TABLE OF CONTENTS:

1) PROJECT DESCRIPTION:

Project Description - Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving. This problem statement is targeted at enabling churn reduction using analytics concepts. Data Sets - 1) Test_data.csv 2) Train_data.csv

Problem statement - The objective of this Case is to predict customer behaviour. We are providing you a public dataset that has customer usage pattern and if the customer has moved or not. We expect you to develop an algorithm to predict the churn score based on usage pattern.

The predictors provided are as follows:

- account length
- international plan
- voicemail plan
- number of voicemail messages
- total day minutes used
- day calls made
- total day charge
- total evening minutes
- total evening calls
- total evening charge
- total night minutes
- total night calls
- total night charge
- total international minutes used
- total international calls made
- total international charge
- number of customer service calls made

Target Variable : move: if the customer has moved (1=yes; 0 = no)

1) PROBLEM STATEMENT:

Churn Prediction is a problem that is faced by brands across multiple domains such as E-Commerce and Telecom Industry. It is basically the process of losing customers to competition or due to some other factor. This is a classification problem where the target variable is a two class variable with values Yes or No. The independent variables could be a mix of categorical or numeric variables. We will apply multiple pre-processing techniques, exploratory data analysis and machine learning algorithms to predict and reduce churn in this project.

2) INTRODUCTION:

We start analysing the data by getting a glimpse of the rows of each variable and knowing their type. Here is a glimpse of the data that fits in R Studio:

```
Variables: 21
$ state                          <chr> "KS", "OH", "NJ", "OH", "OK", "AL", "MA...
$ `account length`               <int> 128, 107, 137, 84, 75, 118, 121, 147, 1...
$ `area code`                    <int> 415, 415, 415, 408, 415, 510, 510, 415,...
$ `phone number`                 <chr> "382-4657", "371-7191", "358-1921", "37...
$ `international plan`           <chr> "no", "no", "no", "yes", "yes", "yes", ...
$ `voice mail plan`              <chr> "yes", "yes", "no", "no", "no", "no", "...
$ `number vmail messages`        <int> 25, 26, 0, 0, 0, 0, 24, 0, 0, 37, 0, 0,...
$ `total day minutes`            <dbl> 265.1, 161.6, 243.4, 299.4, 166.7, 223....
$ `total day calls`              <int> 110, 123, 114, 71, 113, 98, 88, 79, 97,...
$ `total day charge`             <dbl> 45.07, 27.47, 41.38, 50.90, 28.34, 37.9...
$ `total eve minutes`            <dbl> 197.4, 195.5, 121.2, 61.9, 148.3, 220.6...
$ `total eve calls`              <int> 99, 103, 110, 88, 122, 101, 108, 94, 80...
$ `total eve charge`             <dbl> 16.78, 16.62, 10.30, 5.26, 12.61, 18.75...
$ `total night minutes`          <dbl> 244.7, 254.4, 162.6, 196.9, 186.9, 203....
$ `total night calls`            <int> 91, 103, 104, 89, 121, 118, 118, 96, 90...
$ `total night charge`           <dbl> 11.01, 11.45, 7.32, 8.86, 8.41, 9.18, 9...
$ `total intl minutes`           <dbl> 10.0, 13.7, 12.2, 6.6, 10.1, 6.3, 7.5, ...
$ `total intl calls`             <int> 3, 3, 5, 7, 3, 6, 7, 6, 4, 5, 6, 5, 2, ...
$ `total intl charge`            <dbl> 2.70, 3.70, 3.29, 1.78, 2.73, 1.70, 2.0...
$ `number customer service calls` <int> 1, 1, 0, 2, 3, 0, 3, 0, 1, 0, 4, 0, 1, ...
$ Churn                          <chr> "False.", "False.", "False.", "False.",...
```

After going through the data we can say that there are 21 variables with character, double and integer type of data. There are a total of 3333 rows in the dataset of train data we are working on.

3) DATA PRE-PROCESSING:

4.1) Missing Value Analysis:

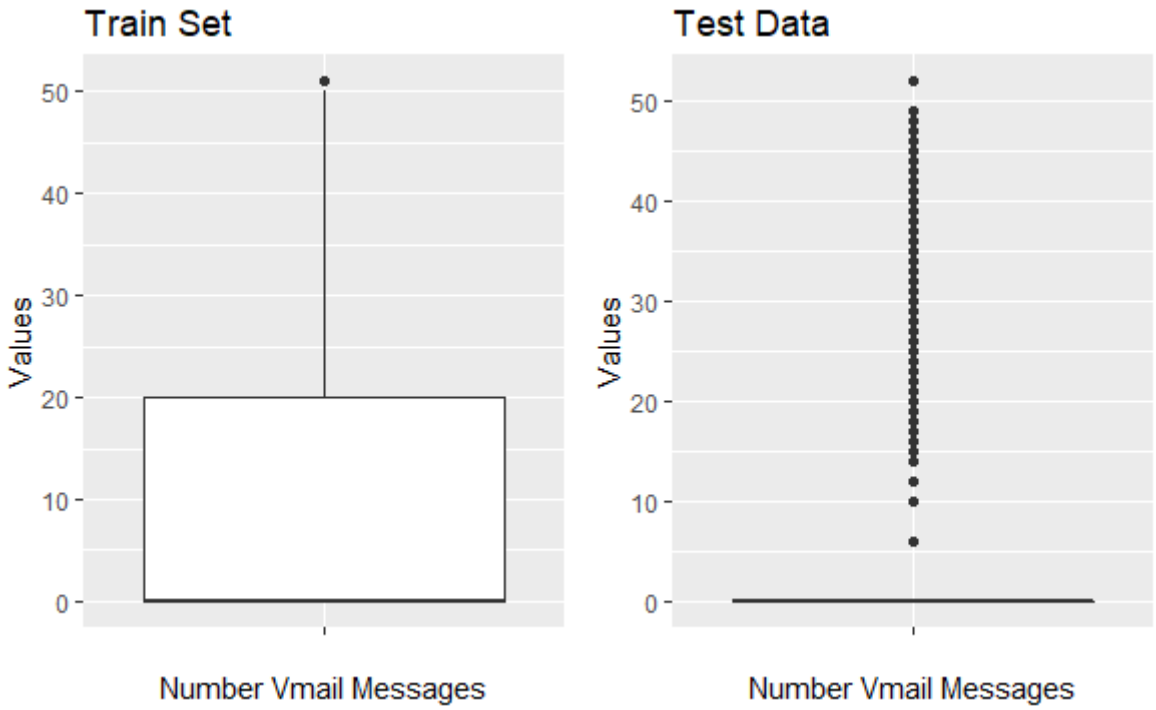
We start the Pre-Processing with Missing Value Analysis. Missing Values are the values in the data that are represented by the value NA. They are either imputed or deleted from the data based on their frequency in the variables.

