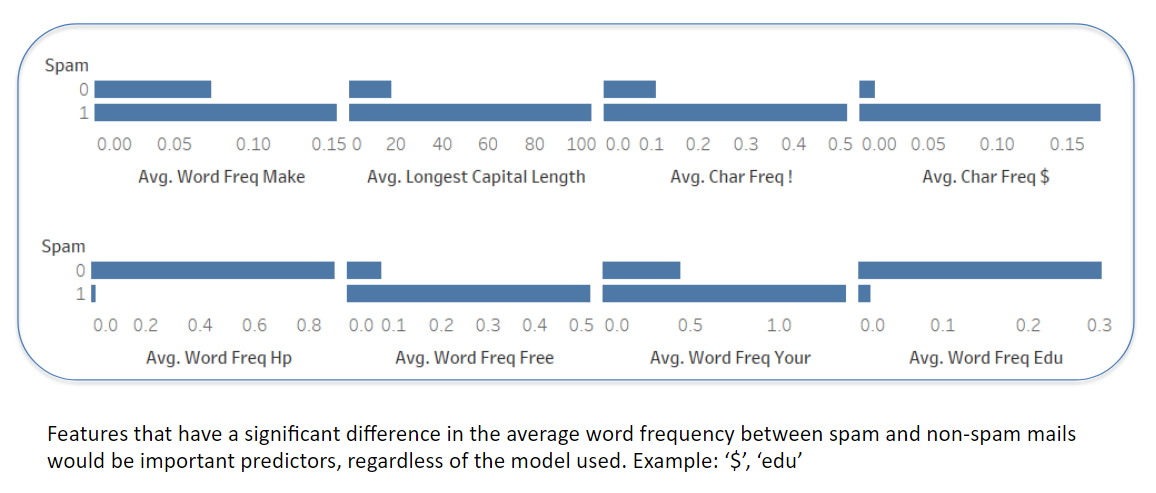
* + Obtained more recent sample of spam and non-spam emails to assess relevancy of original predictors on current emails
  + Spamassassin (2005)

This data set was used to assess performance of the selected model on more recent email.

### Explore

Exploration of the data was accomplished using features in JMP Pro and multiple data visualization techniques in both JMP Pro and Tableau. Boxplots and scatterplots proved to be useful tools to gain an understanding of the continuous data relationships and irregularities.

**Visual Data Exploration:**

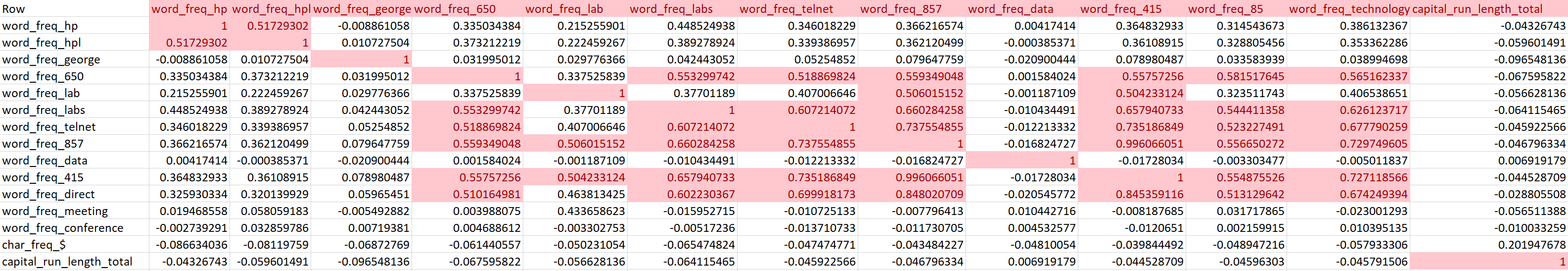


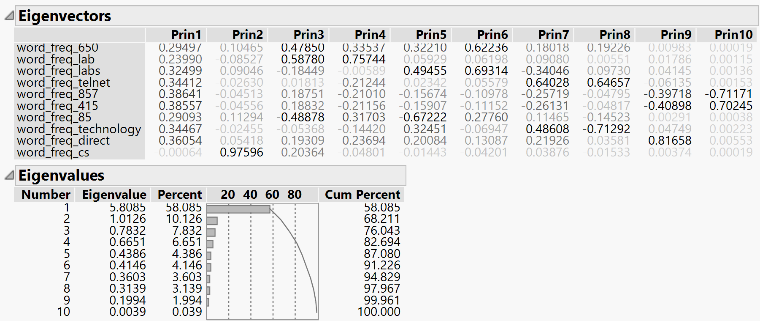
### Modify

Further data analysis was conducted to identify opportunities to reduce data dimension including Principal Component Analysis in JMP Pro. It was ultimately decided that modeling would be conducted on the full set of variables

**Principal Component Analysis:**

In-order to reduce the number of variables in hand, we decided to perform Principal Component Analysis on all the variables. However, out of all the 57 variables, we cannot conclude saying all th 57 would have a linear relationship amidst them. Since PCA only works for variables with a linear relationship amidst them, we had to first identify these 57 variables and then apply PCA on them. The following screenshots indicate this process:



From the 57 variables, we only identified 10 variables to have a linear relationship between them. Out of the three, on performing PCA, we had the freedom to eliminate only 4 variables in-order to capture 90% of the variance. Hence PCA, proved of not much help for us in our model creation.

**MODEL**

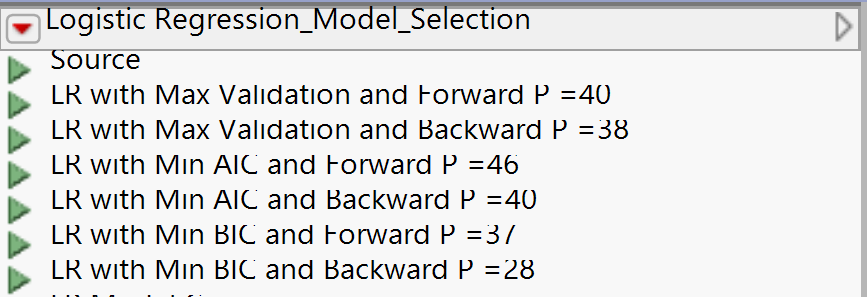
**JMP Modelling:**

Utilizing the ease of model creation in SAS JMP pro, we decided to create multiple classification models of different types. Finally, the best model with the lowest misclassification rate, highest ROC and the lowest Asymmetric cost was to be selected. Asymmetric cost is a selection parameter which was introduced by us to help us evaluate the models better. The requirement for a new selection parameter arose to consider the importance of False Positives and False Negatives in a classification model. The importance of False Negatives and False Positives vary from one model application to another. In our case of spam mail classification, False Positives are of more concern than False Negatives. False Positives arise when our model classifies a non-spam mail as a spam mail. In such cases, there is a possibility for the user to miss out some very important mails, which would have been incorrectly classified as spam by our model. On the other hand, the effect of False Negatives, when a spam mail is classified as a non-spam by our model, is considerably less. In order to capture this difference in level of importance, we had to introduce a new parameter for model evaluation called, Asymmetric Cost. For, every Misclassified observation weights were assigned based on whether they are False Positives or False Negatives. Later the sum of all the Asymmetric weights were taken and used as a evaluation parameter. In our case, we assigned a weight of 1 for False Positive observations and 0.1 for False Negative observations. The model with the lowest Asymmetric cost eventually has the least number of false Positives present in them.

A considerable amount of time and effort was put into the process of model creation process, using various tuning parameters different types of models created were, which are:

* Boosted Trees
* Bootstrap Forest Trees
* Decision Trees
* Logistic Regression model
* Neural Networks

Considering the fact that we would be creating Decision trees, Bootstrap Forest Trees and Boosted Trees, we had to include a 3 split dataset consisting of Training, Validation and Test. Since we need an unseen set of data to test the pruned trees, we created a 60-20-20 split of the data in hand as Training, Validation and Test respectively. Below is a detailed description of the various types of models created.

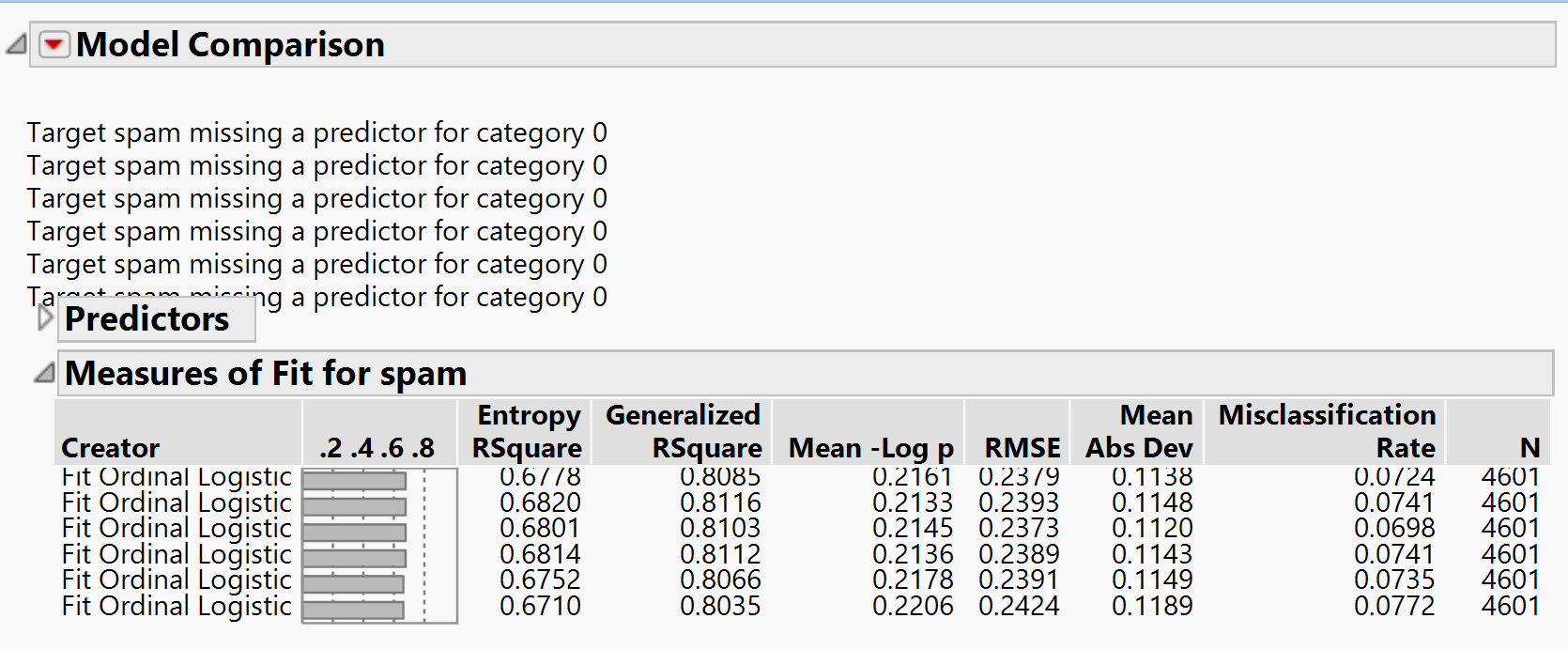
1. **Logistic Regression**

We started our model creation stage by creating a Logistic Regression model. We experimented the performance of Logistic regression model by varying the stopping rule and directions. The following screenshots Indicate the different Logistic Regression models we have created:

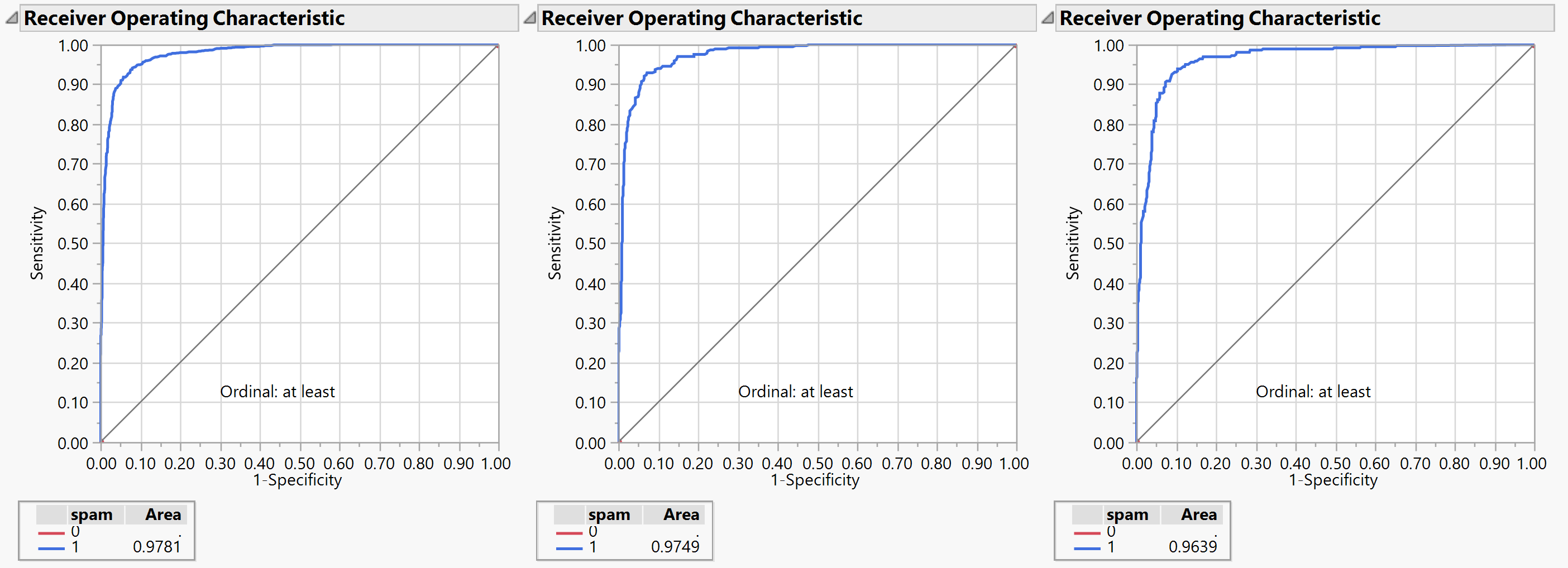
After various combinations of stopping rules and directions, the final 6 best performing logistic regression models are:

* Logistic Regression with Maximum Validation stopping rule in the Forward Direction with 40 variables.
* Logistic Regression with Maximum Validation stopping rule in the Backward Direction with 38 variables.
* Logistic Regression with Minimum AIC stopping rule in the Forward Direction with 46 variables.
* Logistic Regression with Minimum AIC stopping rule in the Backward Direction with 40 variables.
* Logistic Regression with Minimum BIC stopping rule in the Forward Direction with 37 variables.
* Logistic Regression with Minimum BIC stopping rule in the Backward Direction with 28 variables.

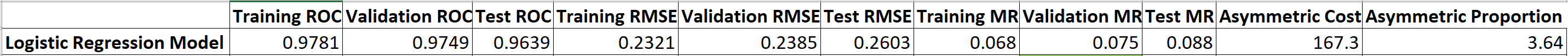
From the above mentioned 6 Logistic regression models, a comparison of their performance was done using the model comparison function in JMP. Below is a screenshot showing the model comparison window.

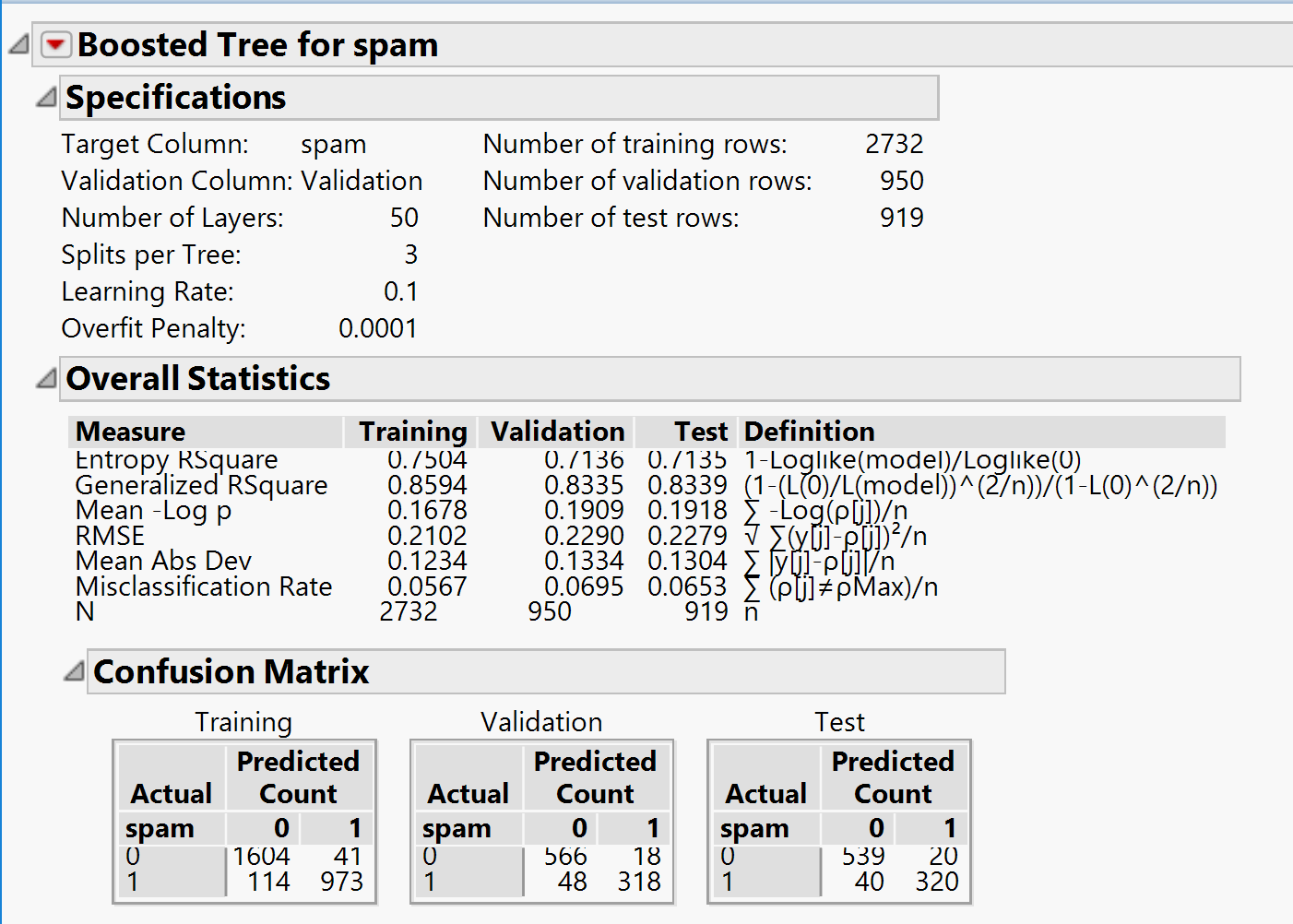
As we can see, the model with Maximum Validation stopping rule in the Backward Direction having 38 variables has the best performance. The number of variables considered in the model is also a considerably small amount of 38 considering the nature of the problem in hand and the total number of variables in the dataset.

Below are the ROC curve of the final logistic model selected:



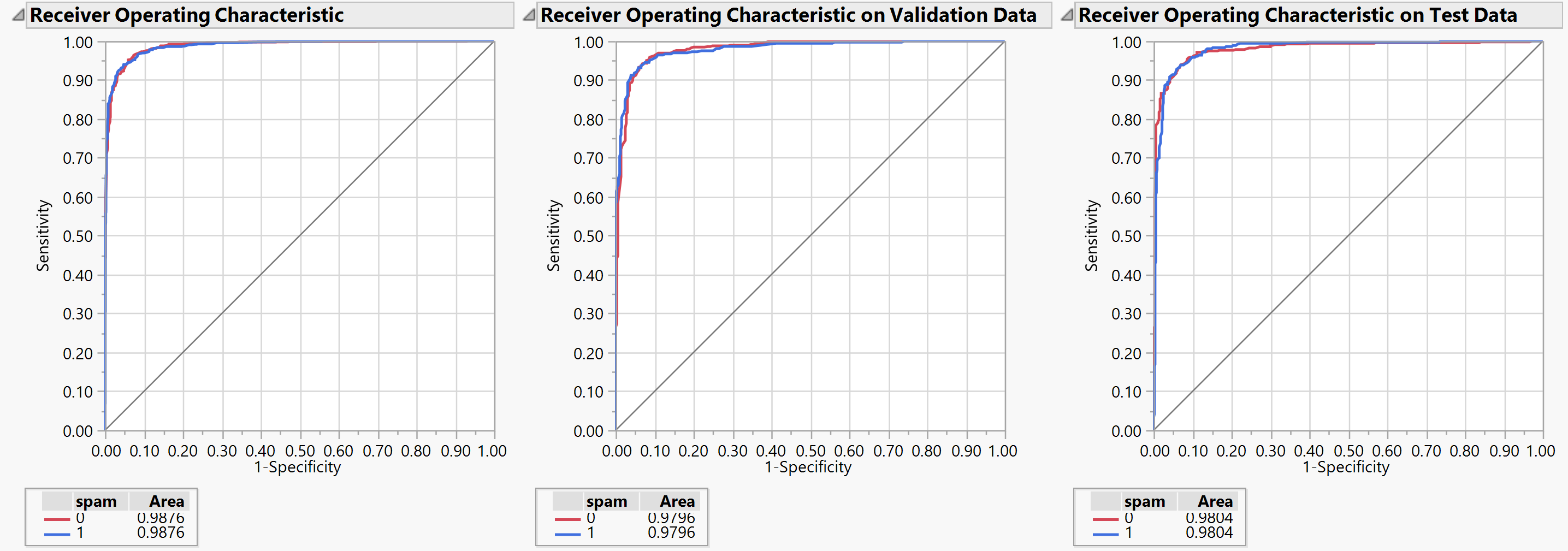
The final statistics of the Logistic Regression model were as follows:



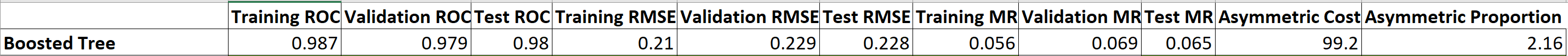
1. **Boosted Tree**

Following the Logistic regression model, we created a boosted tree model to classify the mails as spam or non-spam. Since boosted trees generate trees in a sequential manner, one following the other, based on the residuals of the preceding tree, we were hoping to see an improvement in performance when compared with the Logistic Regression model. We tried creating multiple boosted trees by varying the tuning parameters such as learning rate and number of splits per tree. We noticed that as the learning rate increased the performance of the model reduced. The best performing model had a learning rate of 0.1 and number of splits per tree as 3.

