Churn Reduction

Sudipto Mukherjee

31st October, 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

Basically, customer churn occurs when the customers stop doing business with a company or service. No customer churn is an important attribute in the growth of a company since it is less expensive to retain the existing customer than it is to acquire the new ones as the company has already acquired the trust and loyalty of the existing customers. The aim of this project is to study and predict the customer behaviour to enable to organisation to make changes in the rules and policies so as to reduce customer churn.

1.2 Data

Our task is to build a classification model based on the data which will basically classify the churn into yes or no thereby specifying the rules due to which the customers churn out frequently.

A sample of the dataset is given below.

Table 1.1 Train Data (Columns 1 - 7)

State	Account Length	Area Code	Phone Number	International Plan	Voice Mail Plan	Number Vmail Messages
KS	128	415	382-4657	No	Yes	25
OH	107	415	371-7191	No	Yes	26
NJ	137	415	358-1921	No	No	0
OH	84	408	375-9999	Yes	No	0
OK	75	415	330-6626	Yes	No	0

Table 1.2 Train Data (Columns 8 - 16)

Total Day Minutes	Total Day Calls	Total Day Charge	Total Eve Minutes	Total Eve Calls	Total Eve Charge	Total Night Minutes	Total Night Calls	Total Night Charge
265.1	110	45.07	197.4	99	16.78	244.7	91	11.01
161.6	123	27.47	195.5	103	16.62	254.4	103	11.45
243.4	114	41.38	121.2	110	10.3	162.6	104	7.32
299.4	71	50.9	61.9	88	5.26	196.9	89	8.86
166.7	113	28.34	148.3	122	12.61	186.9	121	8.41

Table 1.3 Train Data (Columns 17-21)

Total Intl Minutes	Total Intl Calls	Total Intl Charge	Number Customer Service Calls	Churn
10	3	2.7	1	False
13.7	3	3.7	1	False
12.2	5	3.29	0	False
6.6	7	1.78	2	False
10.1	3	2.73	3	False

As seen from the above table, below are the predictor variables using which we have to predict whether the customer(s) will churn out or not.

Table 1.4 Predictor Variables

S. No	Predictors
1	Account Length
2	International Plan
3	Voicemail Plan
4	Number of Voice Mail Messages
5	Total Day Minutes used
6	Total Calls Made
7	Total Day Charge
8	Total Evening Minutes used
9	Total Evening Calls made
10	Total Evening Charge
11	Total Night Minutes used
12	Total Night Calls made
13	Total Night Charge
14	Total International Minutes used
15	Total International Calls made
16	Total International Charge
17	Number of Customer Service Calls made

Chapter 2

Methodology

2.1 Data Pre-processing

Data pre-processing is basically a data mining technique that involves transforming raw data into a format that is understandable. In here we explore the data, understand the meaning of each and every aspect of it including the variables and what each variable means and try to figure out how and in what way the variable will be influencing the target variable.

In the given data set, as a part of data pre-processing, we first remove the variables which are not contributing towards the target variable which is basically whether the customer will churn out. Those variables were state, phone number and area code. After that, we converted the categorical variables (with factor datatype) to their corresponding levels. Following is the code snippet to execute the same.

```
for(i in 1:ncol(TrainData))
{
  if(class(TrainData[,i]) == 'factor')
  {
    TrainData[,i] = factor(TrainData[,i], labels=(1:length(levels(factor(TrainData[,i])))))
  }
}
```

2.1.1 Missing Value Analysis

The concept of missing values is important to understand in order to successfully manage the data. If the missing values are not handled properly, then inaccurate inferences may be drawn about the data. Also, due to improper handling, the result obtained will be different from the ones where missing values are present.

In the given dataset, there were no missing values found. Following is the code snippet to know the same.

```
#Missing Value Analysis
MissingValue = data.frame(apply(TrainData,2,function(f){sum(is.na(f))}))
```

Following is the sample output obtained.

account.length	0
international.plan	0
voice.mail.plan	0
number.vmail.messages	0
total.day.minutes	0
total.day.calls	0
total.day.charge	0
total.eve.minutes	0
total.eve.calls	0
total.eve.charge	0
total.night.minutes	0
total.night.calls	0
total.night.charge	0
total.intl.minutes	0
total.intl.calls	0
total.intl.charge	0
number.customer.service.calls	0
Churn	0

2.1.2 Outlier Analysis

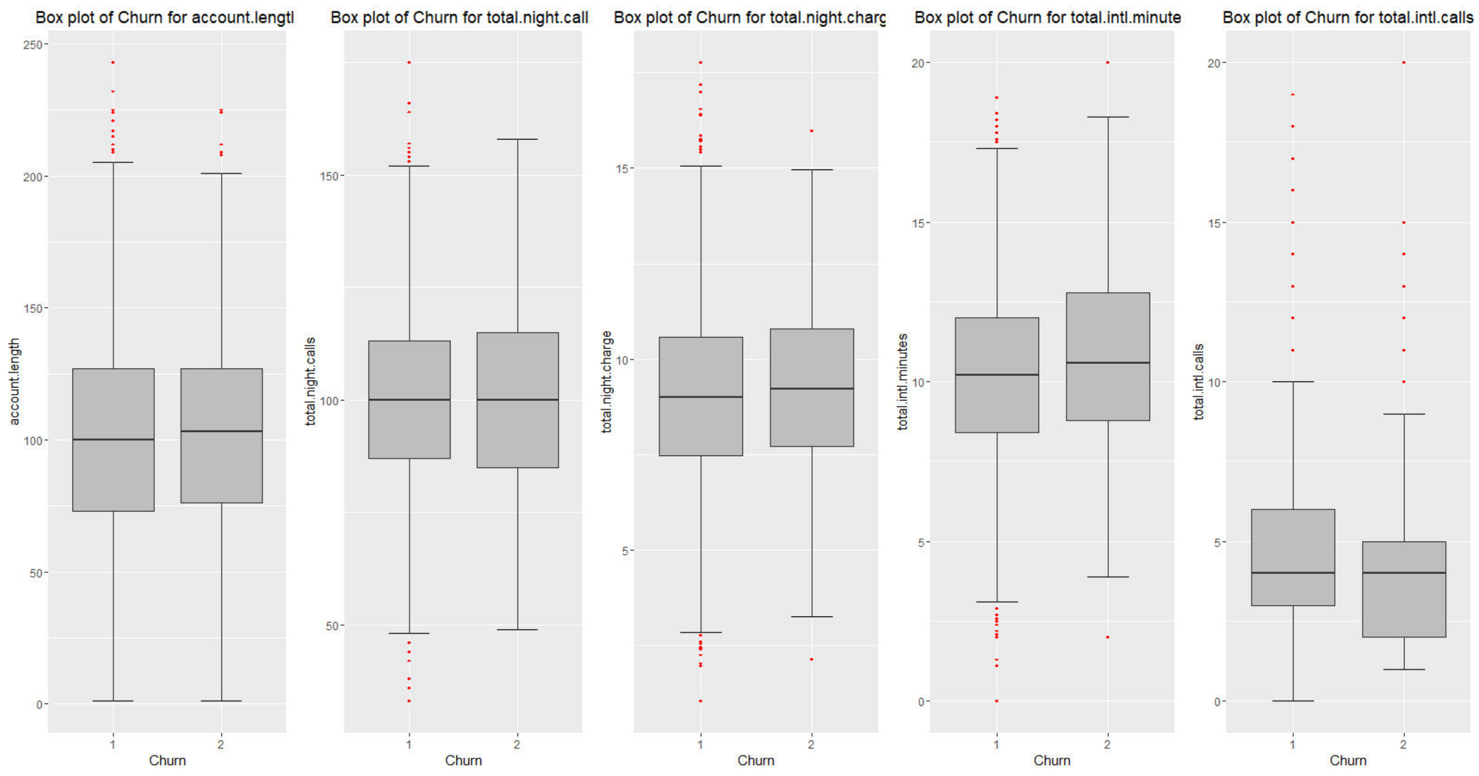
Outliers are the extreme values that fall a long way outside the other observations. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times and less accurate models. There has been a constant discussion on whether to remove the outliers or not.