Upon doing this analysis on our train and test data, we find that there are no missing values in both the data sets. Hence, there is no need of working on any kind of missing value treatment for this data.

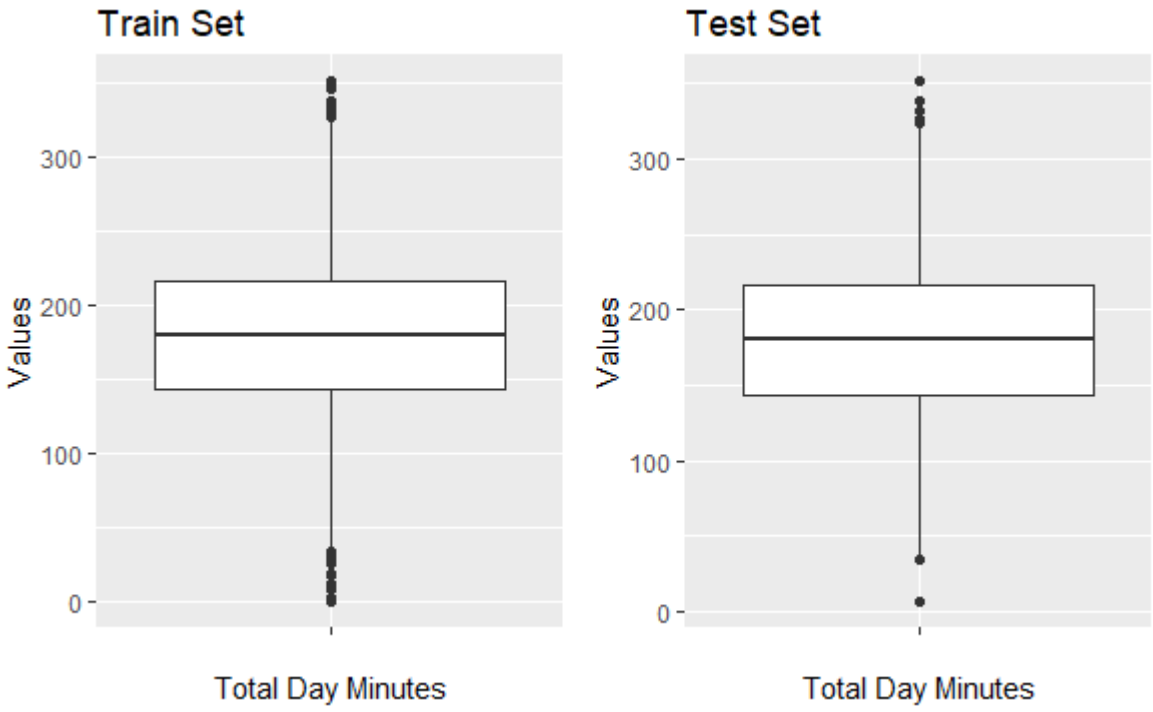
4.2) Outlier Analysis:

We perform outlier analysis on the train and test data side by side to understand the range of values the numeric variables have in both train and test set. Whether the outliers have to be imputed or deleted will be decided after viewing the boxplots of each numeric variable.

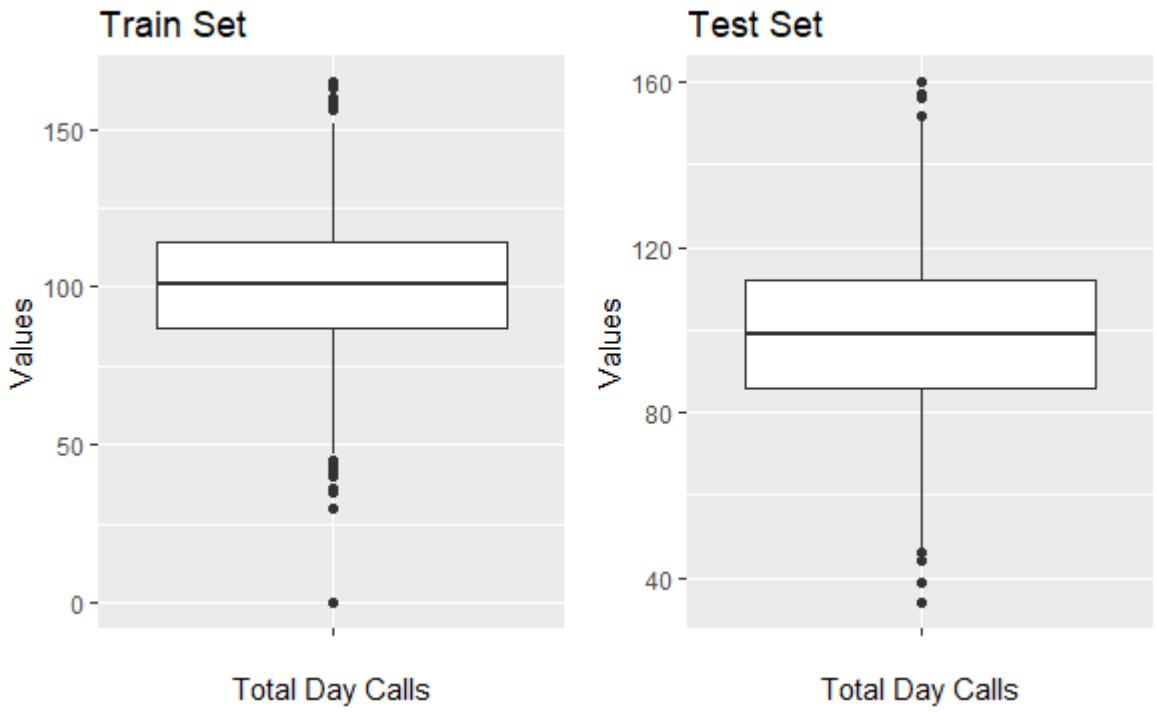
Boxplot of Number Vmail Messages:



Boxplot of Total Day Minutes:



Boxplot Of Total Day Calls:



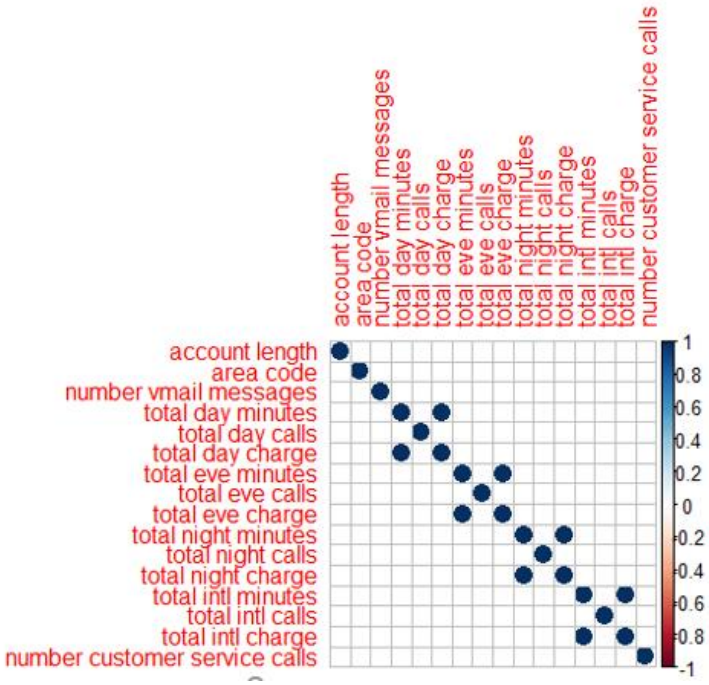
I will not elaborate on the boxplots of all the numeric variables as we are not going to impute or delete them. From the above figures, we can conclude that there are outliers in the test data also. The machine learning model should be trained to predict these outliers. If they are imputed or deleted, it might cause huge out of sample error even if we achieve low in sample error rate. Hence, the outliers are not to be deleted.

4.3) Feature Selection:

We have to perform tests to check the multi collinearity between the variables that might deteriorate our machine learning model's performance. For the purpose of bivariate multi collinearity, we perform correlation analysis for numeric variables and Chi Square tests on categorical variables.

Correlation Analysis:

We plot the correlation plot to understand if we have any bivariate collinearity between variables.



We can see from the above correlation plot that there are four pairs of correlated variables. Following are the pairs of correlated variables:

- total day charge ~ total day minutes
- total eve charge ~ total eve minutes
- total night charge ~ total night minutes
- total intl charge ~ total intl minutes

We have to eliminate one variable in each pair while building the model or apply principle component analysis to make sure our machine learning model does not face the problem of multicollinearity.

Now calculating correlation between categorical variables using the Chi Square Tests. We get the following table after doing the Chi Square analysis:

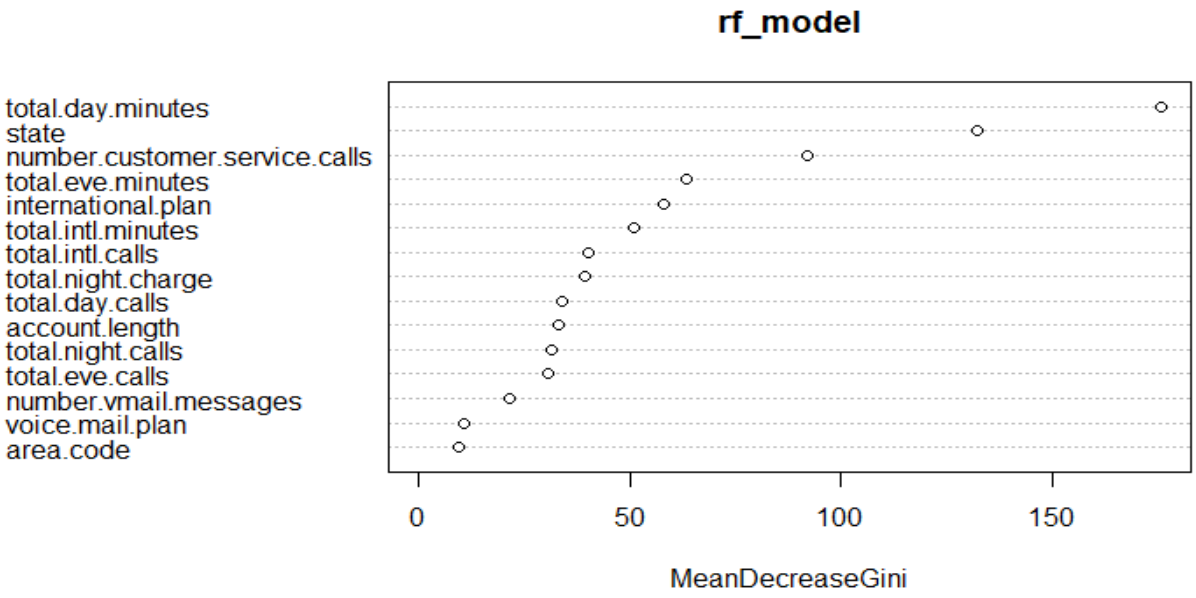
Parameter	X-Squared	df	p-value
Voice Mail Plan	34.134	1	5.15e-9
State	83.04	50	0.002
Phone Number	3333	3332	0.4919
International Plan	222.57	1	2.2e-16

As we can see, Phone Number has a p-value greater then 0.05. This indicates that this variable is statistically unimportant for predicting Churn. Hence, we directly remove it from both train and test data.

We perform Pre-Processing or Principle Component Analysis in the model development stage to deal with the collinearity problem in numeric variables. The PCA treatment will turn the variables that are collinear into orthogonal directions where the data spreads the most in multiple iterations. This will help us avoid the collinearity problem and also retain the variable information. Some algorithms might give better results by removing the features that are collinear. We will have to decide this during the model development stage.

4.4) FEATURE IMPORTANCE:

We calculate the feature importance using the random forest model. A random forest model is built on the entire dataset and the Mean Decrease In Gini of each variable is calculated. This number is then plotted on a graph to represent all the variables in decreasing order of importance. Here is the plot representing the Mean Decrease in Gini of all the variables.