The specifications statistics and ROC of the boosted tree are highlighted in the screenshots above and below:



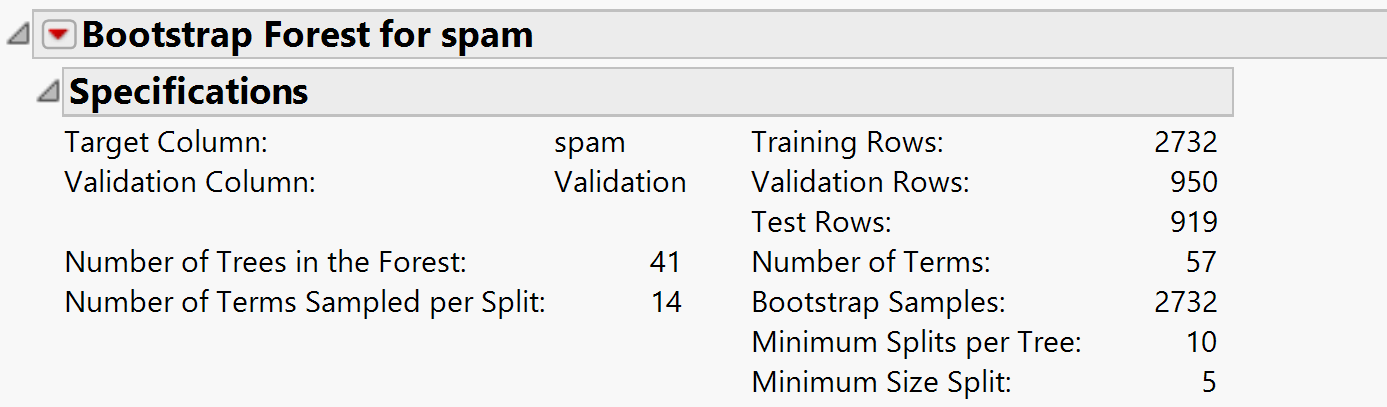
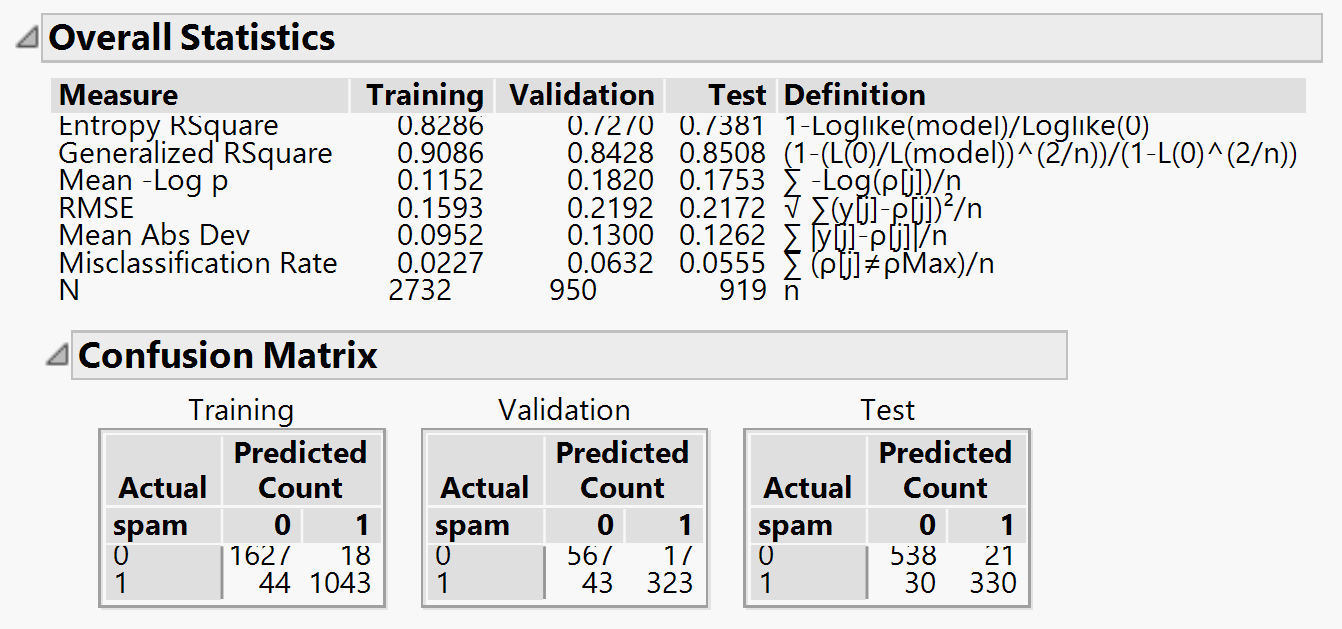
The final statistics of the boosted tree are as follows:



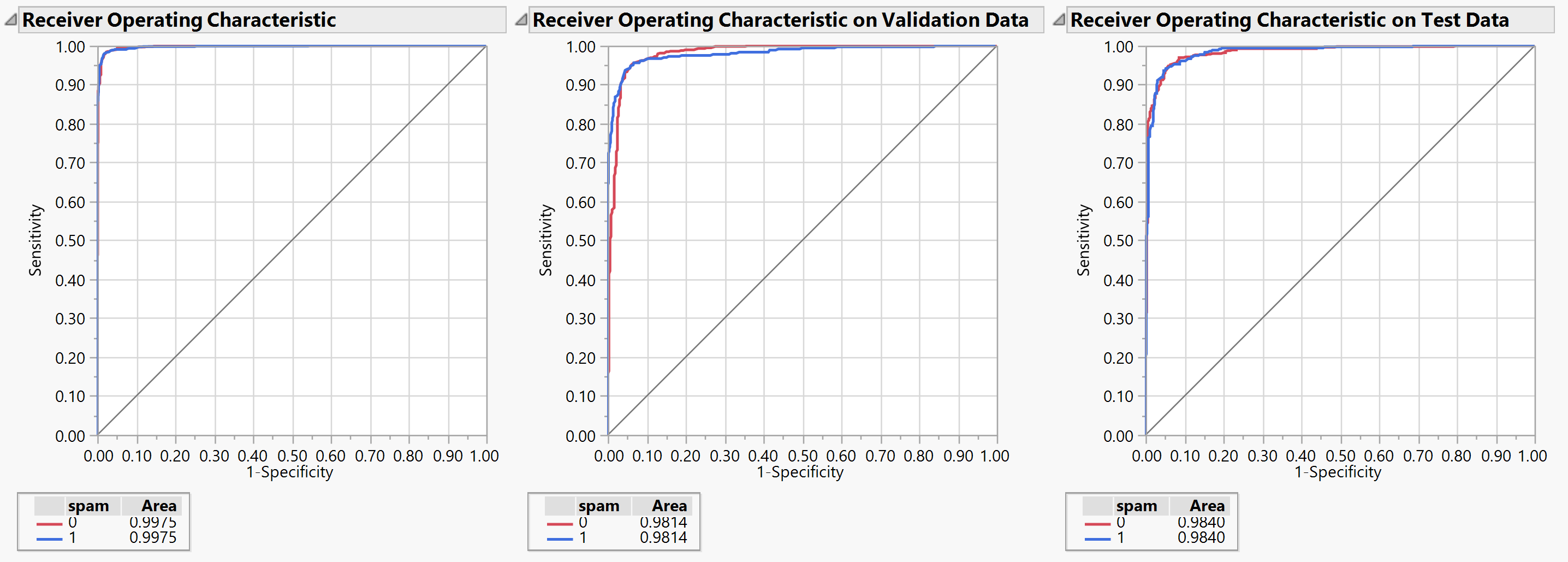
Comparing the two models in hand, we can see that the boosted tree was performing better than the Logistic regression model as we had presumed. The ROC in Training, Test and Validation datasets are higher whereas the RMSE and misclassification rates are lower than the Logistic Regression Model.

1. **Bootstrap Forest**

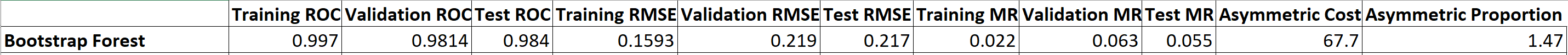
To confirm the best performing model, after the boosted tree model, we created a model using the Bootstrap Forest technique. The bootstrap forest technique creates multiple trees parallelly, using the bootstrapping technique, and the result is a combination of the outputs of all the trees generated. In case of continuous variables, the predicted value is the aggregate of the output from all the trees. In our case, a classification model, the data is of categorical datatype. Hence the output is the classification which occurs the most number of times in all the trees produced by bootstrapping. The specifications and statistics of the best performing Random Forest model are as follows:



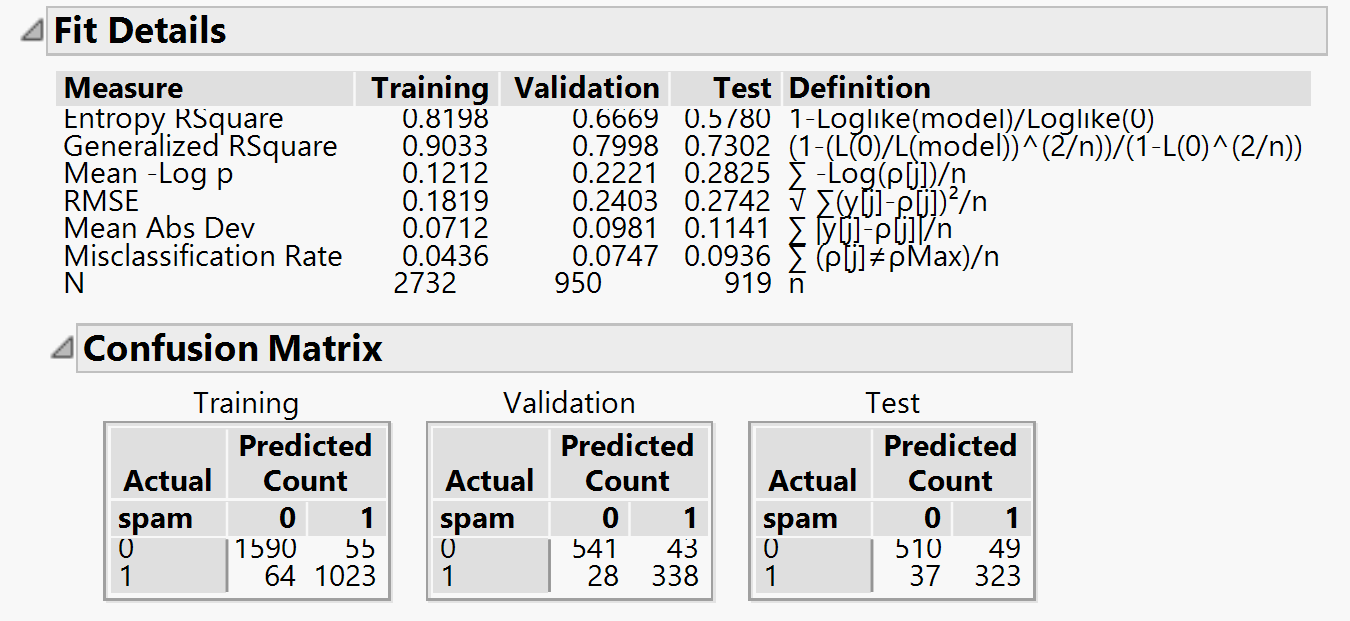
The ROC curves for the bootstrap forest model are as follows:



The final statistics of the bootstrap forest model are as follows:

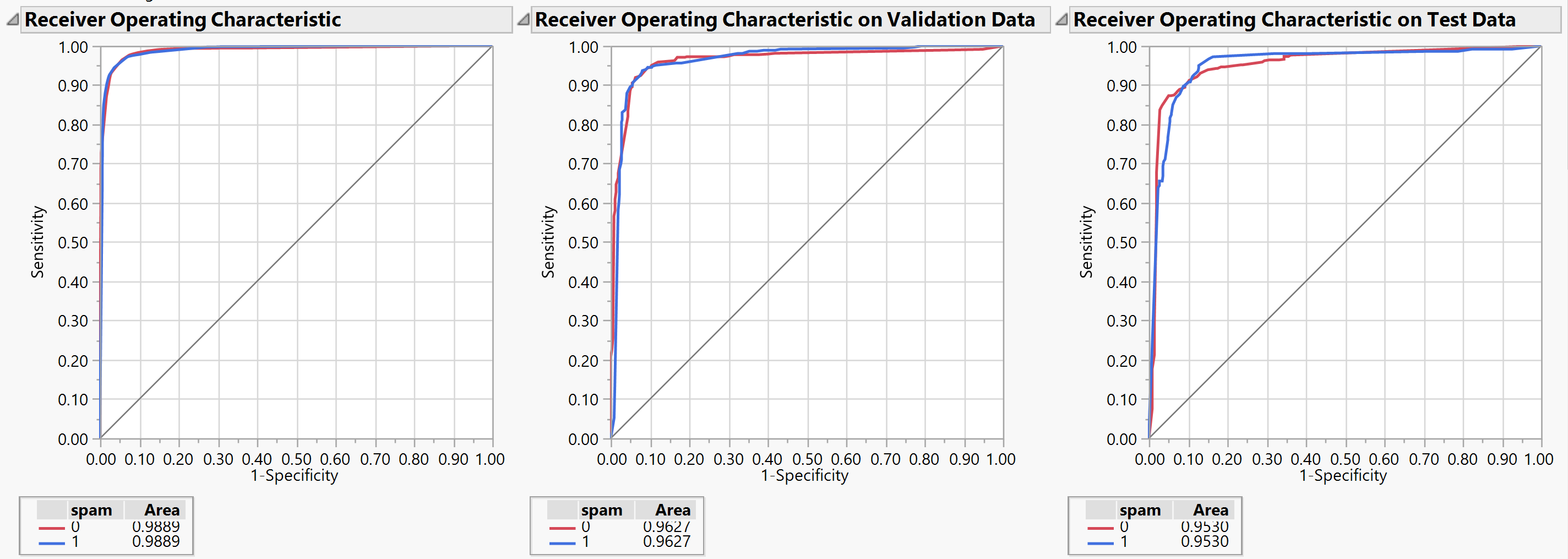
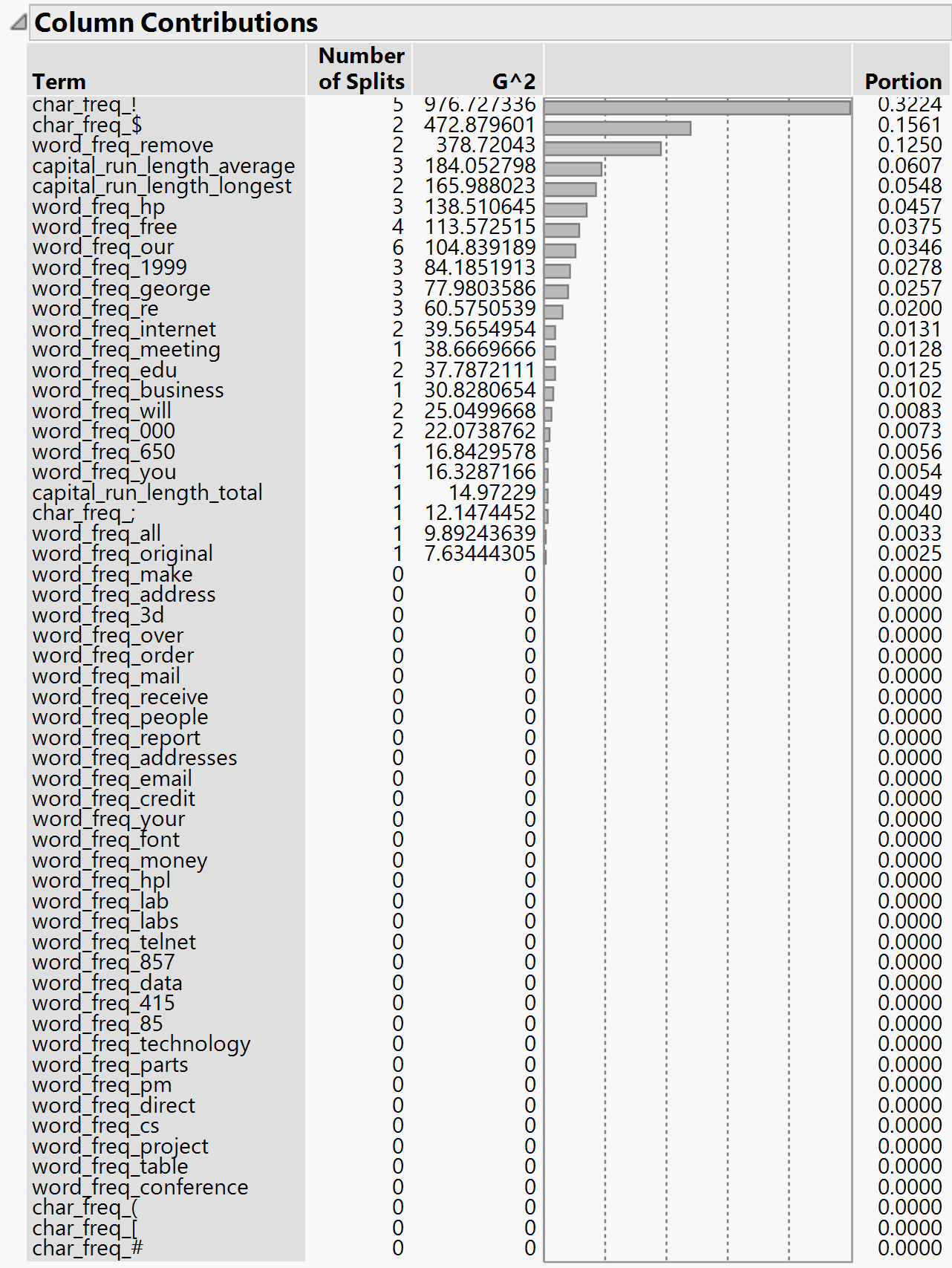


Comparing with the 3 models in hand, we can see that the Random Forest model has the best performance statistics, the highest ROC in all three datasets and the Lowest RMSE and Misclassification rates amidst the three models.

1. **Decision Trees:**

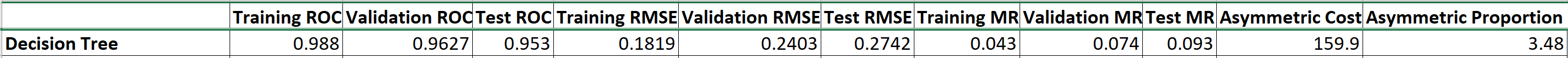
We also created a decision tree to classify the observations. We used the 60-20-20 Training, validation and Test dataset split to grow, prune and test the trees. As we all know, in decision trees the tree is allowed to grow to its full extent in the training dataset and then the tree is pruned using the Validation dataset. The statistics of the decision tree generated are shown in the adjacent screenshot:

The ROC curves of the decision tree in the three datasets area as follows:



The decision tree is pruned in JMP in a way to reduce the G^2 value, which is basically an alternative to the Gini index and Entropy Value which measures the dissimilarity between splits. Using column contributions option in JMP, we can see which columns when split produce the greatest dissimilarity in splits.

The final performance of the decision tree is as follows:



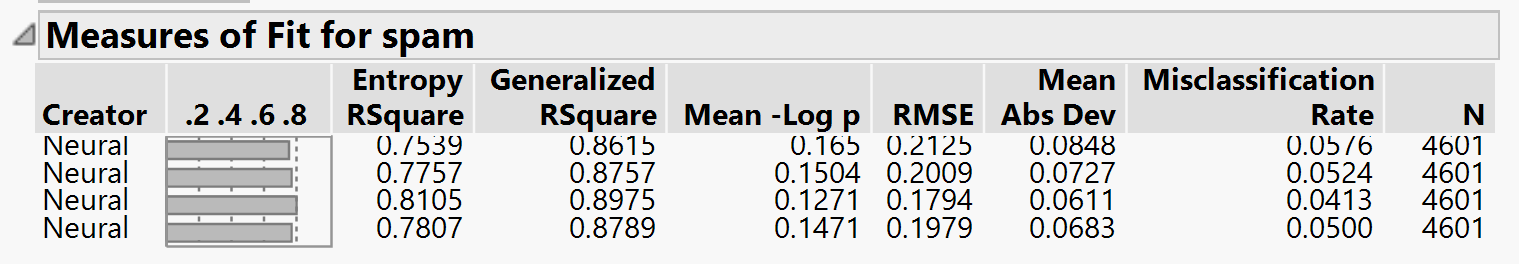
We can notice that the ROC values are lower than the Bootstrap Forest model. Further, the Misclassification rate and RMSE are higher than the Bootstrap Forest model for the decision tree. Hence, we can say that the performance of the Bootstrap Forest model is better than the decision tree.

1. **Neural networks with all variables**

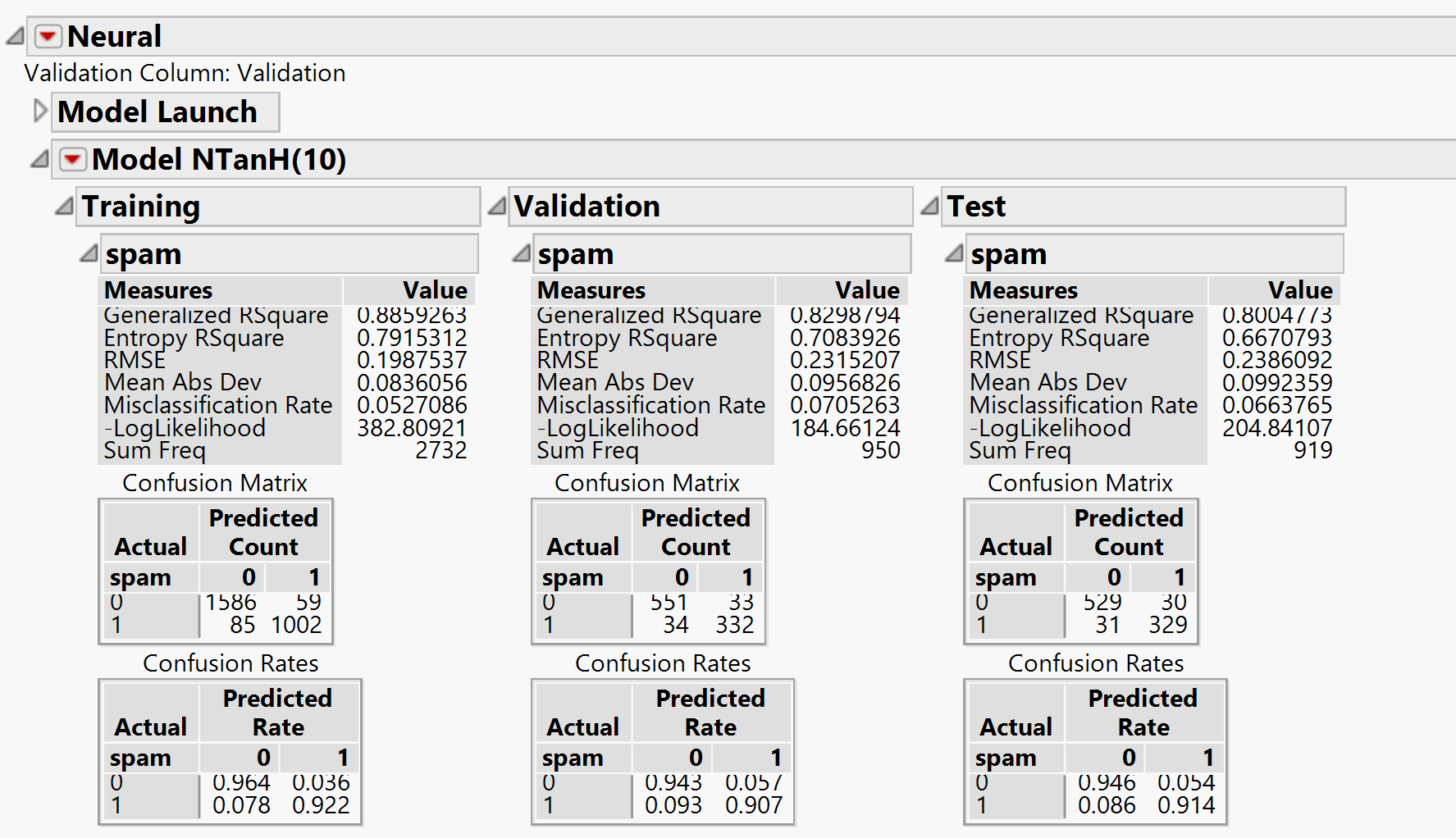
Finally, we decided to create a neural network model to see if there is any improvement in the performance compared to the already well performing Bootstrap Forest model. In neural networks we created multiple neural networks using different Activation functions, different number of nodes and different number of layers. The final neural network was chosen by comparing the performance of all these individual neural networks. Initially, we considered all the variables while creating the neural network, no variable selection was done. After studying the performance of different activation functions, we identified the models to be performing well with the TanH activation function. The following were the best variants of neural networks created:

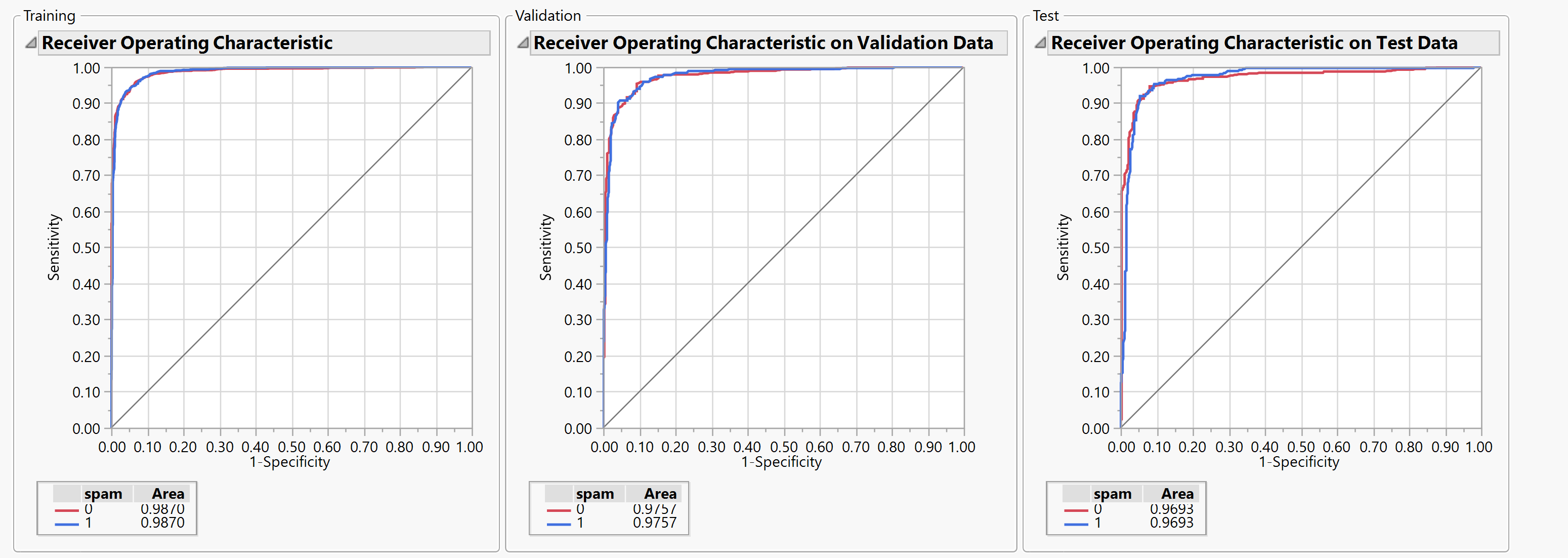
* Neural network with TanH Function, 1 Layer and 3 nodes.
* Neural network with TanH Function, 2 Layer and 3 nodes.
* Neural network with TanH Function, 1 Layer and 10 nodes.
* Neural network with TanH Function, 2 Layer and 10 nodes.

The performance of the different variants were compared using the model comparison function on JMP. The Following screenshot shows this:

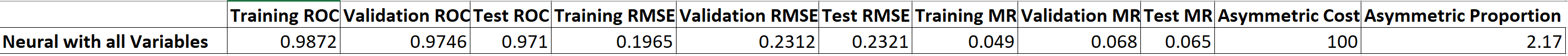
We noticed that, increasing the number of hidden layers produced a fall in performance due to the model overfitting the training data. Similarly increasing the number of nodes beyond 10 also produced a fall in the performance. The best neural network was the one with TanH Function, 1 Layer and 10 nodes.

The performance statistics of the neural network in the 3 datasets and their ROC values are as follows:

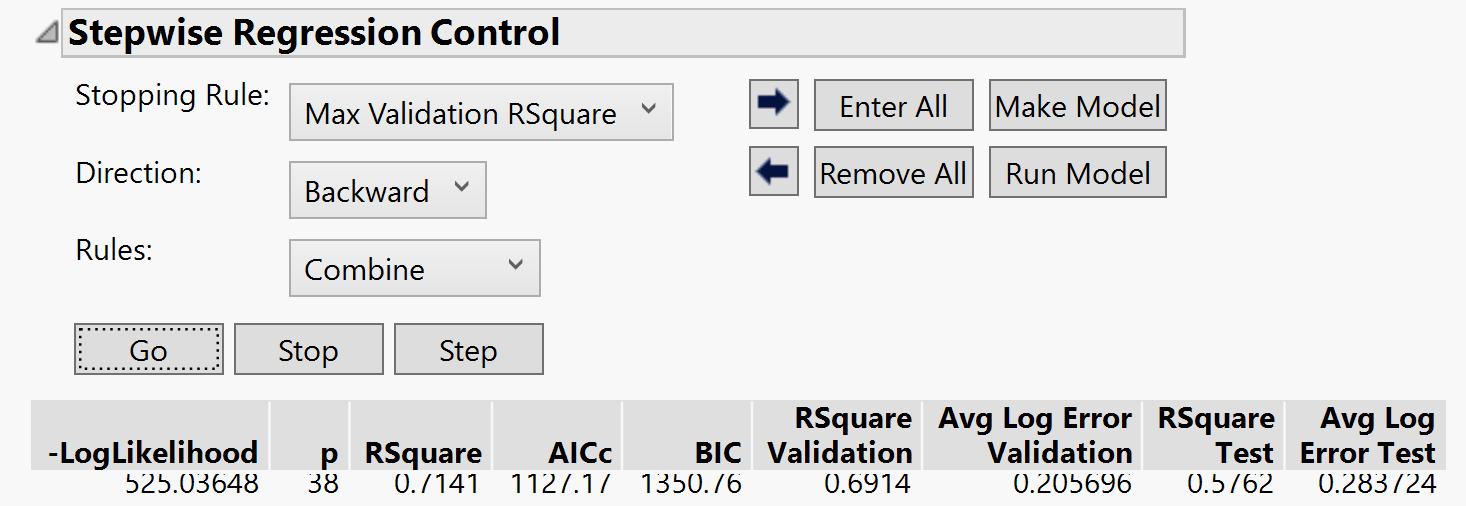




The final performance of the neural network are as follows:

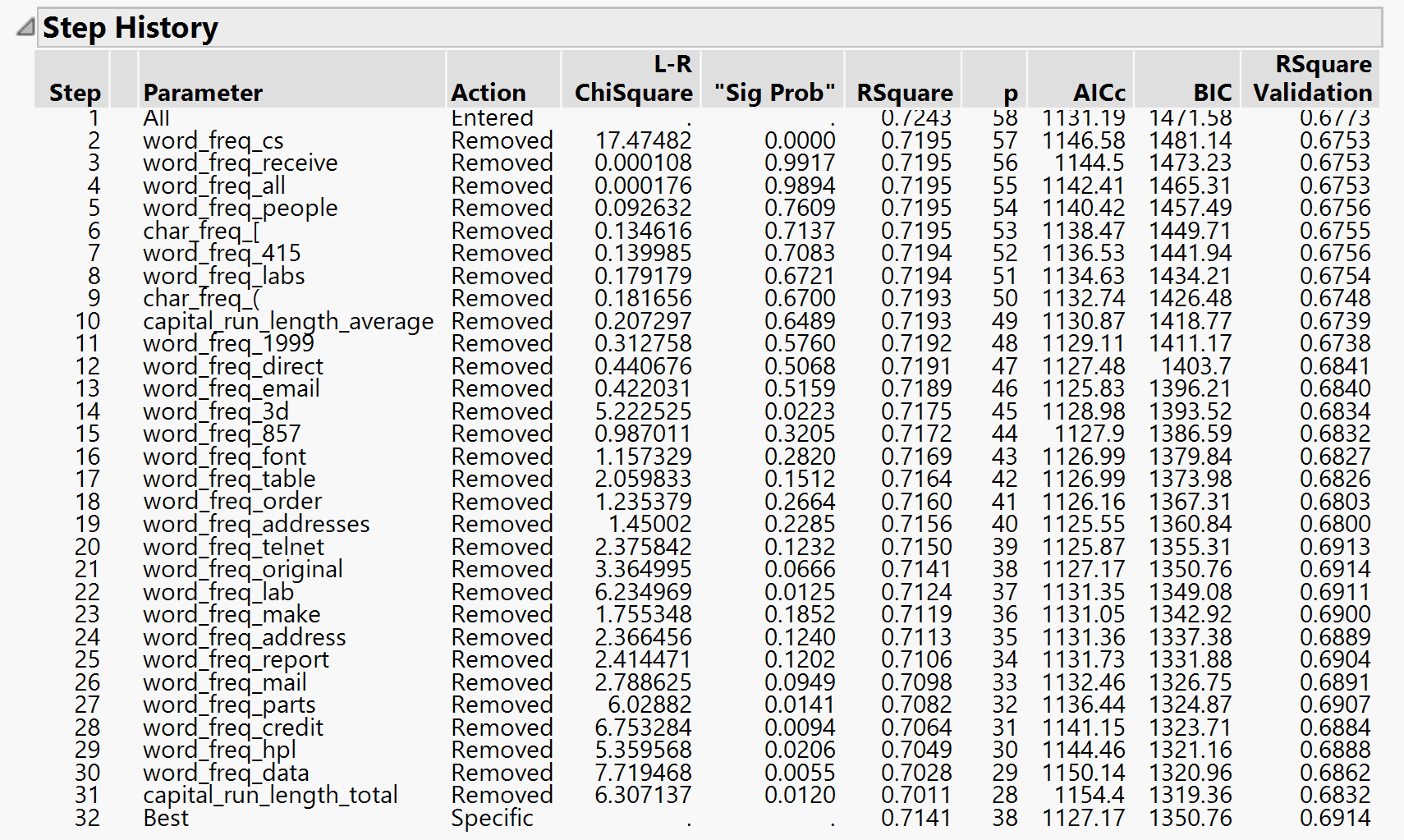
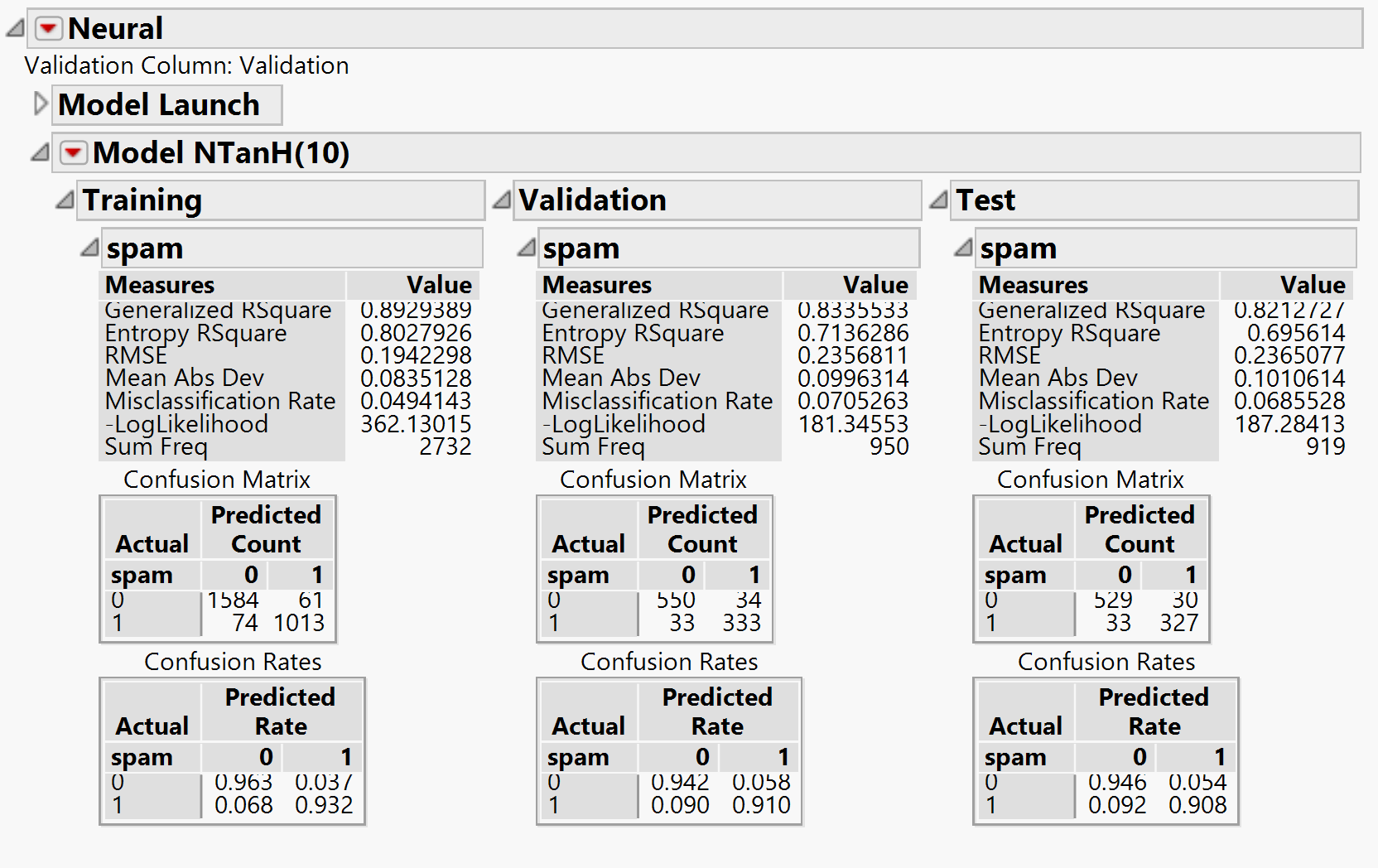