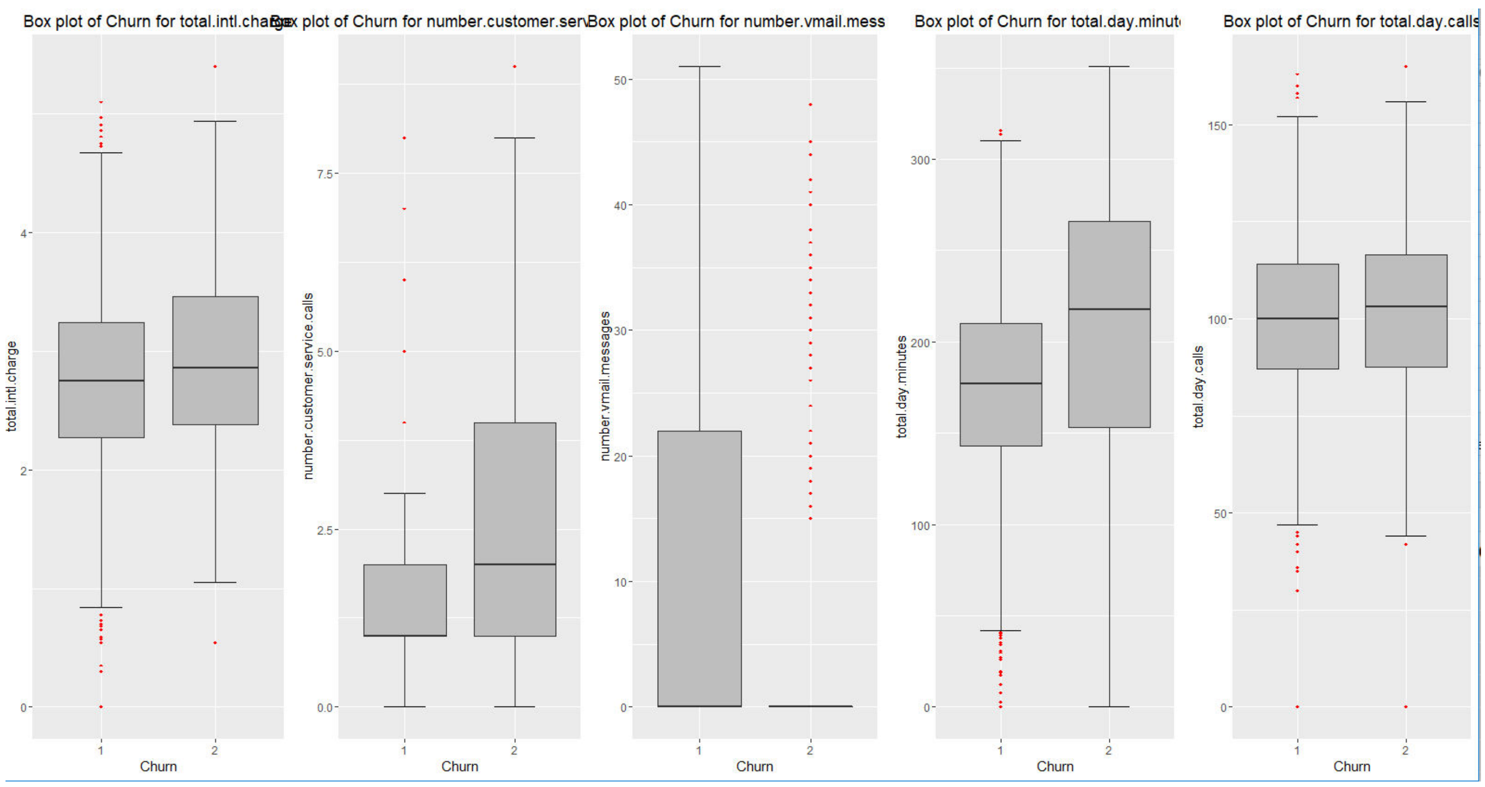
The outliers can be dropped in situations like if we have a lot of data and the sample won't be hurt by dropping the questionable data or if we have an extremely good sense of within what range will the data of a particular fall, then we can drop it.

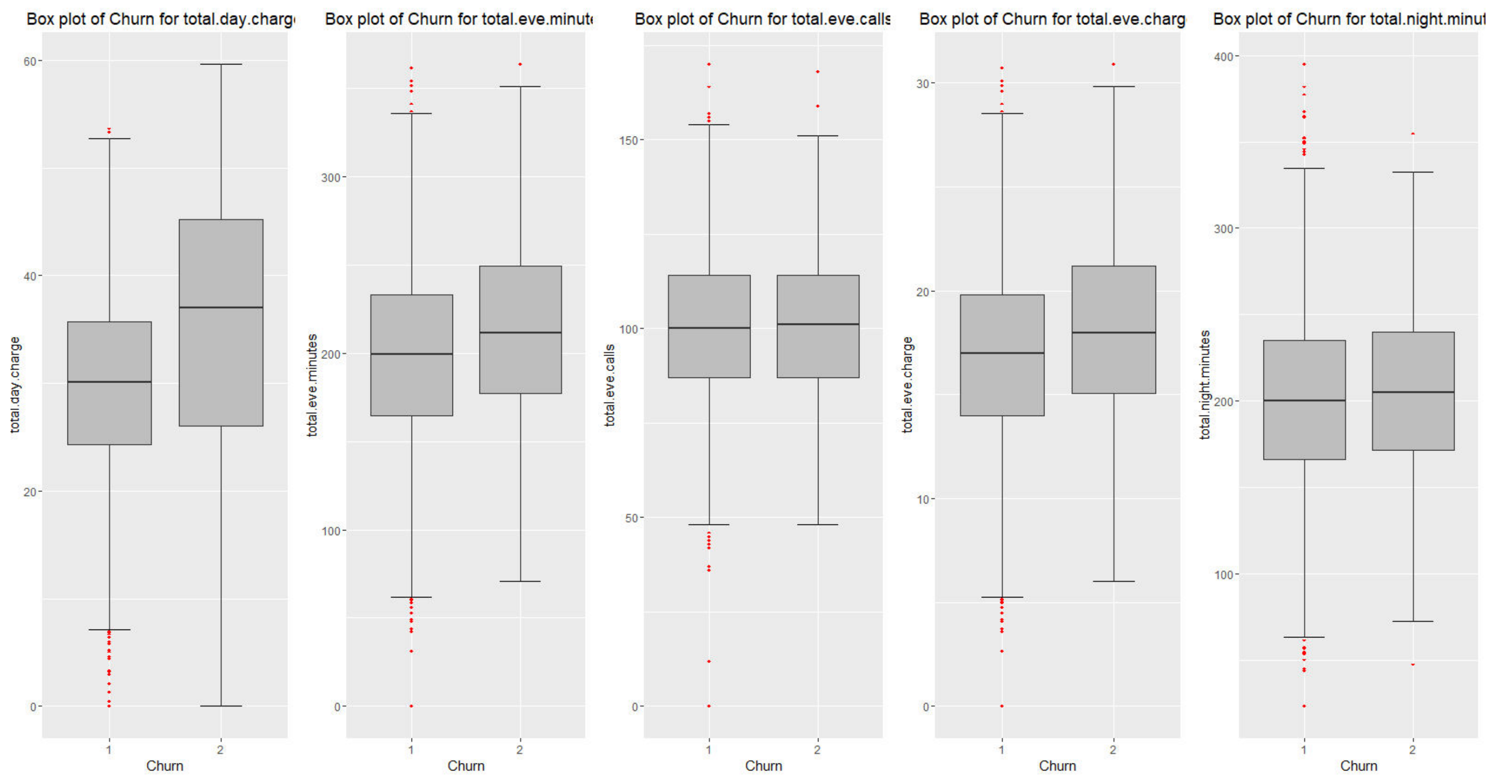
It is not recommended to drop an outlier if the results are critical like analysing the symptoms of a deadly disease. Also, in scenarios where the outliers constitute a major percent of the data, it is considered not to drop the outliers.

For the given data, we have plotted the boxplots for all the predictor variables. Following are the code fragment and corresponding box plots.

```
NumericVariables_Index = sapply(TrainData,is.numeric)
NumericVariables = TrainData[,NumericVariables_Index]
ColumnNames = colnames(NumericVariables)
for (i in 1:length(ColumnNames))
{
 assign(paste0("BoxPlot_Train",i), ggplot(aes_string(y = (ColumnNames[i]), x = "Churn"),
                                           data = subset(TrainData))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                       outlier.size=1, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(y=ColumnNames[i],x="Churn")+
          ggtitle(paste("Box plot of Churn for",ColumnNames[i])))
}

gridExtra::grid.arrange(BoxPlot_Train1,BoxPlot_Train10,BoxPlot_Train11,BoxPlot_Train12,BoxPlot_Train13,ncol=5)
gridExtra::grid.arrange(BoxPlot_Train14,BoxPlot_Train15,BoxPlot_Train2,BoxPlot_Train3,BoxPlot_Train4,ncol=5)
gridExtra::grid.arrange(BoxPlot_Train5,BoxPlot_Train6,BoxPlot_Train7,BoxPlot_Train8,BoxPlot_Train9,ncol=5)
```




Box plot of Churn for total.intl.charge
total.intl.charge
Churn
Box plot of Churn for number.customer.serv
number.customer.service.calls
Churn
Box plot of Churn for number.vmail.mess
number.vmail.messages
Churn
Box plot of Churn for total.day.minut
total.day.minutes
Churn
Box plot of Churn for total.day.calls
total.day.calls
Churn


Box plot of Churn for total.day.charg
total.day.charge
Churn
Box plot of Churn for total.eve.minut
total.eve.minutes
Churn
Box plot of Churn for total.eve.calls
total.eve.calls
Churn
Box plot of Churn for total.eve.charg
total.eve.charge
Churn
Box plot of Churn for total.night.minut
total.night.minutes
Churn