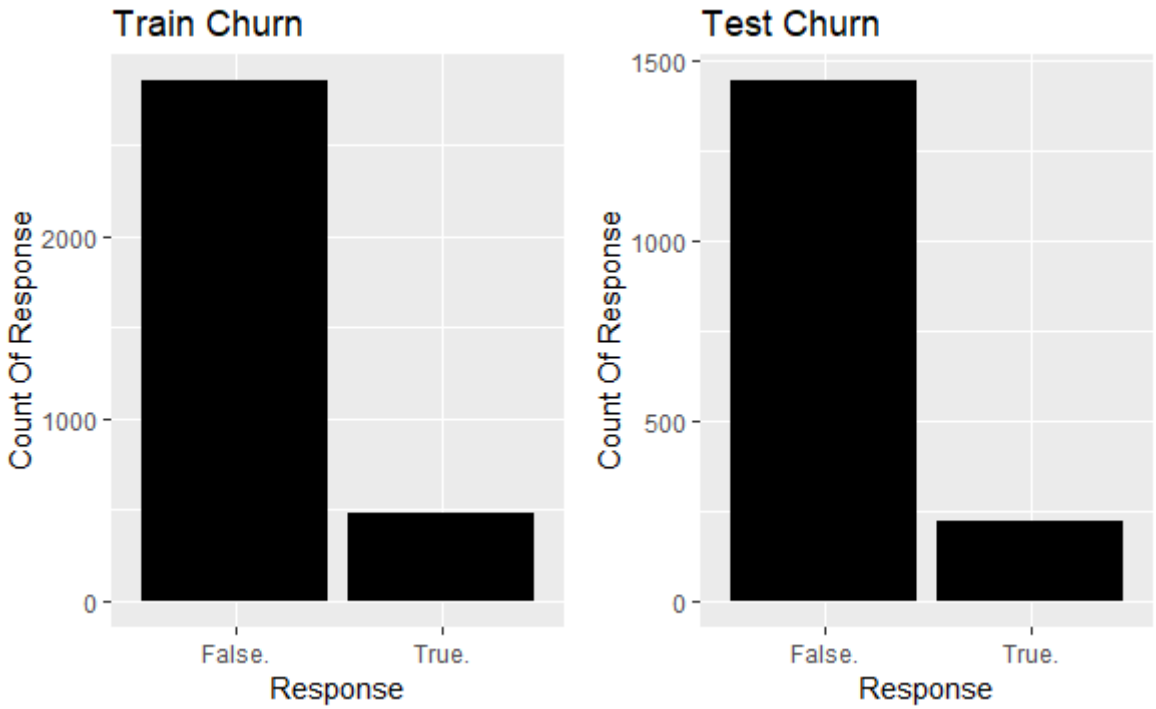


As we can see, variables such as state, total day charge, total day minutes and number of customer service calls are important predictors of Churn in our Data Set.

4) EXPLORATORY DATA ANALYSIS & RECOMMENDATIONS:

Before developing the model, we try to get a feel for the data and derive valuable insights. This could help us in developing the model and expecting tangible results.

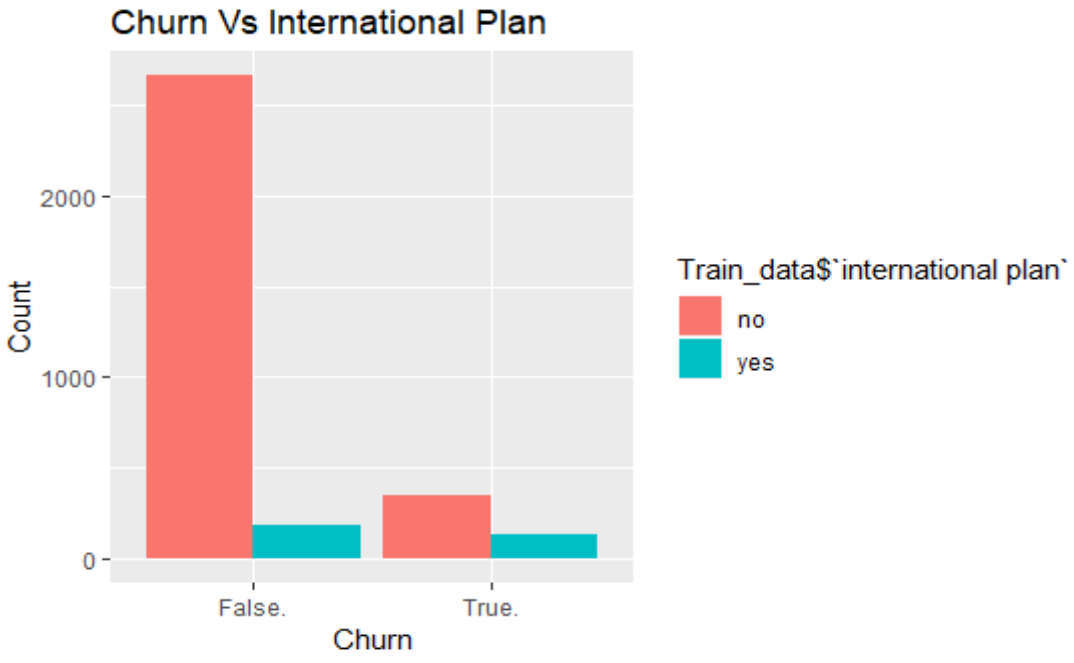
Understanding the two classes of the Predictor variable “Yes” or “No” and its frequency on a bar graph using ggplot:



As we can see we have a class imbalance problem here in the Train and Test Data. This should be accounted for while developing the model. Since the train and test data both have a class imbalance problem, we cannot upscale our train data using synthetic methods.

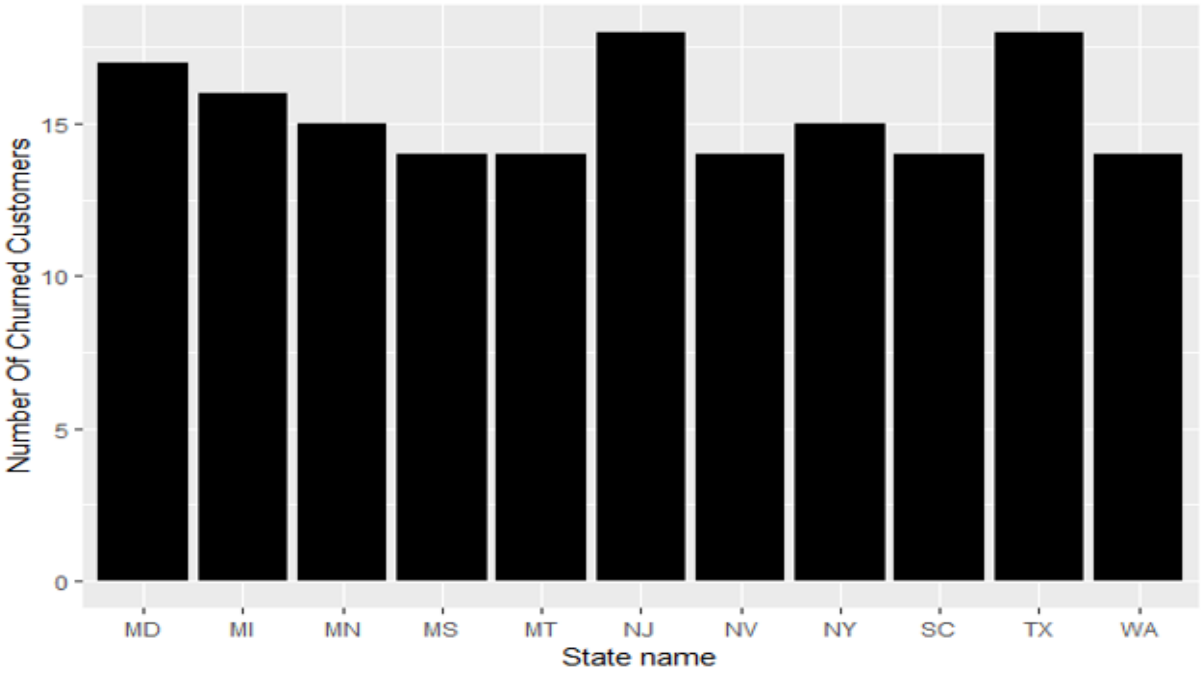
Since we are developing the model on the train data we will be drawing insights only on the train dataset.