We can notice that the ROC values are lower than the Bootstrap Forest model. Further, the Misclassification rate and RMSE are higher than the Bootstrap Forest model for the neural network without any variable selection. Hence, we can say that the performance of the Bootstrap Forest model is better than the Neural Network without any variable selection.

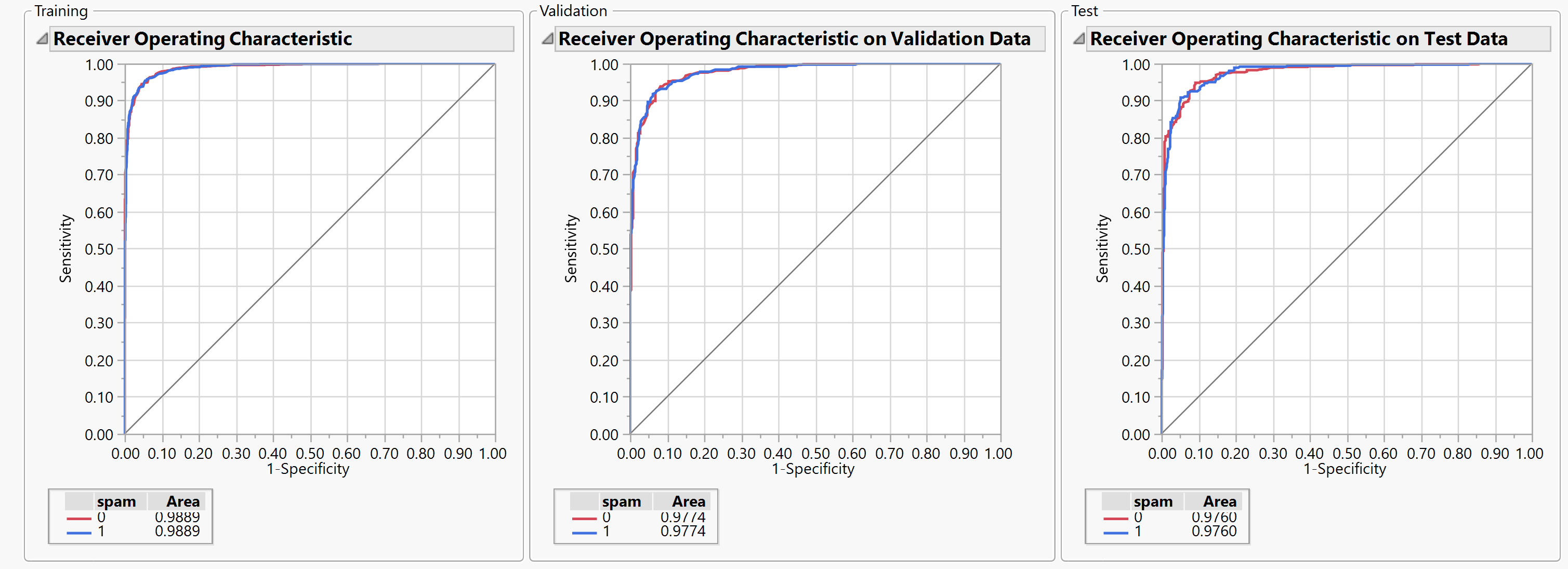
1. **Neural networks with variable selection:**

We wanted to study if reducing the number of variables had an effect in the performance of the neural network. Since Logistic Regression, decision trees, boosted trees and Bootstrap forest models perform variable selection. To select the most contributing variables, we chose the variables used in the best performing Logistic Regression model.

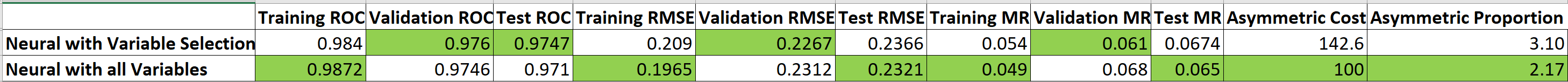
The screenshot of the removed variables, only 38 have been chosen form the original 57, and the statistics of our best performing neural network model are shown below:



The ROC curves of the model created are as follows:

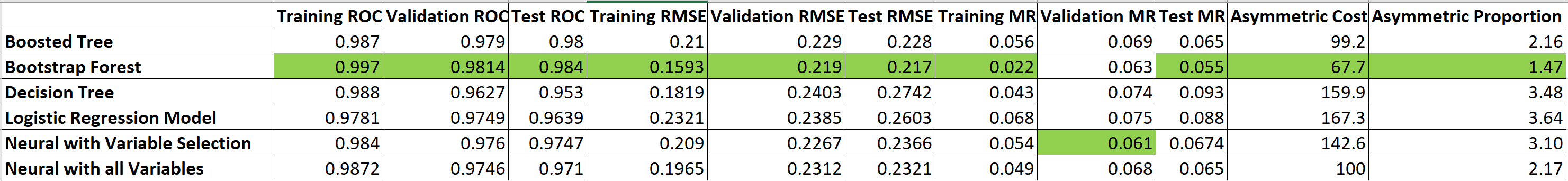


Comparing the final performance of the neural networks, with and without variable selection are as follows:

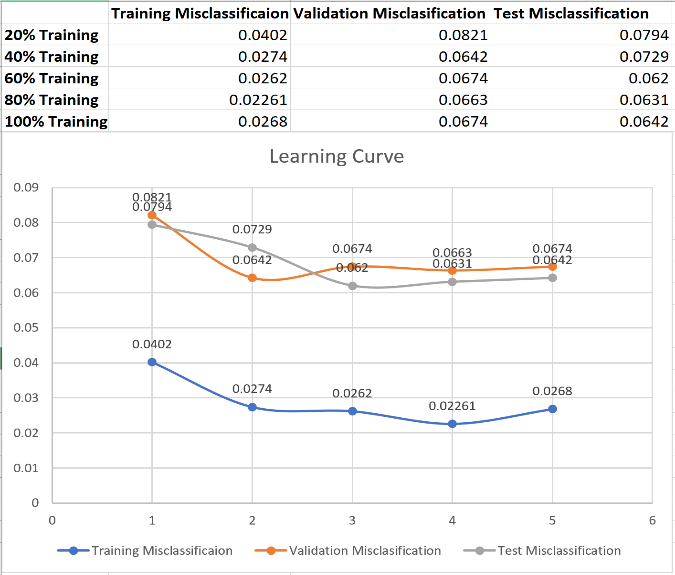


We can see that, the performance is almost on par, Variable selection did not have a drastic effect on the performance of the neural network.

**Comparison of all models Created on JMP:**



The above table shows a detailed comparison of all the models created in JMP. The green cells highlight the best performance values for the parameter. We can clearly see that the Bootstrap Forest model has the best performance statistics of all the models generated.

**Learning Curve of Bootstrap Forest model:**

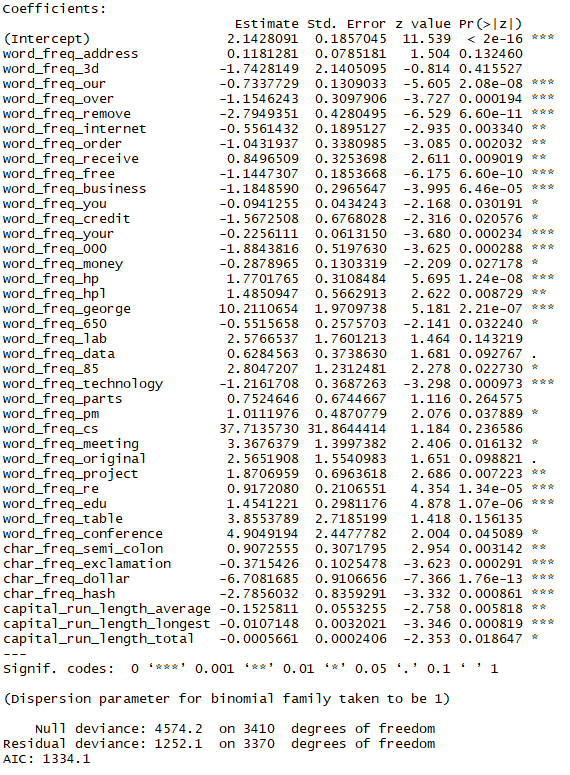
In-order to evaluate the effect of increasing the number of observations on the performance of our final Bootstrap Forest model, we created learning curves. By keeping the model and all other parameters constant, we plotted the misclassification rate against the training dataset size. We can infer from the learning cure that as the training dataset size increases the misclassification rate falls for all the three datasets. There is a huge negative slope between 20% and 40% of Training data, however on further increase of Training dataset portion the slope graduate reduces.

After finalizing on the best performing JMP model, we wanted to reconfirm the performance of all our models. We took to R to reconfirm the performance of our model. Below, we have mentioned the process we took to create models in R.

**Modelling in R:**

\* The Ratios considered for Asymmetric Cost are same across models to enable model comparison. But they are modified in the final model performance evaluation after comparison the better reflect the impact of False positives and False Negatives.