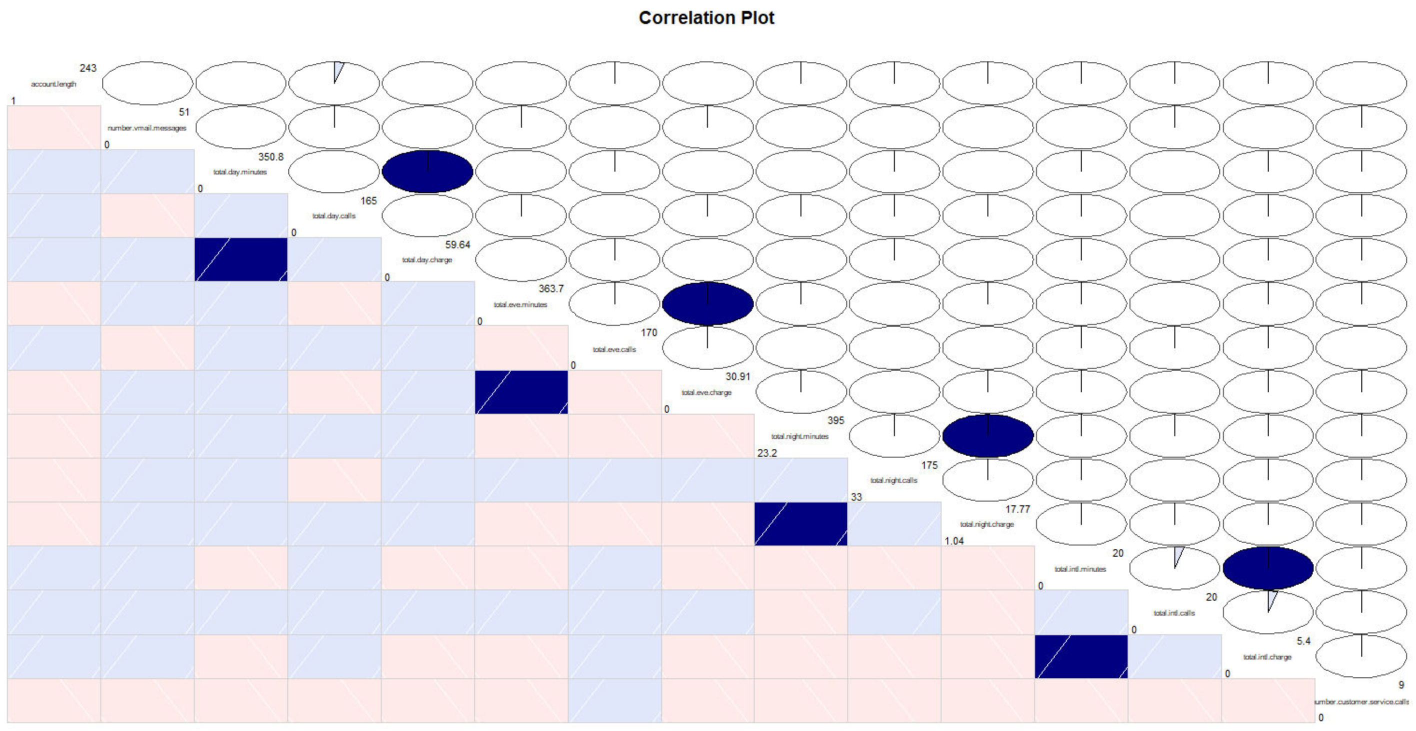
2.1.3 Feature Selection

Feature selection basically refers to reducing the inputs for processing and analysis or of finding the most meaningful inputs. Data now almost always contains way more information than it is needed to build a model. And also, sometimes wrong kind of information. Not only feature selection improves the quality of the model but also makes the process of modelling more efficient. It also reduces the complexity of the model and makes it easier to interpret.

For the given data, we have used correlation plot for continuous variables and chi-square test for categorical variables. Following are the code fragments for plotting a correlogram.

```
corrgram(TrainData[,NumericVariables_Index], order = FALSE, upper.panel = panel.pie, lower.panel = panel.shade,
         diag.panel = panel.minmax ,text.panel = panel.txt, main = "Correlation Plot")
```

The correlogram obtained for the data is the following.



From the above graph, we can deduce the following.

- total.day.charge is highly positively correlated with total.day.minutes.
- total.eve.charge is highly positively correlated with total.eve.minutes.
- total.night.charge is highly positively correlated with total.night.minutes.
- total.intl.charge is highly positively correlated with total.intl.minutes.

Since, the above variables are highly positively correlated with the ones mentioned, we can remove the same from the dataset.

Since there are some categorical variables as well, we need went forward with the Chi-Square test of Independence for the same. Below is the attached code fragment for the same.

```
FactorVaiables_Index = sapply(TrainData,is.factor)
FactorVariables = TrainData[,FactorVaiables_Index]
names(FactorVariables)
for (i in 1:2)
{
  print(names(FactorVariables)[i])
  print(chisq.test(table(FactorVariables$Churn,FactorVariables[,i])))
}
```

Following is the result obtained from the test.

```
[1] "international.plan"

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(FactorVariables$Churn, FactorVariables[, i])
X-squared = 222.57, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

        Pearson's Chi-squared test with Yates' continuity correction

data:  table(FactorVariables$Churn, FactorVariables[, i])
X-squared = 34.132, df = 1, p-value = 5.151e-09
```

We know that in case of Chi-Squared Test for Independence, the NULL hypothesis is that the 2 variables, one of which is the target variable, are independent. And the alternate hypothesis is that the 2 variables are not independent .i.e. change in one affects the other. So basically, if the p-values is greater than 0.05, then it shows that the predictor variable is independent of target variable .i.e. the target variable has no dependency on the predictor variable and the predictor variable does not give any information about the target variable. Hence, the NULL hypothesis is accepted. And if the p-variable is greater than 0.05, we accept the alternative hypothesis which is, both the variables are dependent on each other and the predictor gives some information about the target variable.

Form the result obtained, it is seen that none of the variables' p-value is greater than 0.05. So, it is not required to eliminate any of the categorical variables available.

So, after correlation analysis and chi-square test of independence, it is concluded that the following variables can be removed from the dataset.

- total.day.minutes
- total.eve.minutes
- total.night.minutes
- total.intl.minutes.

2.1.4 Feature Scaling

As we can see that the range of the variables in the data thus obtained after dimension reduction is very high. Such huge range might cause anomalies in the model. So, in order to deal with this, we need to scale the variables in a proper range. This is when feature scaling comes into play. Feature scaling is a method to limit the range of variables so that they can be compared on a common ground.

In the dataset that we have, since the data is not normally distributed, we go for normalisation. Normalisation is a technique which changes the values of numeric variables in a dataset to use a common scale, without distorting the differences in the ranges of values or losing information. So, for our dataset we have converted the values of the variables within the range of 0-1.

Following is the code fragment for the same.

```
colNames = c("account.length","number.vmail.messages","total.day.calls","total.day.charge",
             "total.eve.calls","total.eve.charge","total.night.calls","total.night.charge","total.intl.calls",
             "total.intl.charge","number.customer.service.calls")
for(i in colNames)
{
  print(i)
  TrainData_Deleted[,i] = (TrainData_Deleted[,i]-min(TrainData_Deleted[,i]))/
    (max(TrainData[,i])-min(TrainData_Deleted[,i]))
}
```

2.2 Modeling

2.2.1 Model Selection

From the problem statement, we came to know that it is a classification problem since there are only 2 categories of target variables. Either the customer will churn out or not. From the category of the variable, we could make out that it is a nominal variable. So, it makes sense if we proceed with classification models for this.

2.2.2 Decision Trees

It is a predictive model based on the branching series of Boolean tests. It is a type of supervised machine learning algorithm. Its general motive is to create a training model which can be used to predict the class of the target variable by learning the distinct roots inferred from the historical data. In here, there are 3 metrics which help us to extract rules which are statistically significant. Those are:

- Support: It tells us how frequently the item appears in the database.
- Confidence: It is an indication of how often the rule has been found to be true.
- Lift: it is the ratio of observed support to that expected if the variables in question were independent.

For the data that is provided, we implemented the C5.0 model of decision tree. From the decision tree on train data, we could extract 21 odd rules of which the maximum value of lift was found to be 6.8. This model was then fed to the test data.

Following is the code fragment for the same.

```
DecisionTree_C50 = C5.0(Churn ~.,TrainData_Deleted,trails = 500, rules = TRUE)
summary(DecisionTree_C50)
Test_Prediction = predict(DecisionTree_C50, TestData_Deleted[,-16], type = "class")
```

Following are the rules with highest lift value.

```
Rule 9: (60, lift 6.8)
        international.plan = 2
        total.intl.calls <= 0.1
        ->  class 1  [0.984]

Rule 10: (57, lift 6.8)
        international.plan = 2
        total.intl.charge > 0.65
        ->  class 1  [0.983]
```

For evaluating the performance of this model, we take the help of error metrics. Error metrics basically help us to know the evaluate the analytical model and also to know the recommendations and the business point of view. Since the problem is a classification model, we go for confusion matrix to evaluate its performance.

The confusion matrix thus obtained is the following.

```
  Test_Prediction
      0    1
0 1316  127
1   76  148
```

From the confusion matrix, it was found that the accuracy of the model was 87.82% and that the False Negative Rate (FNR) is 33.92%

2.2.3 Logistic Regression

Since our target variable is a categorical variable, we can use logistic regression as well. Output of logistic regression can either be class or probability. So, it will calculate the target class in terms of probability and the we need to assign a threshold value and divide the probabilities into 2 halves.

Before applying logistic regression model on the data, we must keep in mind the following assumptions of logistic regression.

- Ratio of cases to variables: It states that using discrete variables requires that there are enough responses in the given category. Suppose we have a target variable with 2 classes, the logistic regression assumes that the ratio of the two classes should be balanced.
- Absence of multi-collinearity.
- There should not be any outliers.

- The error should be independent of each other.

For the data provided, the logistic regression was performed. Following is the code fragment for the same.

```
LogisticModel = glm(Churn ~., data = TrainData_Deleted, family = "binomial")
summary(LogisticModel)
Test_Prediction_LogRegressn = predict(LogisticModel, newdata = TestData_Deleted, type = "response")
Test_Prediction_LogRegressn = ifelse(Test_Prediction_LogRegressn > 0.5,1,0)
```

For performance evaluation, we take the help of confusion matrix in here as well. The confusion matrix thus obtained is the following.

```
   Test_Prediction_LogRegressn
       0    1
  0 1383   60
  1  168   56
```

From the confusion matrix, it was seen that the accuracy of the model was 86.32% and its False Negative rate is 77%.