Trying to understand how the international plans effect the Churn of a customer, we draw a bar plot using ggplot to visualize the two categorical variables:



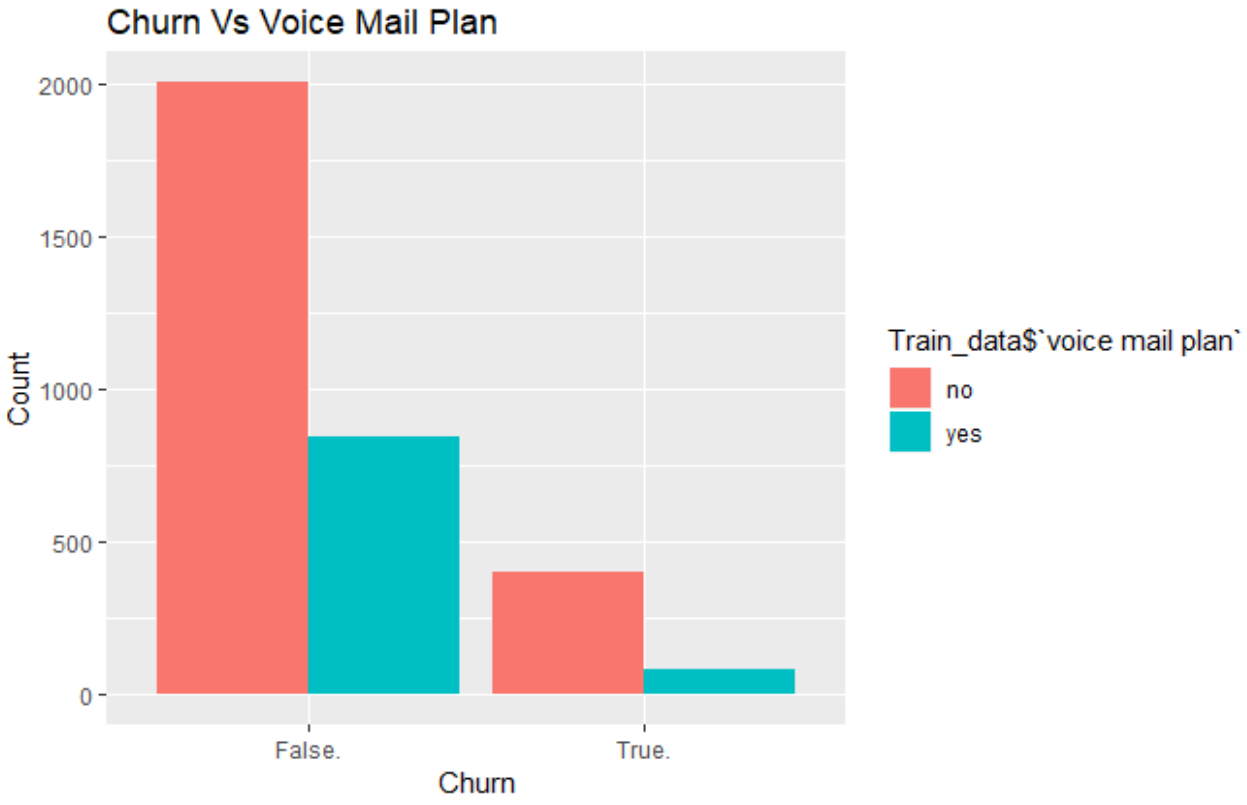
As we can see, a majority of the customers that do not churn did not have International plans. Since the number of customers that did not have international plans are a majority in Churn and retained customers, nothing can be concluded from this graph.

Studying the effects of geographic location on the Churn, we plot a bar plot of the top five states where the Churn was prominent.



As we can see, these are the top 10 states where churn is reported. Among these, NJ and TX are the states where the Churn is maximum. If at all the company plans on doing a customer feedback survey, it should start from these states.

Let's check the effect of Voice Mail Plan on Churn:

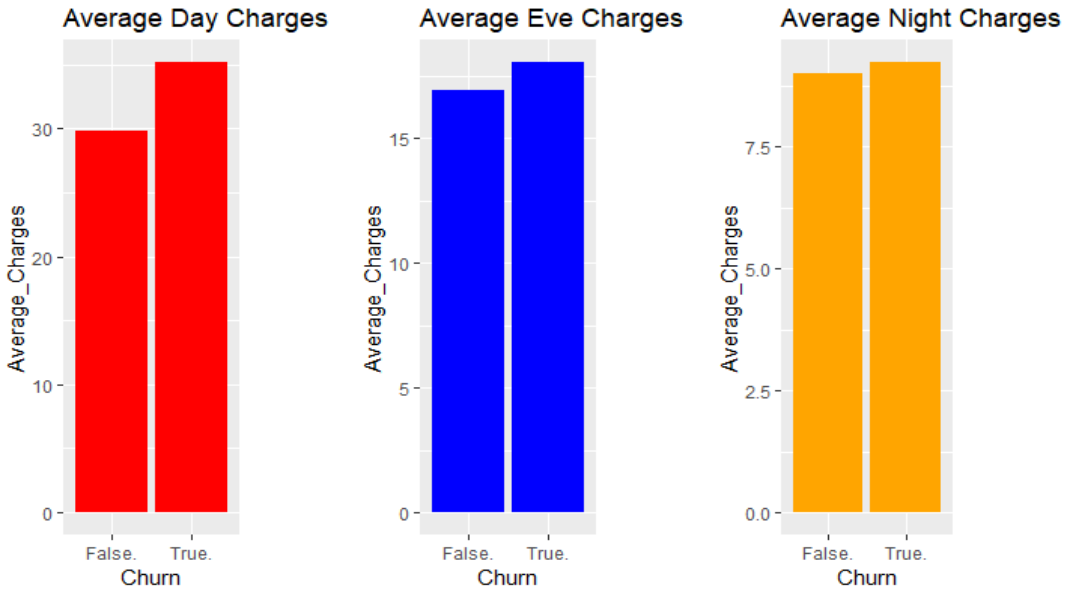


As we can see, there are no clear insights that can be drawn from this as both Churn classes have the same pattern of voice plans.