1. **Logistic Regression with variable selection based on minimum AICC value.**



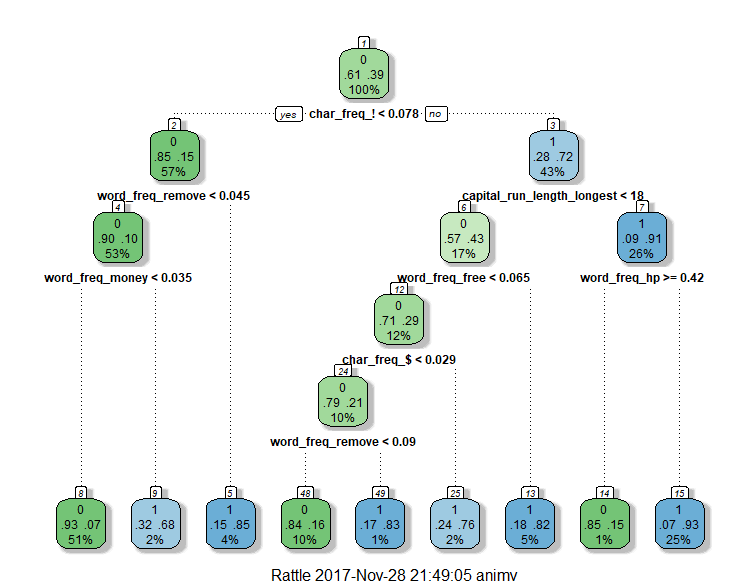
The model was obtained using backward selection. The forward and the mixed stepwise also resulted in the same model with this stopping rule. Columns used in this case and their coefficients can be seen in the image on the left. This model makes use of 40 of the 57 available columns, thereby eliminating only 17 columns.

|  |  |
| --- | --- |
| **Performance Metric** | **Value** |
| Validation ROC | 0.9510358 |
| Training ROC | 0.9544 |
| Validation AMR | 3.42% |
| Training AMR | 3.71% |
| Overall AMR | 3.61% |

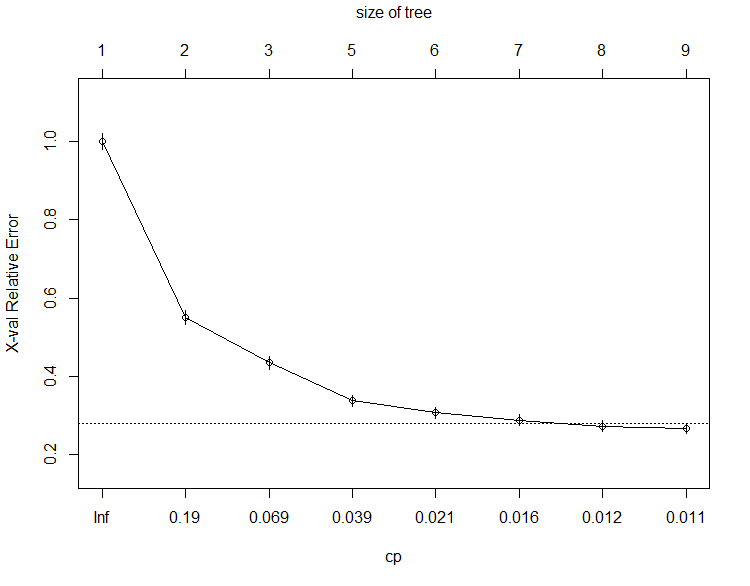
The performance of this model is as follows:

The performance of this model was good but it wasn’t selected because other models resulted in ROC’s which were closer to 1 than this model and lower asymmetric misclassification rates were achieved in them. Since, one of the objectives is to avoid false positives while classifying the emails into spam and non-spam categories, a reduction in the asymmetric misclassification rate is necessary and moreover the current values reflect only the performance based on the specific training validation split (75:25) with which the model was trained.

1. **CART based Classification Decision Tree**

This Classification Decision Tree was created using the rpart library in R. This tree makes use of only 7 of the total 57 columns present in the training data set. The performance of this tree based model is as follows:

|  |  |
| --- | --- |
| **Performance Metric** | **Value** |
| Validation ROC | 0.951 |
| Training ROC | 0.9166 |
| Validation AMR | 5.16% |
| Training AMR | 4.85% |
| Overall AMR | 4.93% |

This tree was created using a 75-25 split between validation and training set. In the rpart library, the tree is not pruned using the validation set and instead it is done using a cost-parameter. The cost-parameter is computed by using cross validation on the training set and the one which reduces the AMR the most is selected. The graph on the left shows the error values for various values of cost-parameter. As per the graph, the cost parameter value was selected to be 0.01 in our model because the decrease in error is low after the value of 0.01 as the cost-parameter value decreases. The size of the tree is also controlled by the value of cost-parameter because lower the value of it, the higher would be the size/total terminal nodes in the tree. The graph also makes it clear that by decreasing the cost-parameter and thus increasing the size of the tree, not much reduction in error would be seen.

Finally, the validation set was ultimately never used for training or pruning the tree but instead it was used only for gauging the performance of the tree on the unseen data.

1. **Boosted Tree using Xgboost**

A variation of Boosted tree was created in R with the xgboost package using a 75-25 training validation split. The boosted tree was created by gradient boosting on the misclassification error and at the same time, just like random forest, for each successive tree, 50% of the total columns were considered for the splits. This variation increased the performance of the model as compared to a normal boosted tree.

After testing multiple values for the various parameters, the following parameters were used because they gave the highest ROC and lowest AMR values. Also, the final performance metrics of the model can be seen below.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Description** |
| eta | 0.1 | Learning Rate |
| max\_depth | 15 | Maximum depth of a tree at any level |
| nround | 25 | Maximum number of iterations |
| colsample\_bytree | 0.5 | Portion of columns to be considered in each iteration |
| subsample | 0.5 | Portion of Training data to be considered to prevent overfitting |

|  |  |
| --- | --- |
| **Performance Metric** | **Value** |
| Validation ROC | 0.993 |
| Training ROC | 0.983 |
| Validation AMR | 2.08% |
| Training AMR | 1.15% |
| Overall AMR | 1.44 % |
| 10 Cross Validation AMR | 4.10% |

|  |  |  |
| --- | --- | --- |
| Actual | Predicted 0 | Predicted 1 |
| 0 | 2753 | 35 |
| 1 | 172 | 1641 |

Confusion Matrix (Whole data) at cutoff value of 0.7

This variant of Boosted Tree gave a very good ROC without overfitting the data but it was not able to bring down the False Positives as effectively as the Random Forest model which will be seen next without increasing the normal misclassification rate by a huge margin. The performance metrics of this model and the random forest model is close but since false positives is something which we are trying to avoid and this model even with a cut-off value of 0.7 still have more false positives than the random forest model, we selected the random forest model instead of this variant of boosted tree.

**4. Random Forest**

We created a Random Forest model in R using a training validation split of 75 – 25. The model was created using the following parameters:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Description** |
| ntree | 100 | Total number of trees in the forest |
| nodesize | 1 | Minimum size of terminal nodes |
| replace | TRUE | To allow Bootstrapping |
| mtry | Sqrt (57) | Portion of columns to be considered in each iteration |
| sampsize | Same as Training data | Size of each sample on which the tree is generated. |
| Cut-off value | 0.6 | Probability above which mail is spam |

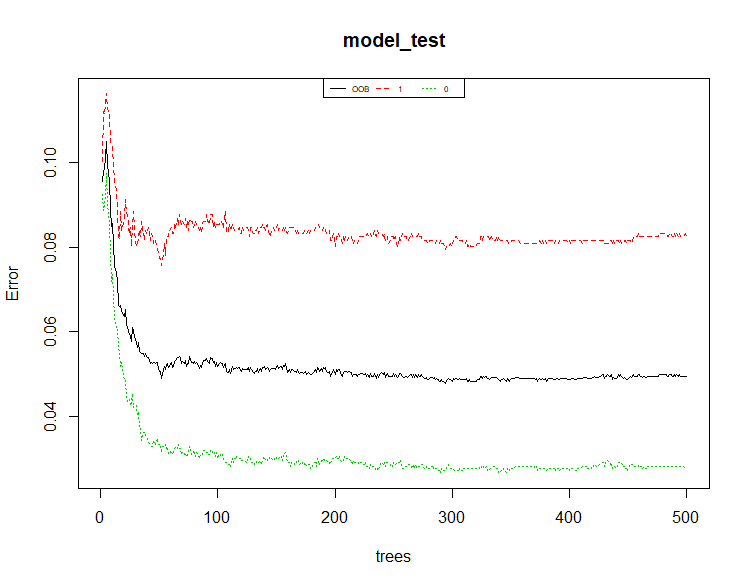
|  |  |
| --- | --- |
| **Performance Metric** | **Value** |
| Validation ROC | 0.993 |
| Training ROC | 0.983 |
| Validation AMR | 2.08% |
| Training AMR | 1.15% |
| Overall AMR | 1.44 % |
| 10 Cross Validation AMR | 4.10% |

|  |  |  |
| --- | --- | --- |
| Actual | Predicted 0 | Predicted 1 |
| 0 | 2773 | 15 |
| 1 | 77 | 1736 |

Confusion Matrix (Whole data) at cutoff value of 0.6

AMR = 1.48%

**Reasons for Parameter Selection:**

1. Number of Trees:

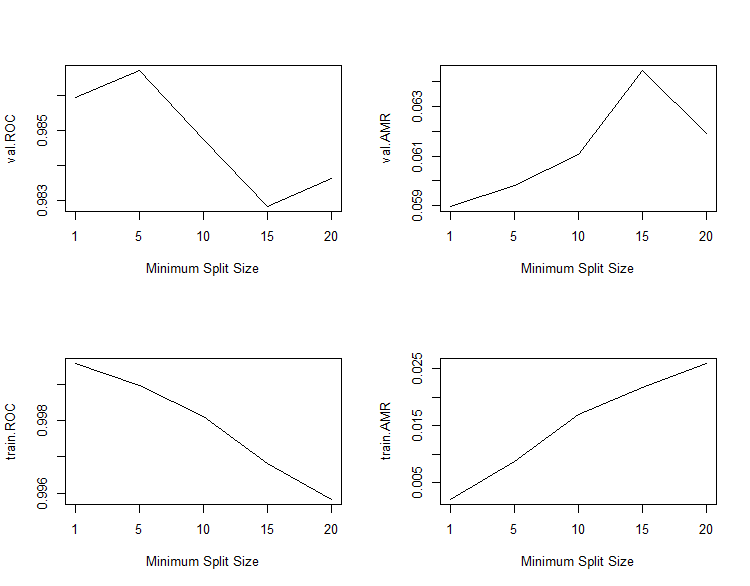
Based on the graph plotted here between the number of trees and the error associated with the, we chose the total number of trees in out random forest model to be 100. This is because on increasing the number of trees beyond 100, the reduction in error is not much which is quite clearly seen in the graph.

1. Number of columns to be considered in each iteration.

In case of classification, the number of columns to be considered in each iteration is normally the square root of the total number of columns present in the dataset.

1. Sample Size and replacement

In case of random forest, the samples used for generating various tress are obtained through bootstrapping which requires the replacement of the rows after being sampled and the sample size to be same as the original data size used for training.

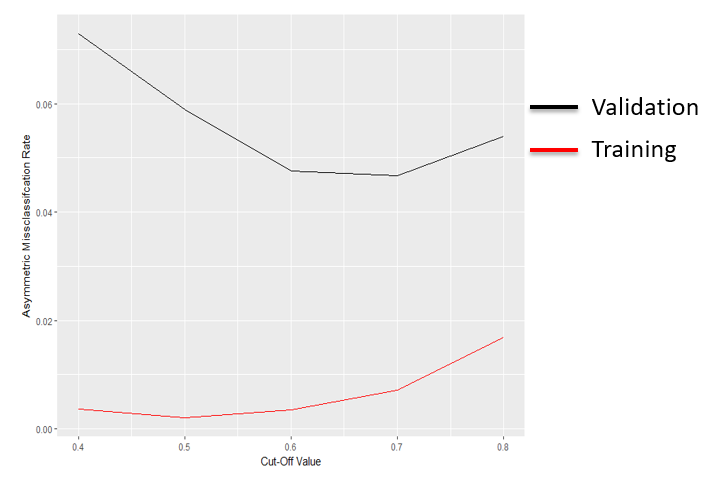
1. Minimum size of terminal nodes for individual trees

The graph on the side represents the effect of various minimum split sizes of trees on the training and validation set ROC’s and AMR’s.

It is seen that the ROC is maximum and the AMR is minimum when the minimum split size is 1.

The values at split size 1 may look like the data may be getting overfitted but since these values are good for both training as well as validation splits, any doubts that the model is overfitting the data can be put to rest.