2.4 Random Forest

The basic idea of random forest is to have n number of trees to have more accuracy on the dataset. It is basically an ensemble that contains a huge number of decision tress. For a dataset consisting of huge data, it is not possible to build a model on the basis of a single decision tree. In such cases, the ensemble model helps in looking the data in a multi-dimensional way and extract as much variance as it can. In here, higher the number of trees, higher the number of results and more is the number of options to select the best possible result.

For the dataset available, we built a total of 500 decision trees to make a random forest and fed the training model to the test data. Following is the code fragment for the same.

```
RandForest_model = randomForest(Churn ~.,TrainData_Deleted, impportance = TRUE, ntree = 500)
List = RF2List(RandForest_model)
Rules = extractRules(List, TrainData_Deleted[,-16])
readableRules = presentRules(Rules, colnames(TrainData_Deleted))
Metrics = getRuleMetric(Rules, TrainData_Deleted[,-16], TrainData_Deleted$Churn)
Test_Prediction_RandForest = predict(RandForest_model,TestData_Deleted[,-16])
```

For performance evaluation, we take the help of confusion matrix. The confusion matrix for random forest is the following.

```
   Test_Prediction_RandForest
       0    1
  0 1343  100
  1   66  158
```

From the confusion matrix, the accuracy was found to be 90.04% and the False Negative rate was found to be 30.8%.

Chapter 3

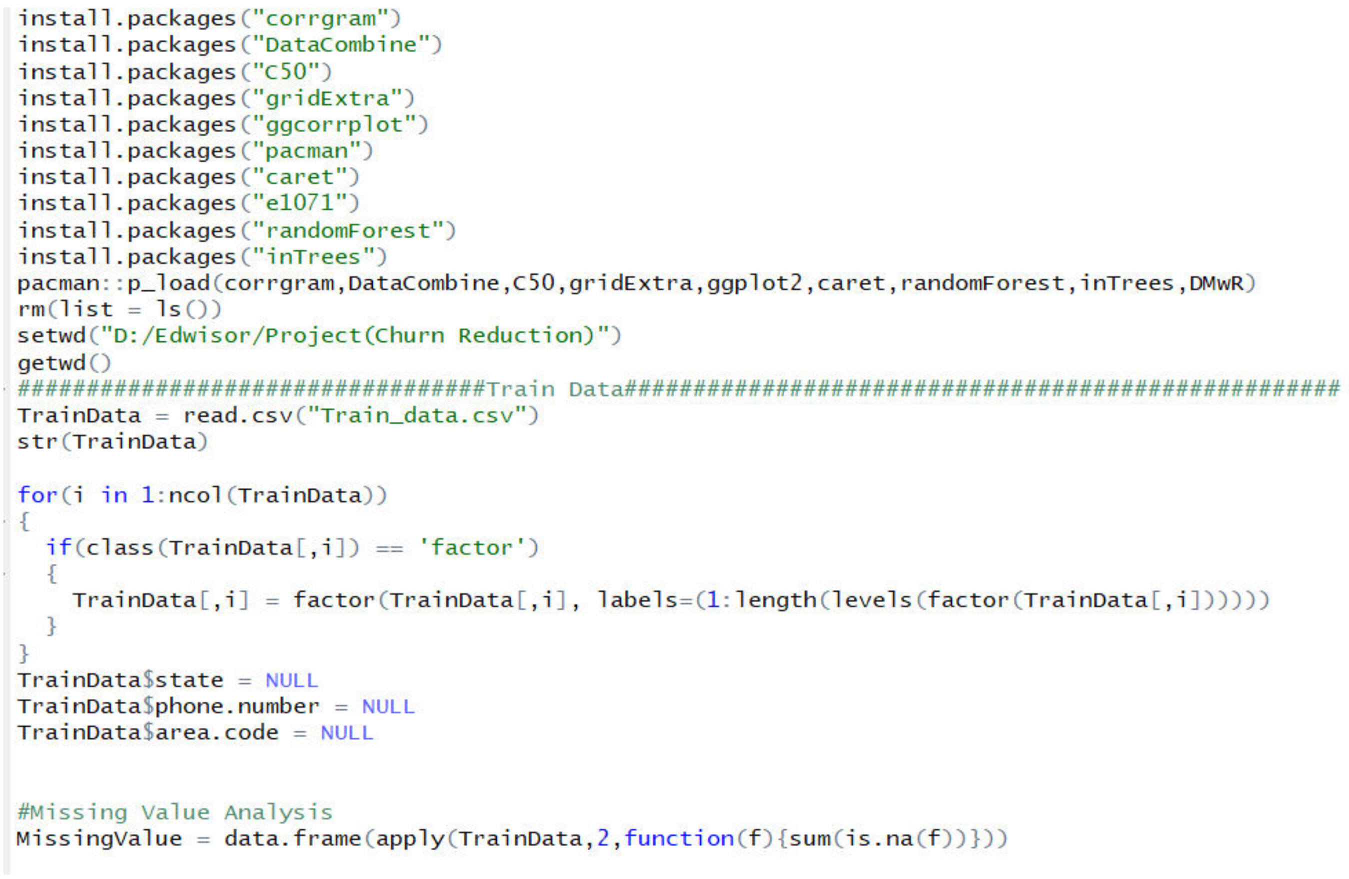
Conclusion

Now that we have a few models for predicting the target variable, we need to decide which one to choose. Logically, we need to select a model which has an accuracy rate of more than 80% and the least False Negative Rate. If the false negative rate is high, it will have an adverse effect on the business.

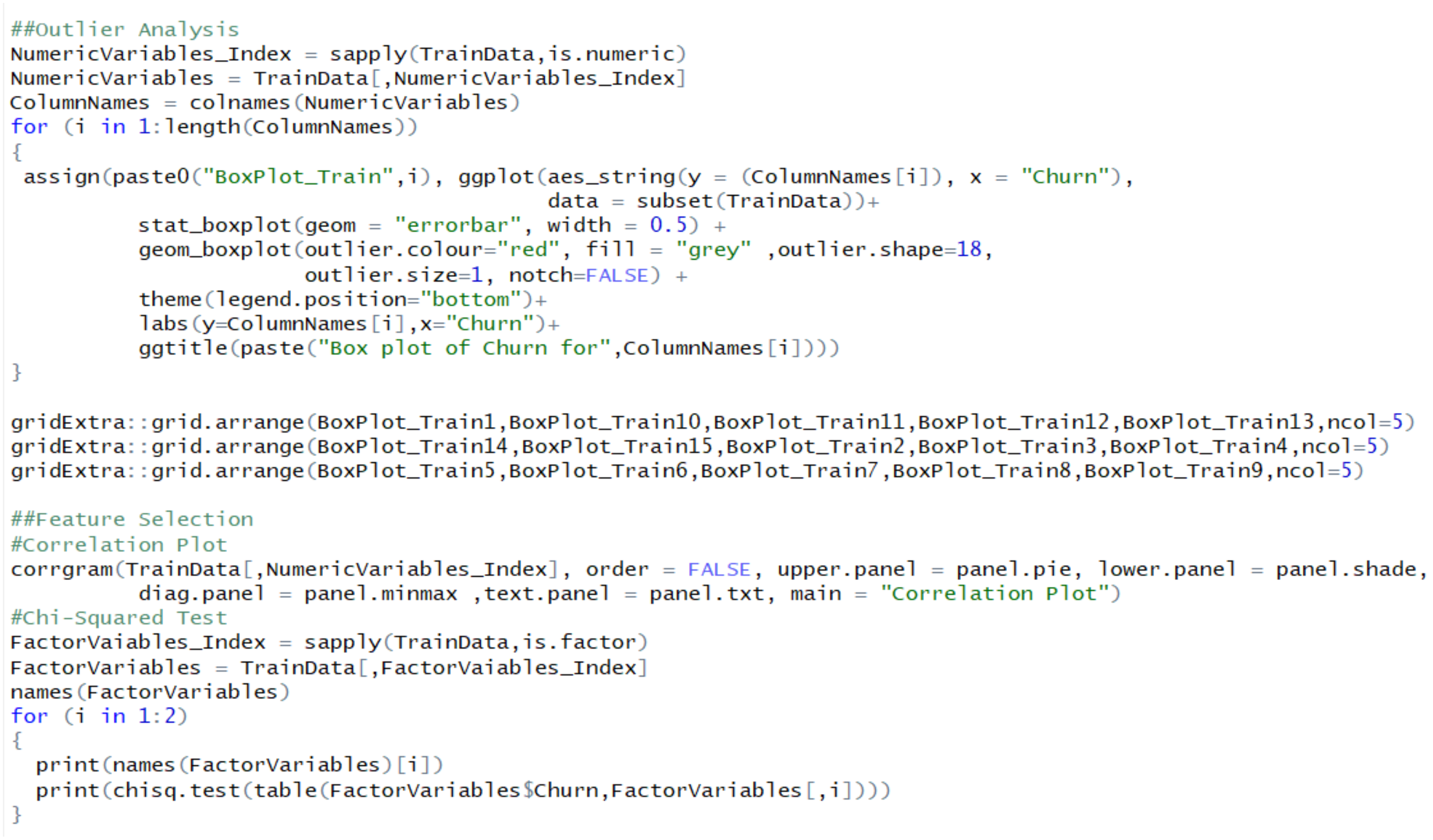
From the performance evaluation of all the models, we can conclude that we can select Random Forest as the final model.