Let us check the effect of total charges on the Churn of Customers. These charges are of three types:

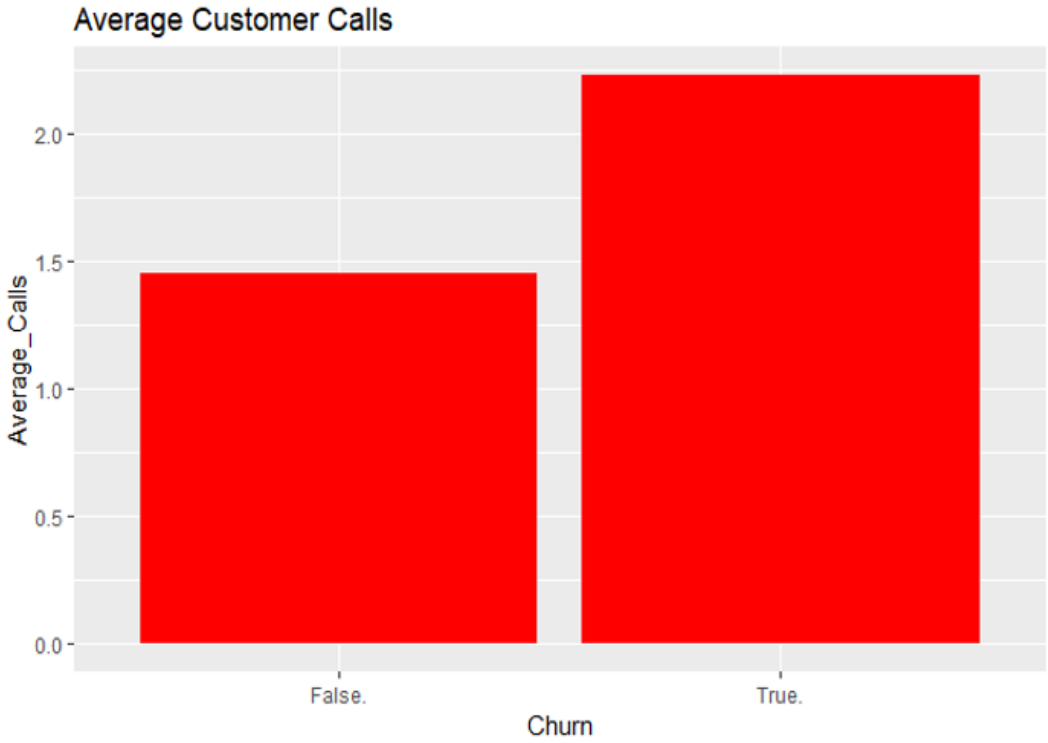
1) Day Charge
2) Evening Charge
3) Night Charge

The following graph shows a relationship between the Churn and Average Charges:



As we can see, the average charges for Customers who have churned were higher. The higher charges to customers could have been one possible reason for the Churn of customers.

Now we try to look at the average number of customer service calls in relation to the Churn.



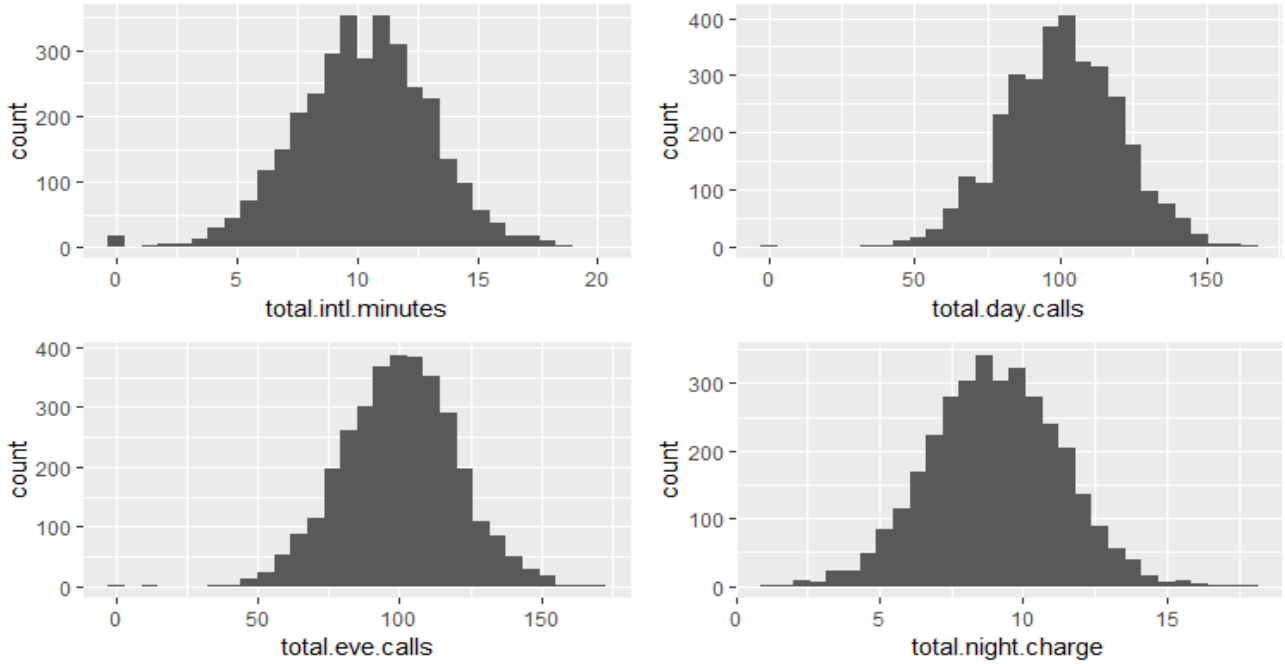
We can see that the average number of customer calls were high for the customers that churned. This could indicate an inefficiency on the part of the customer service department in resolving the customer's problems.

6)MODEL DEVELOPMENT:

After doing the Pre-Processing of the data, we move to model development where we develop a predictive model to fit on our train data and predict the responses on the test data.

Let's first check the spread of numeric variables in the data before developing the model.

Checking for the normalization of the data using ggplot:



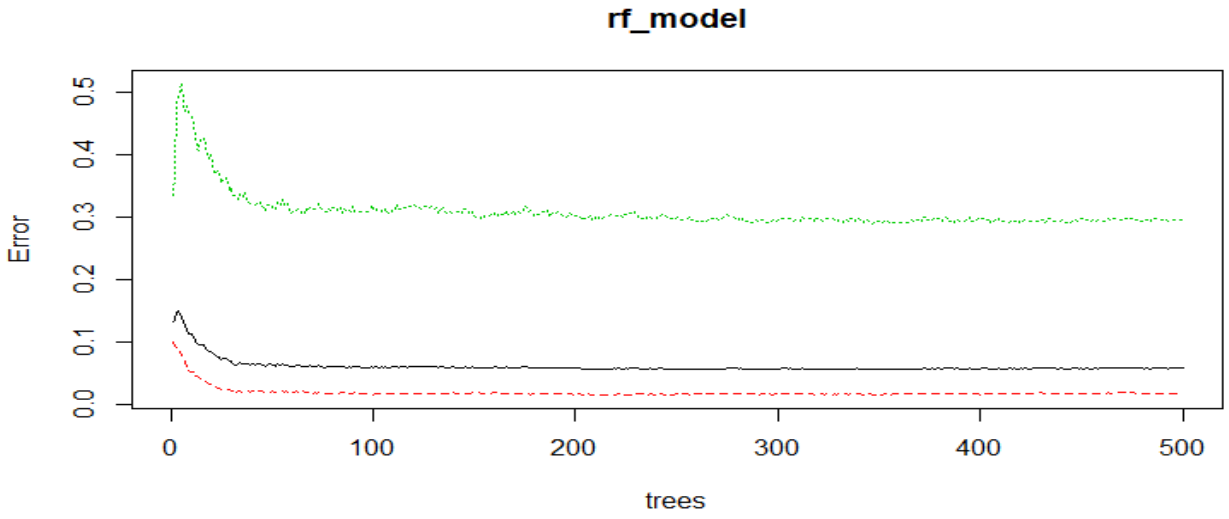
As we can see the data is normalized. Feature Scaling is not important for certain type of algorithms. If the algorithm is a distance based algorithm, then we have to make sure that feature scaling is done. But in tree based and ensemble models, the actual values are more important than the scaled values. In the subsequent sections, we will scale the data if needed.

We remove the variables that are causing multicollinearity by calculating the collinearity between them and removing the one variable from each of the collinear variables. The list for numeric variables is named as list_removed in the code.

6.1) RANDOM FOREST MODEL:

Random Forest Model is an ensemble model which creates Trees by bagging variables in multiple iterations and creating a forest of decision Trees by performing splits at each node on the basis of Mean Decrease in Gini. At the onset, we fit a primitive random forest model on our test data to check the accuracy and false positive rate.

After fitting the primitive random forest model named rf_model, we plot it on a graph to see the error vs number of trees plot:



As we can see, the three colors of lines represent different types of splitting rules and the errors become static once the number of trees have crossed 100. So there is no point of having ntree variable more than 100.

Once we fit the primitive random forest model, we predict on the test data set using the same model. After making the predictions, the actual predictor values and the prediction are put into a confusion matrix. The following table and statistics are obtained for rf_model:

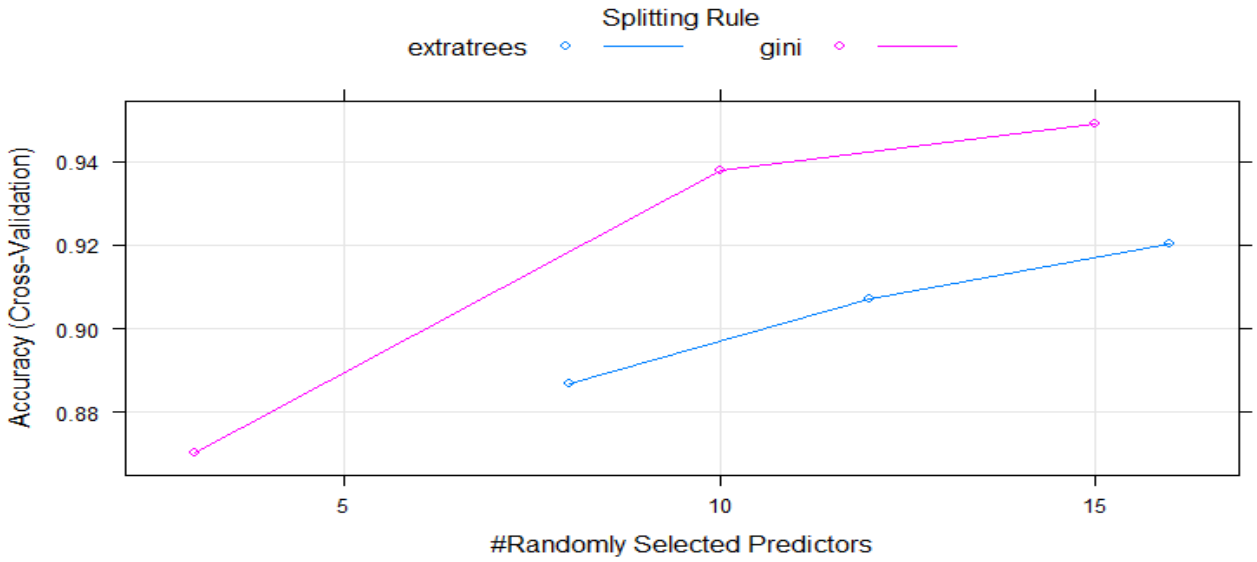
```
Confusion Matrix and Statistics

          predictions_rf_model
           False. True.
  False.    1419    24
  True.       65   159

                Accuracy : 0.9466
                  95% CI : (0.9347, 0.9569)
    No Information Rate : 0.8902
    P-Value [Acc > NIR] : 4.696e-16

                   Kappa : 0.7513
  Mcnemar's Test P-Value : 2.235e-05

             Sensitivity : 0.9562
             Specificity : 0.8689
          Pos Pred Value : 0.9834
          Neg Pred Value : 0.7098
              Prevalence : 0.8902
          Detection Rate : 0.8512
    Detection Prevalence : 0.8656
       Balanced Accuracy : 0.9125
```

The accuracy is is **94.66** percent. The false positive rate seems to be 65/(65+159) = 0.2901 or **29.01** percent. We will try to improve on the accuracy by reducing the false negatives(24) and false positives(65). We can achieve this by Hypertuning the parameter mtry and performing cross validations on the data using a custom tuning grid.

6.2) Hypertuning Random Forest Parameters:

We hypertune the parameter mtry from 2 to 20 and choose a min.node.split of 1.Here we use a train control to perform the cross validations. We also give a PreProcess argument of “corr” to the train function of the caret package.

All these arguments are fed to the train function of the Caret package in order to improve efficiency and computational time. After fitting the model, we get the following plot of accuracy vs #Randomly Selected Predictors”



As we can see, the splitting rule gini gives the best results for an mtry value of 15. The name of the final tuned model after aggregating all the data is model_tuned.

We check the accuracy and false positive rate of this data by computing the confusion matrix:

```
Confusion Matrix and Statistics

          predictions_model_tuned
           False. True.
  False.    1439     4
  True.       62   162

                Accuracy : 0.9604
                  95% CI : (0.9499, 0.9692)
    No Information Rate : 0.9004
    P-Value [Acc > NIR] : < 2.2e-16

                   Kappa : 0.8089
  Mcnemar's Test P-Value : 2.28e-12

             Sensitivity : 0.9587
             Specificity : 0.9759
          Pos Pred Value : 0.9972
          Neg Pred Value : 0.7232
              Prevalence : 0.9004
          Detection Rate : 0.8632
    Detection Prevalence : 0.8656
       Balanced Accuracy : 0.9673
```

As indicated by the confusion matrix summary, the accuracy has improved to **96.04** percent from the previous **95.56** percent. The false positive rate has reduced from **29.01** to **26.78** percent. Parameters such as sensitivity and specificity have also increased giving an indication that our model is tuned to its best performance for the mtry value of **15**,splitting rule called **gini** and min.split.node value of **1**.