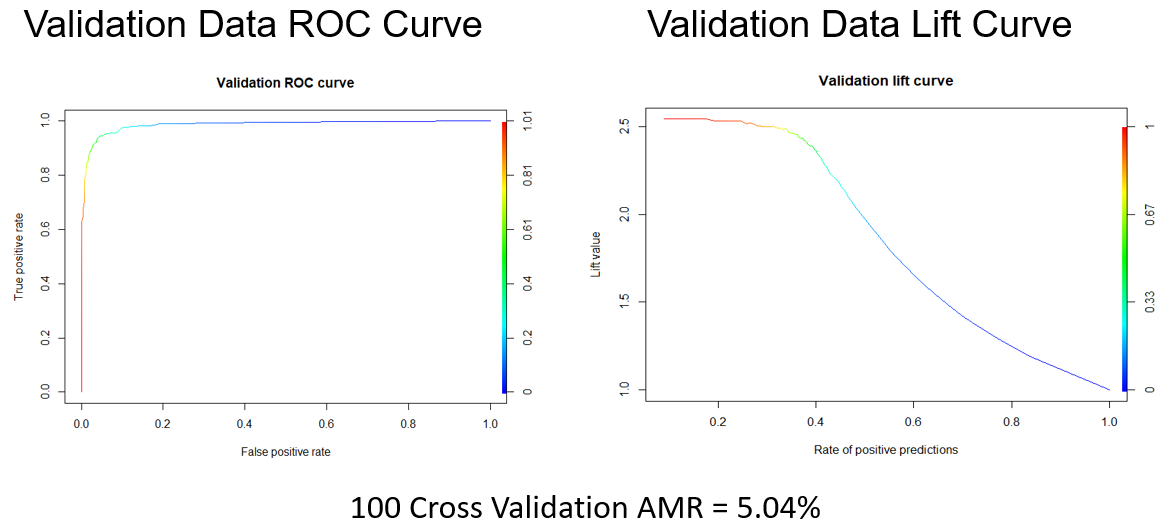
1. **Cut-off Probability**



Since, we want to decrease the number of false positives, it makes sense to increase the cut-off. The graph on the side indicates the asymmetric misclassification rate from increasing the cut-off value from 0.4 to 0.8. It is seen that at a cut-off of 0.6, the validation AMR is almost the lowest and beyond 0.6 and reduces very slowly. For 0.6 the training AMR has increase from what it was at 0.5 but the increase is much lower than the decrease in the case of validation set. Beyond 0.6 the increase in training AMR looks a bit more than decrease in the validation AMR and that is why the final cut-off is chosen to be 0.6 and not 0.5 or 0.7.

This information can also be realized by looking at the ROC curve which is present below, where it is seen that the false positive rate hardly reduces when the cut-off increase form 0.6 and it reduces sharply till 0.6.

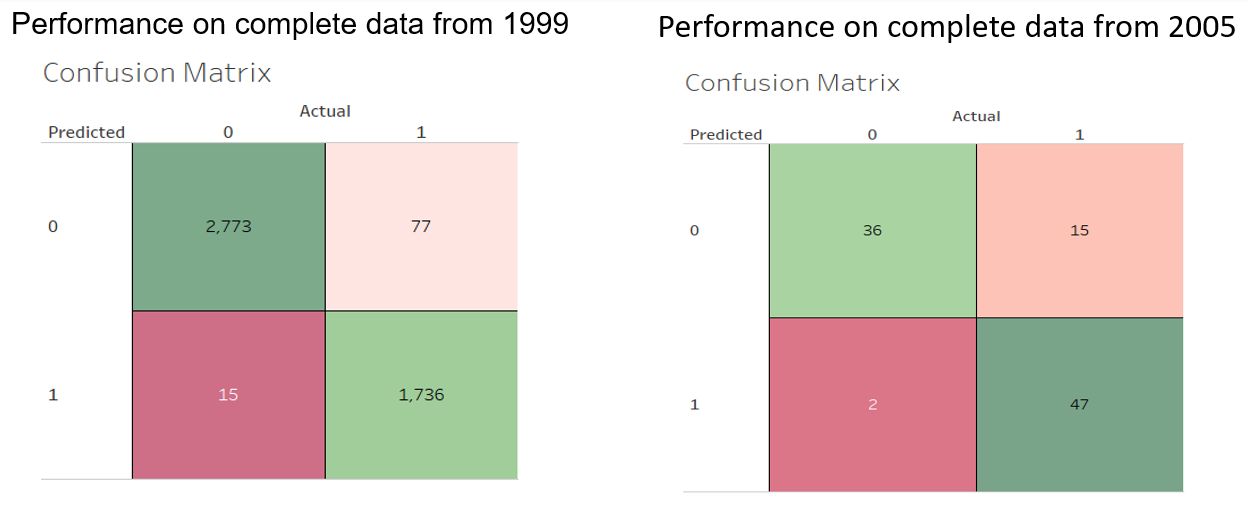
**Final Model Performance:**



The above images show the ROC and Lift curves of the validation data from our model. This high AUC value indicates that the model is not over-fitting the training data even though when the training data ROC is so close to 1. The ROC curve proves that the cut-off of 0.6 which was chosen s correct as we can see that beyond 0.6 (green part of the ROC curve), there is hardly any decrease on the false positives which we wanted to minimize. Also, beyond 0.6 the true positive rate starts decreasing and the false negative rate starts increasing.

The Lift curve tells us that even on unseen data our model can predict the spam mails successfully 2 to 2.5 times better than no/random selection model.

Below are the confusion matrices obtained from our model.



The matrix in the left is created on the whole of 1999 data which includes both the training as well as validation splits. 98% if the data is classified correctly and only 1.67% of the actual spam mails are predicted as non-spam mails while just 0.32% of the non-spam mails are classified as spam mails.

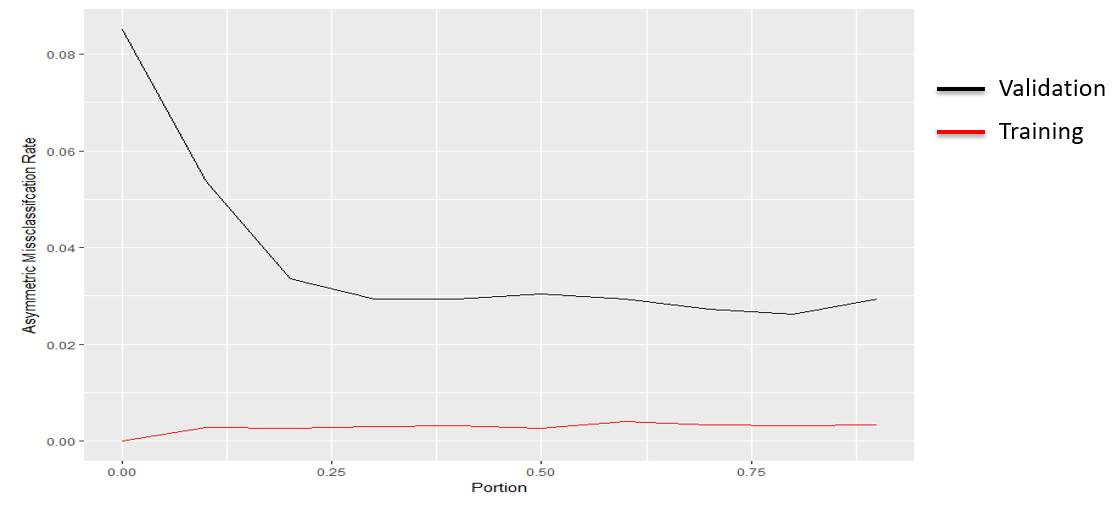
Since the first matrix contains records from both training and validation splits, it does not completely tell us about the model’s performance on unseen data only.

So, we generated a new dataset from getting spam mails from [Spamassassin](http://spamassassin.apache.org/old/publiccorpus/) and converting them to the format required by the model in R. The latest spam mails in this repository were from the year 2005. For non-spam mails, mails from the personal email id were used. Spam mails from the personal email id could not be used because all of them were in the form of images and not text. Also, the recent spam mails available in the online repositories were obtained using web-scraping and were in html format thereby having a lot of html tags which needed an additional step of parsing the html content which was out of scope in the current project.

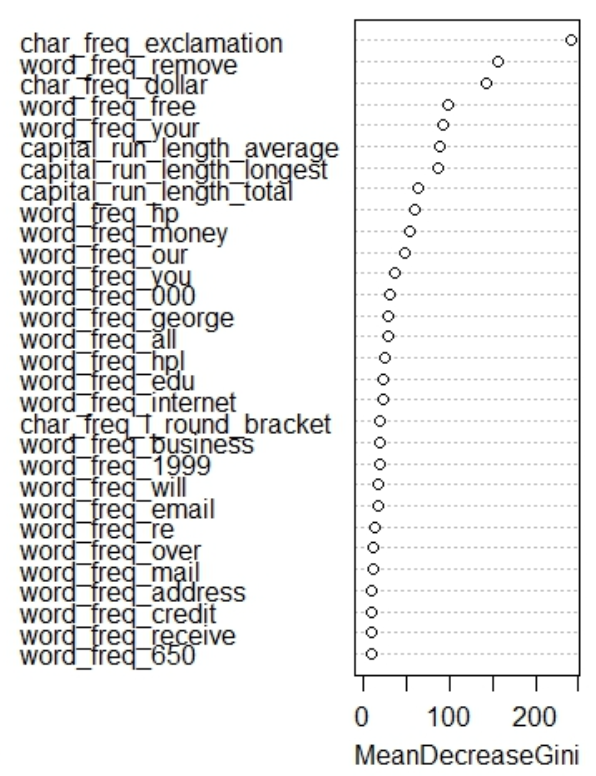
In 2005’s email data which had 100 rows among which 62 were spam mails and 32 were non-spam, 83% were classified correctly, 15% spam mails got classified as non-spam mails and only 2% non-spam mails were classified as spam mails. This shows that the model used for classifying spam mails which was trained on the data from 1999 still performs well on the data from the year 2005.

Also, for better understanding the final error in our model, we did a 100-fold cross-validation on the data from 1999 and we got a validation asymmetric misclassification rate of 5.04%

**Final Model Learning Curve:**



The above graph shows us the learning curve of our final model. It was created by using 10% subset of the 1999 data as the validation set which was kept constant. The model was trained on the same parameters mentioned above multiple times while the training data fed into the model was increased incrementally. The learning curve shows us that of the 90% of the whole 4601 observations, just around 30% of it i.e. around 1250 observations are enough to train the data to achieve the lowest validation error. Beyond that, any increase in data does not lead to any reduction in the validation error.

**Variable Importance as per Final Model Learning Curve:**

The image in the left shows the various columns and their contributions in the final model. These contributions are based on how much the various columns are contributing towards the decrease in impurity in the various trees present in the random forest.

It should be kept in in mind that the columns with high contribution do not mean that an increase in the frequency of those columns would lead to an increase in the probability of a mail being classified as a spam. Instead they could lead to an increase in the probability in either direction i.e. of a spam as well as that of a non-spam. The graphs created in the exploration part would help us I deciding whether an increase in frequency leads to an increase in the probability of spam or non-spam.

**Key Findings and Business Implications**

* **Improved Open Rate for Commercial Marketing mails:** Using the list of important predictors from the chosen model helps in widening the knowledge on words to avoid/include in commercial marketing emails, such that the marketing mail shifts from spam folder to important folder. This eventually improves the open rate of ad campaign mails
* **Personalization helps:** Usage of name, organization or address, increases the probability of a mail being classified as a non-spam
* **Cost Reduction:** Majority of spam costs stem from employees who spend working time identifying and deleting spam. The working time losses caused by spam are approximately 1,200 minutes per employee per year; these costs could be reduced by roughly 35% through the installation of a spam filter mechanism. The individual efficiency of a spam filter installation depends on the amount of spam that is received and on the level of knowledge about spam.
* **Relevance in today’s world:** We believe that our model works well in classifying all text-based spam mails on data as recent as today. Also, the model can be customized to include several additional word frequencies and character frequencies as predictors, in order to widen the spam mail lexicon

**Future Works:**

* **Handling picture-based spams:** Attaching an OCR module before the prediction model would help in identifying picture-based spams containing a single picture with embedded URLs
* **Word combinations:** Instead of using word/character frequencies as predictors, if the actual text mails can be accessed, we can use N-gram algorithm to train the model in identifying combinations of word and character occurrences, and use these combinations as predictors to enhance classification.

**References**

1. Spambase Data Set- Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.<https://archive.ics.uci.edu/ml/datasets/spambase>

1. Spamassassin- http://svn.apache.org/repos/asf/spamassassin/trunk/CREDITS

**Appendix**

#### SPAMBASE data set documentation

<https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.DOCUMENTATION>