Appendix A – R Code

Complete R File.

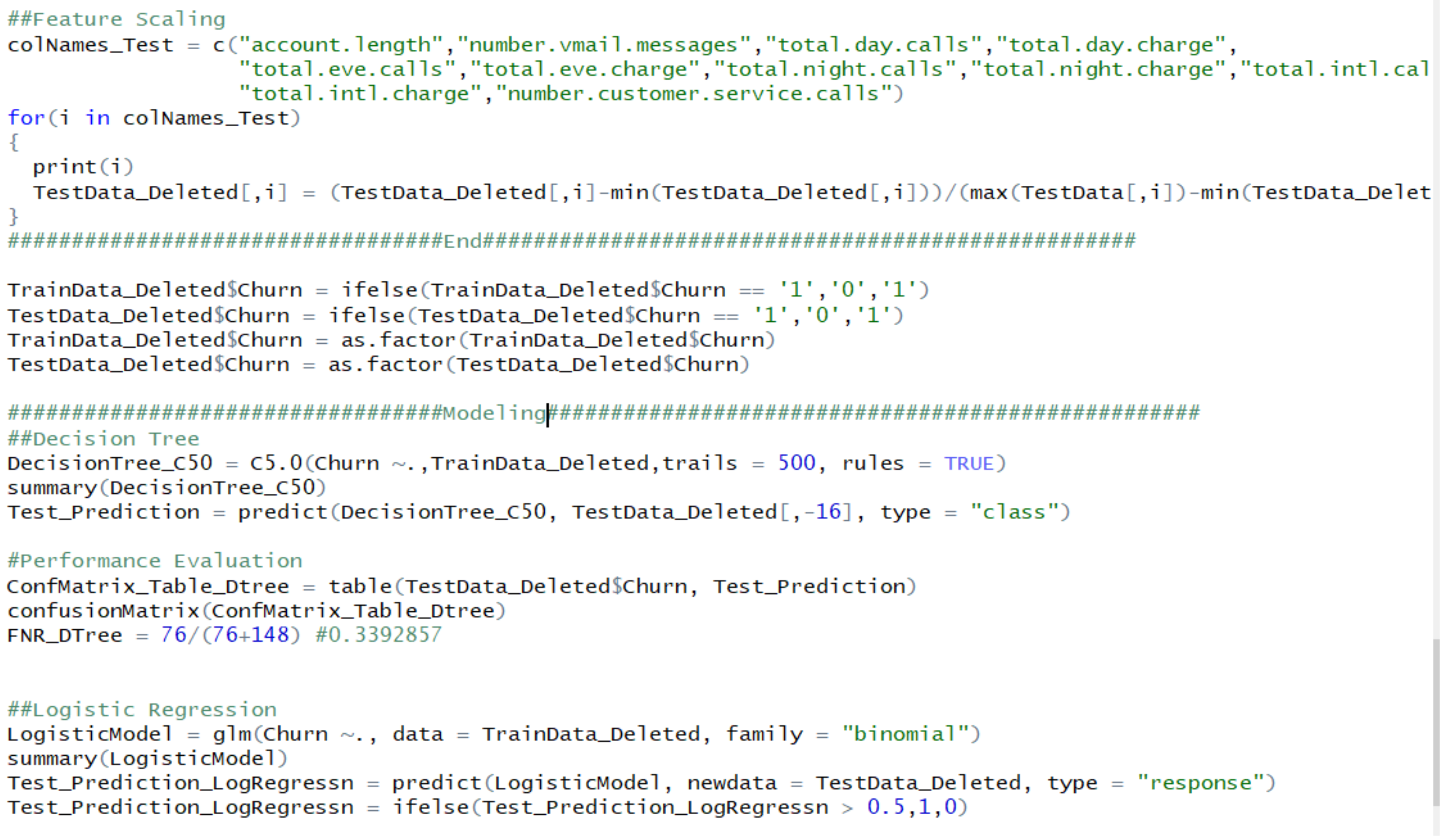


```r
install.packages("corrgram")
install.packages("DataCombine")
install.packages("C50")
install.packages("gridExtra")
install.packages("ggcorrplot")
install.packages("pacman")
install.packages("caret")
install.packages("e1071")
install.packages("randomForest")
install.packages("inTrees")
pacman::p_load(corrgram,DataCombine,C50,gridExtra,ggplot2,caret,randomForest,inTrees,DMwR)
rm(list = ls())
setwd("D:/Edwisor/Project(Churn Reduction)")
getwd()
##################################Train Data#####################################################
TrainData = read.csv("Train_data.csv")
str(TrainData)

for(i in 1:ncol(TrainData))
{
  if(class(TrainData[,i]) == 'factor')
  {
    TrainData[,i] = factor(TrainData[,i], labels=(1:length(levels(factor(TrainData[,i])))))
  }
}
TrainData$state = NULL
TrainData$phone.number = NULL
TrainData$area.code = NULL


#Missing Value Analysis
MissingValue = data.frame(apply(TrainData,2,function(f){sum(is.na(f))}))
```

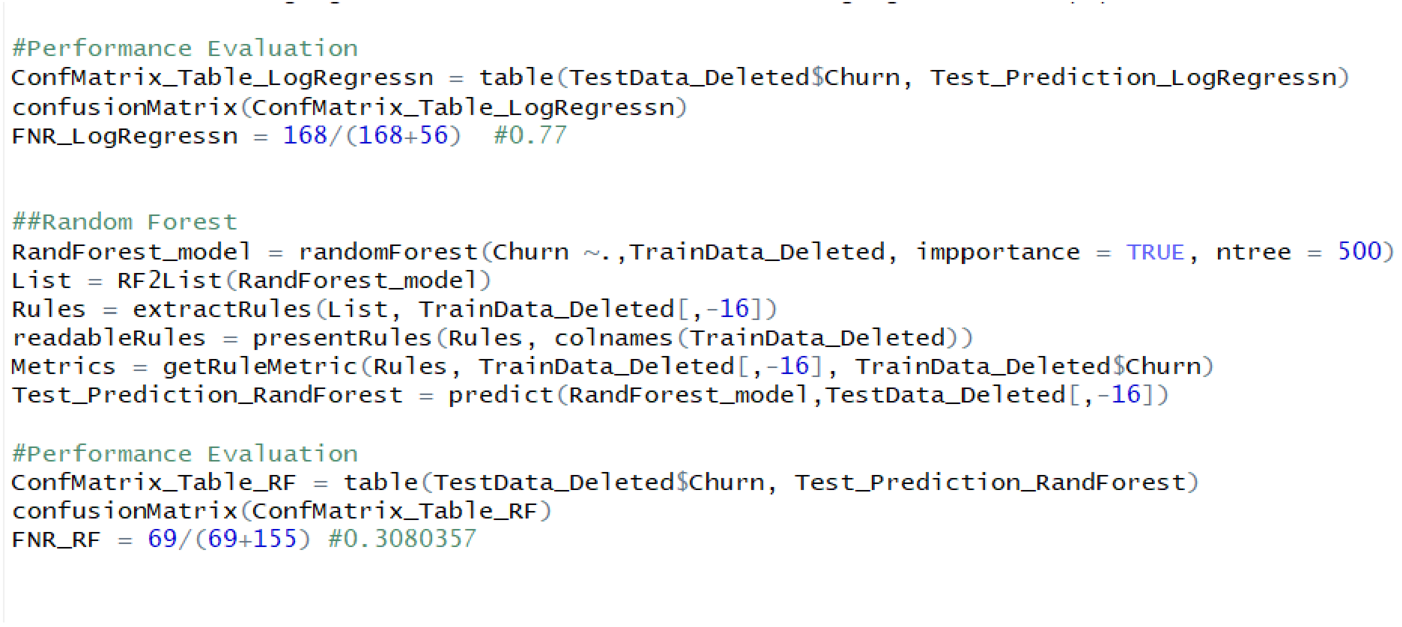


```r
##Outlier Analysis
NumericVariables_Index = sapply(TrainData,is.numeric)
NumericVariables = TrainData[,NumericVariables_Index]
ColumnNames = colnames(NumericVariables)
for (i in 1:length(ColumnNames))
{
 assign(paste0("BoxPlot_Train",i), ggplot(aes_string(y = (ColumnNames[i]), x = "Churn"),
                                         data = subset(TrainData))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                       outlier.size=1, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(y=ColumnNames[i],x="Churn")+
          ggtitle(paste("Box plot of Churn for",ColumnNames[i])))
}

gridExtra::grid.arrange(BoxPlot_Train1,BoxPlot_Train10,BoxPlot_Train11,BoxPlot_Train12,BoxPlot_Train13,ncol=5)
gridExtra::grid.arrange(BoxPlot_Train14,BoxPlot_Train15,BoxPlot_Train2,BoxPlot_Train3,BoxPlot_Train4,ncol=5)
gridExtra::grid.arrange(BoxPlot_Train5,BoxPlot_Train6,BoxPlot_Train7,BoxPlot_Train8,BoxPlot_Train9,ncol=5)

##Feature Selection
#Correlation Plot
corrgram(TrainData[,NumericVariables_Index], order = FALSE, upper.panel = panel.pie, lower.panel = panel.shade,
         diag.panel = panel.minmax ,text.panel = panel.txt, main = "Correlation Plot")
#Chi-Squared Test
FactorVaiables_Index = sapply(TrainData,is.factor)
FactorVariables = TrainData[,FactorVaiables_Index]
names(FactorVariables)
for (i in 1:2)
{
  print(names(FactorVariables)[i])
  print(chisq.test(table(FactorVariables$Churn,FactorVariables[,i])))
}
```

```r
#Dimension Reduction
TrainData_Deleted = subset(TrainData,select = -c(total.day.minutes, total.night.minutes, total.eve.minutes,
                                        total.intl.minutes))

##Feature Scaling
colNames = c("account.length","number.vmail.messages","total.day.calls","total.day.charge",
             "total.eve.calls","total.eve.charge","total.night.calls","total.night.charge","total.intl.calls",
             "total.intl.charge","number.customer.service.calls")
for(i in colNames)
{
  print(i)
  TrainData_Deleted[,i] = (TrainData_Deleted[,i]-min(TrainData_Deleted[,i]))/
    (max(TrainData[,i])-min(TrainData_Deleted[,i]))
}

##################################End#####################################################

##################################Test Data#####################################################
TestData = read.csv("Test_data.csv")
str(TestData)

TestData$state = NULL
TestData$phone.number = NULL
TestData$area.code = NULL

for(i in 1:ncol(TestData))
{
  if(class(TestData[,i]) == 'factor')
  {
    TestData[,i] = factor(TestData[,i], labels=(1:length(levels(factor(TestData[,i])))))
  }
}

#Missing Value Analysis
MissingValue_Test = data.frame(apply(TestData,2,function(f){sum(is.na(f))}))


##Outlier Analysis
NumericVariables_Index_Test = sapply(TestData,is.numeric)
NumericVariables_Test = TestData[,NumericVariables_Index_Test]
ColumnNames_Test = colnames(NumericVariables_Test)
for (i in 1:length(ColumnNames_Test))
{
  assign(paste0("BoxPlot_Test",i), ggplot(aes_string(y = (ColumnNames_Test[i]), x = "Churn"), data = subset(Tes
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=ColumnNames_Test[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",ColumnNames_Test[i])))
}

gridExtra::grid.arrange(BoxPlot_Test1,BoxPlot_Test10,BoxPlot_Test11,BoxPlot_Test12,BoxPlot_Test13,ncol=5)
gridExtra::grid.arrange(BoxPlot_Test14,BoxPlot_Test15,BoxPlot_Test2,BoxPlot_Test3,BoxPlot_Test4,ncol=5)
gridExtra::grid.arrange(BoxPlot_Test5,BoxPlot_Test6,BoxPlot_Test7,BoxPlot_Test8,BoxPlot_Test9,ncol=5)

##Feature Selection
#Correlation Plot
corrgram(TestData[,NumericVariables_Index_Test], order = FALSE, upper.panel = panel.pie, lower.panel = panel.sh
         diag.panel = panel.minmax ,text.panel = panel.txt, main = "Correlation Plot")
#Chi-Squared Test
FactorVaiables_Index_Test = sapply(TestData,is.factor)
FactorVariables_Test = TestData[,FactorVaiables_Index_Test]
names(FactorVariables_Test)

for (i in 1:2)
{
  print(names(FactorVariables_Test)[i])
  print(chisq.test(table(FactorVariables_Test$Churn,FactorVariables_Test[,i])))
}

#Dimension Reduction
TestData_Deleted = subset(TestData,select = -c(total.day.minutes, total.night.minutes, total.eve.minutes,
                                               total.intl.minutes))
```