6.3) Logistic Regression:

As we can see, the random forest model has a very high accuracy. But there is a major drawback with Random Forest Model. We cannot tune the model for an imbalanced class problem in the response variable. Logistic Regression Model does give us the flexibility to tune the model to predict “True.” cases with higher accuracy than the over represented “False.” cases. Logistic Regression model might overcome this problem as we can tune the threshold value to configure the false positive rate.

Let us fit the logistic regression model to the train data. The trained model is called logistic_model. We pick a value of p = 0.5 to make some baseline predictions. After doing this, we make the confusion matrix. Here is the stats of the confusion matrix:

	False	True
False	1396	47
True	169	55

We calculate the accuracy, sensitivity and specificity for this baseline model.

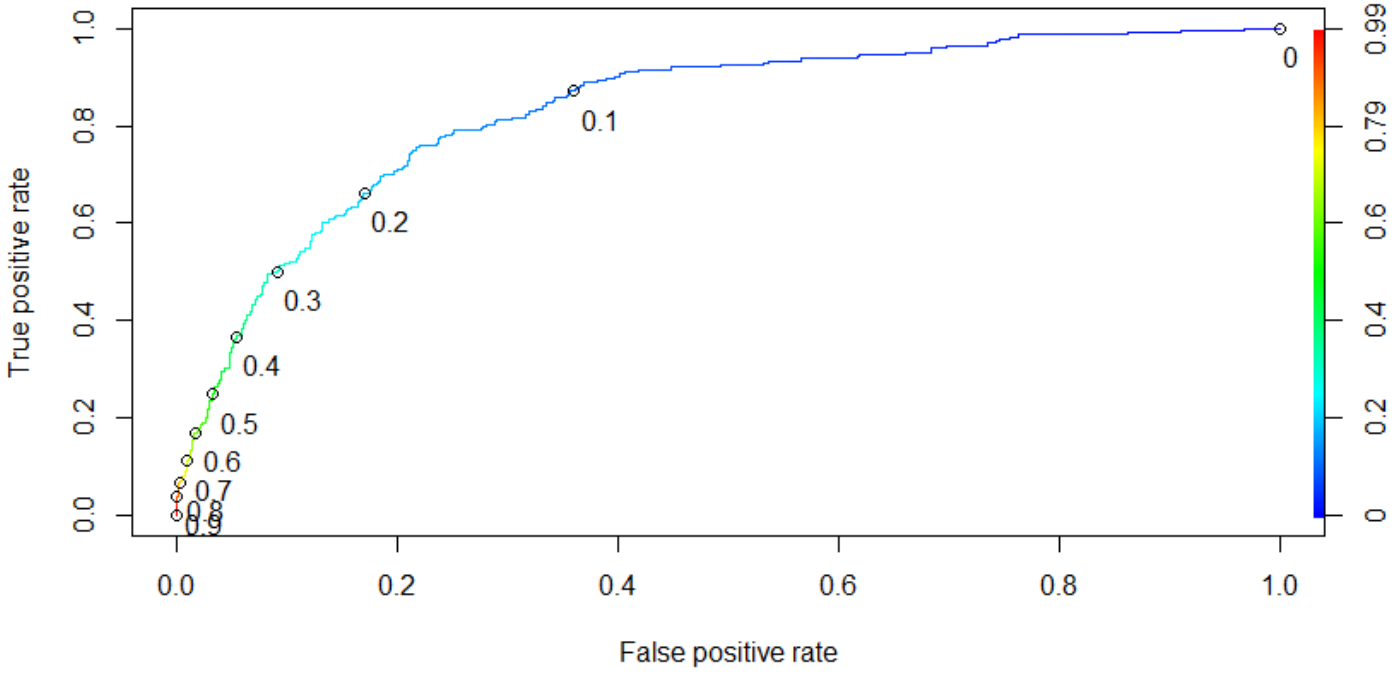
Accuracy: (TP+TN/Total) -> (55+1396)/1667 = 86.98

Sensitivity: (TP/TP+FN) -> (55/55+47) = 53.92

Specificity: (TN/TN+FP) -> (1396/1396+169) = 89.20

As we can see, the specificity is very high and the sensitivity is really low leading to a lot of True Churn cases to be misclassified as False. Our business requirement is to increase the sensitivity to improve the accuracy of the True Churn prediction rate. Let us hypertune the model by changing the threshold value to find an optimized model with a reasonable trade-off between sensitivity(True Positive Rate) and Specificity(False Positive Rate).

6.4) Hypertuning Logistic Regression Model:



After studying the ROC curve, we start experimenting with threshold values ranging between 0.08 to 0.2. After experimenting with a lot of values we finalize on the value p = 0.1. The following is the confusion matrix we obtain after hyper tuning the model at p = 0.1:

	False	True
False	922	521
True	26	198

Calculating the accuracy, sensitivity and specificity for the tuned model:

Accuracy: (TP+TN/Total) -> (922+198)/1667 = 67.18%

Sensitivity: (TP/TP+FN) - > (198/198+521) = 27.06%

Specificity: (TN/TN+FP) -> (922/922+26) =97.25%

This threshold value gives us a reasonable trade-off between the accuracy and the sensitivity. The accuracy must have decreased but the model is predicting the True Churn more effectively.

6.5) DEVELOPMENT OF XGBoost Model:

Xgboost is an advanced Gradient Boosting algorithm that has very high computational speed and performs better than other tree based ensemble models. We develop the xgboost model using some of the default values to its parameters. After that we try to gauge its performance by tabulating a confusion matrix to calculate accuracy, sensitivity and specificity.

After completing the development of model, here is the result of the tabulation of confusion matrix:

	False	True
False	1443	0
True	0	224

As we can see, this model gives a surprisingly high accuracy of 100%. Let us compare the performances of these three models and conclude on the use of one for this data.

Conclusion:

Now that we have developed two types of model that are optimized, we have to decide which one is a better fit for our business problem.

Model Name	Accuracy	Sensitivity	Specificity
Random Forest	95.86%	71.87%	97.04%
Logistic Regression	67.18%	27.06%	97.25%
XgBoost	100%	100%	100%

As we can see, the Xgboost model with default parameters is giving 100% accuracy. I would recommend the Xgboost model for predicting Churn on this data set as its computational speed is high and the accuracy also seems to be better than the other two models.

REFERENCES:

- Machine Learning Mastery By Jason Brownlee
- Stack Overflow
- Datacamp