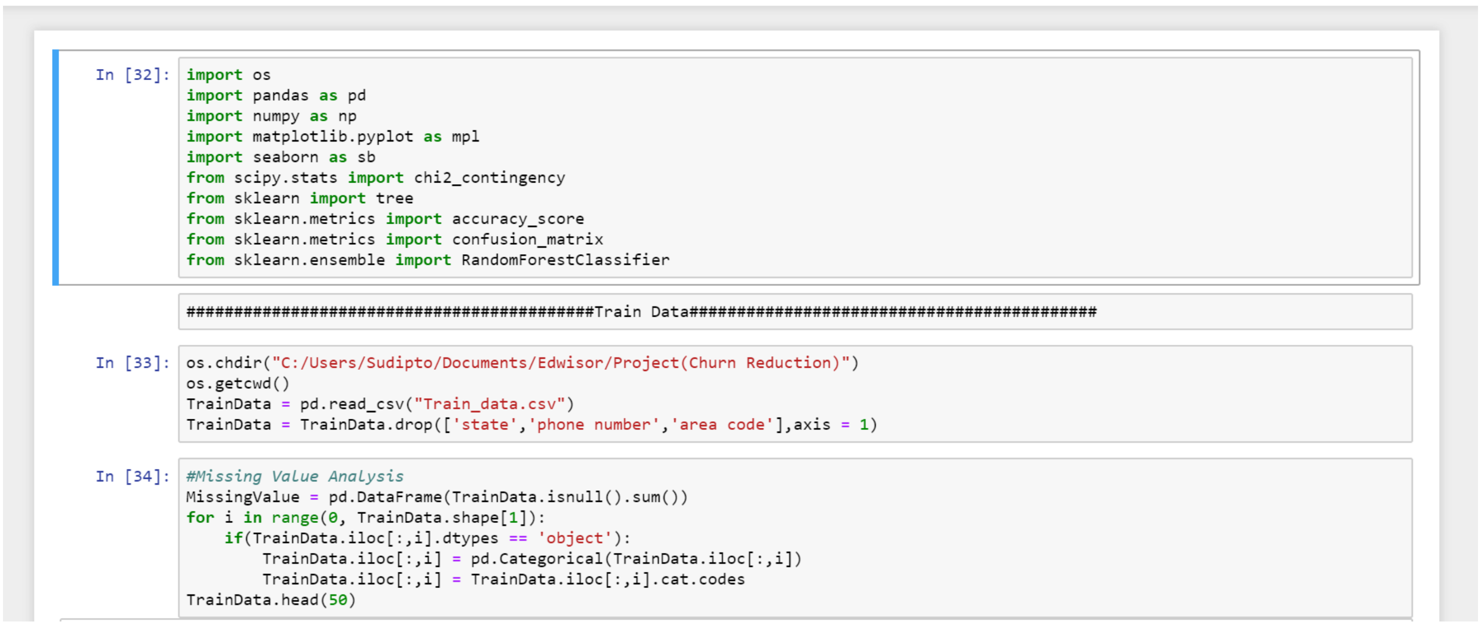


```
##Feature Scaling
colNames_Test = c("account.length","number.vmail.messages","total.day.calls","total.day.charge",
                  "total.eve.calls","total.eve.charge","total.night.calls","total.night.charge","total.intl.cal
                  "total.intl.charge","number.customer.service.calls")
for(i in colNames_Test)
{
  print(i)
  TestData_Deleted[,i] = (TestData_Deleted[,i]-min(TestData_Deleted[,i]))/(max(TestData[,i])-min(TestData_Delet
}
##################################End################################################################

TrainData_Deleted$Churn = ifelse(TrainData_Deleted$Churn == '1','0','1')
TestData_Deleted$Churn = ifelse(TestData_Deleted$Churn == '1','0','1')
TrainData_Deleted$Churn = as.factor(TrainData_Deleted$Churn)
TestData_Deleted$Churn = as.factor(TestData_Deleted$Churn)

##################################Modeling####################################################
##Decision Tree
DecisionTree_C50 = C5.0(Churn ~.,TrainData_Deleted,trails = 500, rules = TRUE)
summary(DecisionTree_C50)
Test_Prediction = predict(DecisionTree_C50, TestData_Deleted[,-16], type = "class")

#Performance Evaluation
ConfMatrix_Table_Dtree = table(TestData_Deleted$Churn, Test_Prediction)
confusionMatrix(ConfMatrix_Table_Dtree)
FNR_DTree = 76/(76+148) #0.3392857


##Logistic Regression
LogisticModel = glm(Churn ~., data = TrainData_Deleted, family = "binomial")
summary(LogisticModel)
Test_Prediction_LogRegressn = predict(LogisticModel, newdata = TestData_Deleted, type = "response")
Test_Prediction_LogRegressn = ifelse(Test_Prediction_LogRegressn > 0.5,1,0)
```

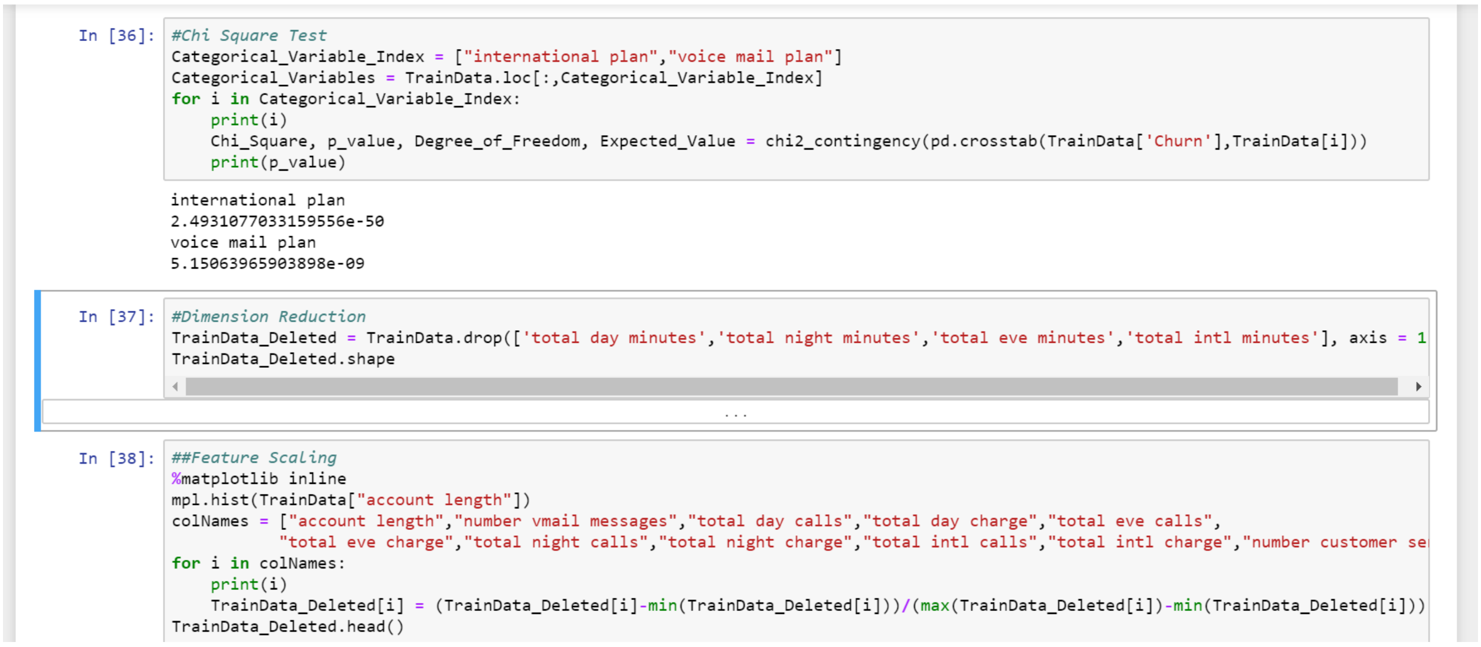


```
#Performance Evaluation
ConfMatrix_Table_LogRegressn = table(TestData_Deleted$Churn, Test_Prediction_LogRegressn)
confusionMatrix(ConfMatrix_Table_LogRegressn)
FNR_LogRegressn = 168/(168+56)   #0.77


##Random Forest
RandForest_model = randomForest(Churn ~.,TrainData_Deleted, impportance = TRUE, ntree = 500)
List = RF2List(RandForest_model)
Rules = extractRules(List, TrainData_Deleted[,-16])
readableRules = presentRules(Rules, colnames(TrainData_Deleted))
Metrics = getRuleMetric(Rules, TrainData_Deleted[,-16], TrainData_Deleted$Churn)
Test_Prediction_RandForest = predict(RandForest_model,TestData_Deleted[,-16])

#Performance Evaluation
ConfMatrix_Table_RF = table(TestData_Deleted$Churn, Test_Prediction_RandForest)
confusionMatrix(ConfMatrix_Table_RF)
FNR_RF = 69/(69+155) #0.3080357
```

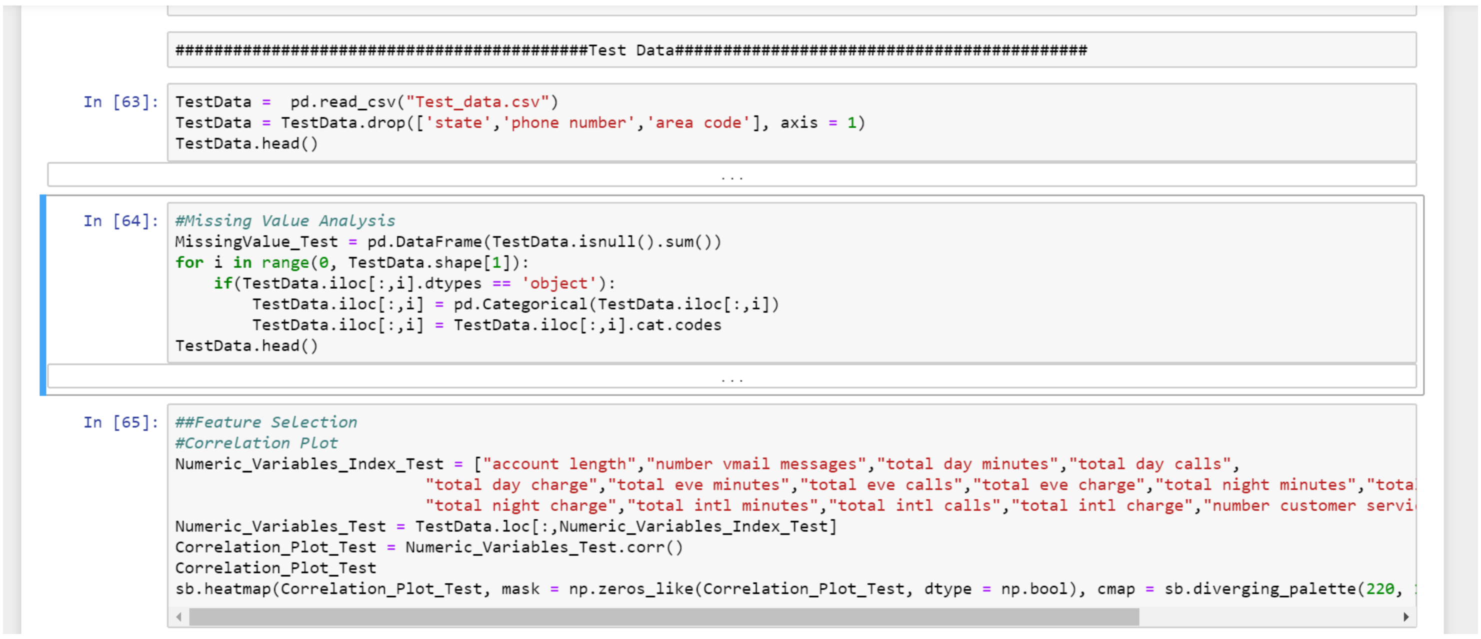
Appendix A – Python Code



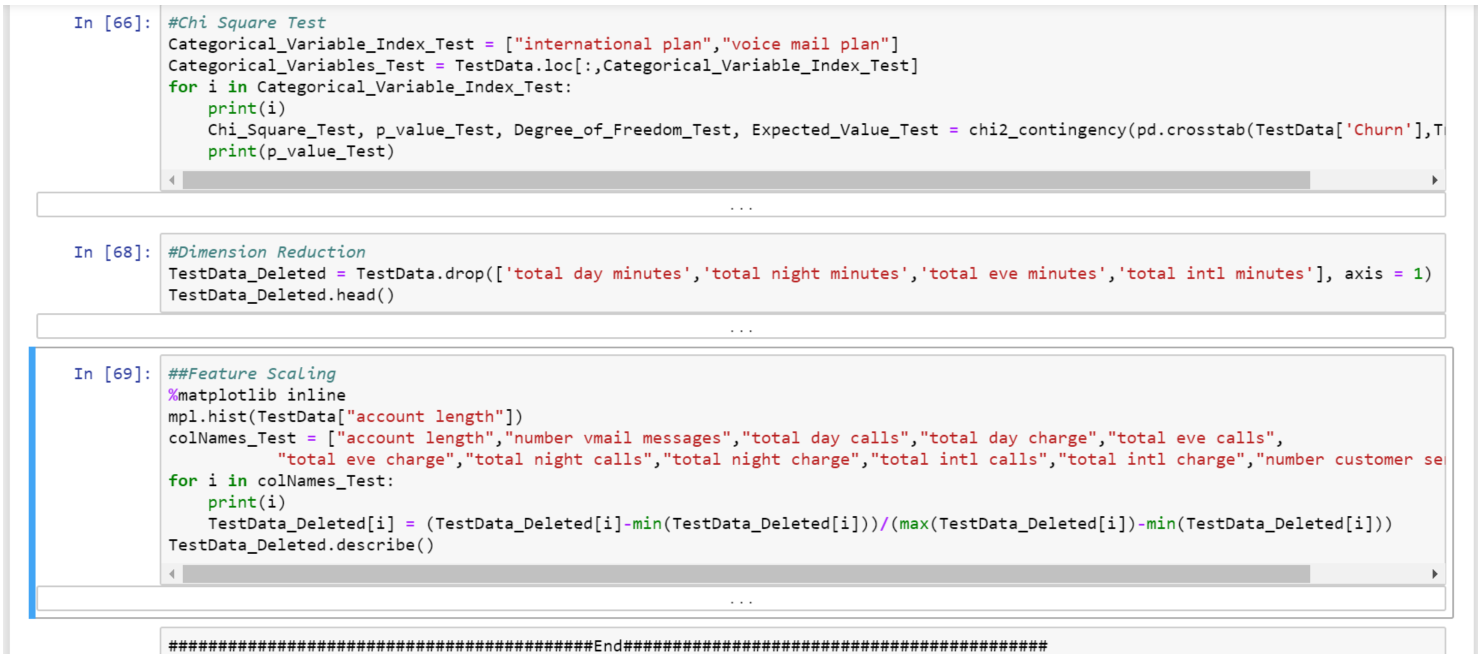
```
In [32]: import os
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as mpl
         import seaborn as sb
         from scipy.stats import chi2_contingency
         from sklearn import tree
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import confusion_matrix
         from sklearn.ensemble import RandomForestClassifier

         ###########################################Train Data##########################################

In [33]: os.chdir("C:/Users/Sudipto/Documents/Edwisor/Project(Churn Reduction)")
         os.getcwd()
         TrainData = pd.read_csv("Train_data.csv")
         TrainData = TrainData.drop(['state','phone number','area code'],axis = 1)

In [34]: #Missing Value Analysis
         MissingValue = pd.DataFrame(TrainData.isnull().sum())
         for i in range(0, TrainData.shape[1]):
             if(TrainData.iloc[:,i].dtypes == 'object'):
                 TrainData.iloc[:,i] = pd.Categorical(TrainData.iloc[:,i])
                 TrainData.iloc[:,i] = TrainData.iloc[:,i].cat.codes
         TrainData.head(50)
```



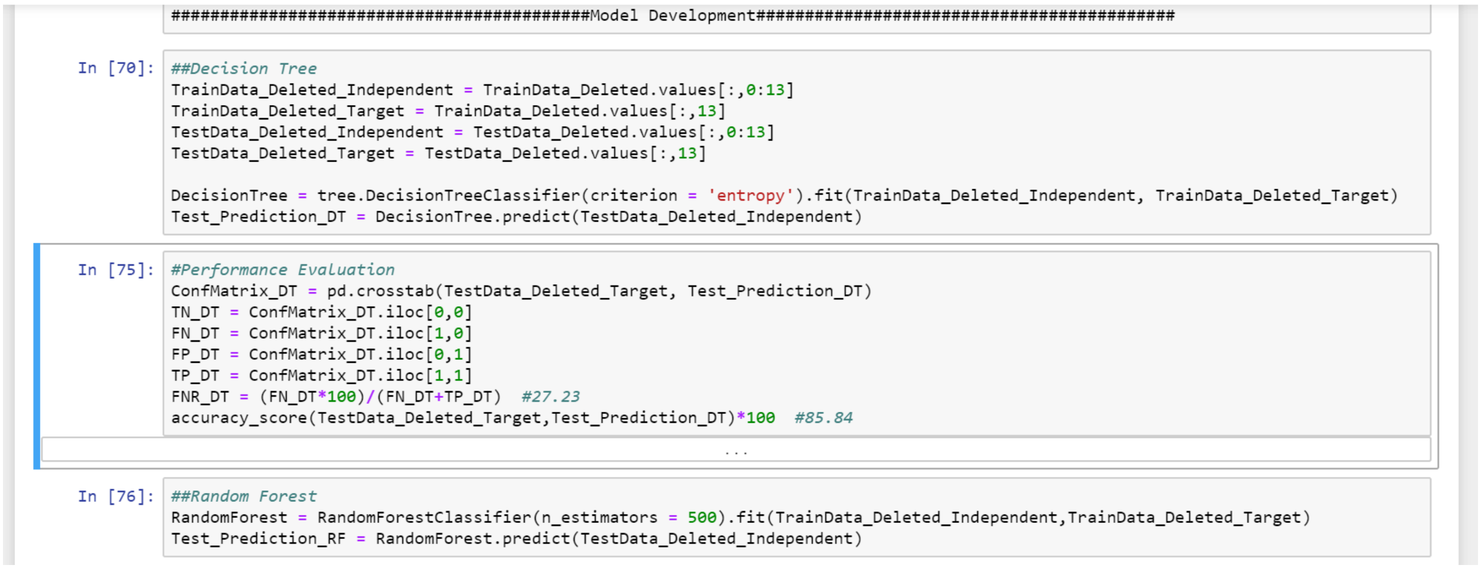
```
In [36]: #Chi Square Test
         Categorical_Variable_Index = ["international plan","voice mail plan"]
         Categorical_Variables = TrainData.loc[:,Categorical_Variable_Index]
         for i in Categorical_Variable_Index:
             print(i)
             Chi_Square, p_value, Degree_of_Freedom, Expected_Value = chi2_contingency(pd.crosstab(TrainData['Churn'],TrainData[i]))
             print(p_value)

         international plan
         2.4931077033159556e-50
         voice mail plan
         5.15063965903898e-09

In [37]: #Dimension Reduction
         TrainData_Deleted = TrainData.drop(['total day minutes','total night minutes','total eve minutes','total intl minutes'], axis = 1
         TrainData_Deleted.shape

                                              ...

In [38]: ##Feature Scaling
         %matplotlib inline
         mpl.hist(TrainData["account length"])
         colNames = ["account length","number vmail messages","total day calls","total day charge","total eve calls",
                     "total eve charge","total night calls","total night charge","total intl calls","total intl charge","number customer se
         for i in colNames:
             print(i)
             TrainData_Deleted[i] = (TrainData_Deleted[i]-min(TrainData_Deleted[i]))/(max(TrainData_Deleted[i])-min(TrainData_Deleted[i]))
         TrainData_Deleted.head()
```



```
##############################################Test Data############################################

In [63]: TestData =  pd.read_csv("Test_data.csv")
         TestData = TestData.drop(['state','phone number','area code'], axis = 1)
         TestData.head()

In [64]: #Missing Value Analysis
         MissingValue_Test = pd.DataFrame(TestData.isnull().sum())
         for i in range(0, TestData.shape[1]):
             if(TestData.iloc[:,i].dtypes == 'object'):
                 TestData.iloc[:,i] = pd.Categorical(TestData.iloc[:,i])
                 TestData.iloc[:,i] = TestData.iloc[:,i].cat.codes
         TestData.head()

In [65]: ##Feature Selection
         #Correlation Plot
         Numeric_Variables_Index_Test = ["account length","number vmail messages","total day minutes","total day calls",
                                  "total day charge","total eve minutes","total eve calls","total eve charge","total night minutes","tota
                                  "total night charge","total intl minutes","total intl calls","total intl charge","number customer servi
         Numeric_Variables_Test = TestData.loc[:,Numeric_Variables_Index_Test]
         Correlation_Plot_Test = Numeric_Variables_Test.corr()
         Correlation_Plot_Test
         sb.heatmap(Correlation_Plot_Test, mask = np.zeros_like(Correlation_Plot_Test, dtype = np.bool), cmap = sb.diverging_palette(220,
```



```
In [66]: #Chi Square Test
         Categorical_Variable_Index_Test = ["international plan","voice mail plan"]
         Categorical_Variables_Test = TestData.loc[:,Categorical_Variable_Index_Test]
         for i in Categorical_Variable_Index_Test:
             print(i)
             Chi_Square_Test, p_value_Test, Degree_of_Freedom_Test, Expected_Value_Test = chi2_contingency(pd.crosstab(TestData['Churn'],T
             print(p_value_Test)

In [68]: #Dimension Reduction
         TestData_Deleted = TestData.drop(['total day minutes','total night minutes','total eve minutes','total intl minutes'], axis = 1)
         TestData_Deleted.head()

In [69]: ##Feature Scaling
         %matplotlib inline
         mpl.hist(TestData["account length"])
         colNames_Test = ["account length","number vmail messages","total day calls","total day charge","total eve calls",
                    "total eve charge","total night calls","total night charge","total intl calls","total intl charge","number customer se
         for i in colNames_Test:
             print(i)
             TestData_Deleted[i] = (TestData_Deleted[i]-min(TestData_Deleted[i]))/(max(TestData_Deleted[i])-min(TestData_Deleted[i]))
         TestData_Deleted.describe()

################################################End############################################
```



```
##############################################Model Development#############################################

In [70]: ##Decision Tree
         TrainData_Deleted_Independent = TrainData_Deleted.values[:,0:13]
         TrainData_Deleted_Target = TrainData_Deleted.values[:,13]
         TestData_Deleted_Independent = TestData_Deleted.values[:,0:13]
         TestData_Deleted_Target = TestData_Deleted.values[:,13]

         DecisionTree = tree.DecisionTreeClassifier(criterion = 'entropy').fit(TrainData_Deleted_Independent, TrainData_Deleted_Target)
         Test_Prediction_DT = DecisionTree.predict(TestData_Deleted_Independent)

In [75]: #Performance Evaluation
         ConfMatrix_DT = pd.crosstab(TestData_Deleted_Target, Test_Prediction_DT)
         TN_DT = ConfMatrix_DT.iloc[0,0]
         FN_DT = ConfMatrix_DT.iloc[1,0]
         FP_DT = ConfMatrix_DT.iloc[0,1]
         TP_DT = ConfMatrix_DT.iloc[1,1]
         FNR_DT = (FN_DT*100)/(FN_DT+TP_DT)  #27.23
         accuracy_score(TestData_Deleted_Target,Test_Prediction_DT)*100  #85.84

In [76]: ##Random Forest
         RandomForest = RandomForestClassifier(n_estimators = 500).fit(TrainData_Deleted_Independent,TrainData_Deleted_Target)
         Test_Prediction_RF = RandomForest.predict(TestData_Deleted_Independent)
```

```
In [79]: #Performance Evaluation
         ConfMatrix_RF = pd.crosstab(TestData_Deleted_Target,Test_Prediction_RF)
         TN_RF = ConfMatrix_RF.iloc[0,0]
         FN_RF = ConfMatrix_RF.iloc[1,0]
         FP_RF = ConfMatrix_RF.iloc[0,1]
         TP_RF = ConfMatrix_RF.iloc[1,1]
         FNR_RF = (FN_RF*100)/(FN_RF+TP_RF)  #30.80
         accuracy_score(TestData_Deleted_Target,Test_Prediction_RF)*100  #89.86
```

...