Churn Reduction

Diksha P. Jagre

Contents

Chapter 1

Introduction

1.1 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.

1.2 Data

Our task is to build a classification models which will classify the customer behaviour depending on the churn score. Given below is a sample of data set that we are using to predict the customer behaviour:

Data Sets -

1) Test_data.csv 2) Train_data.csv

As you can see in the table below we have the following 17 variables, using which we have to correctly predict the customer behaviour:

Account length
International plan
Voice mail plan
Number of voice mail messages
Total day minutes
Total day calls
Total day charge
Total evening minutes
Total evening calls
Total evening charge
Total night minutes
Total night calls
Total night charge
Total international minutes
Total international calls
Total international charge

Number of customer service calls

Figure 1.1 Table : Predictor Variables

Chapter 2

Methodology

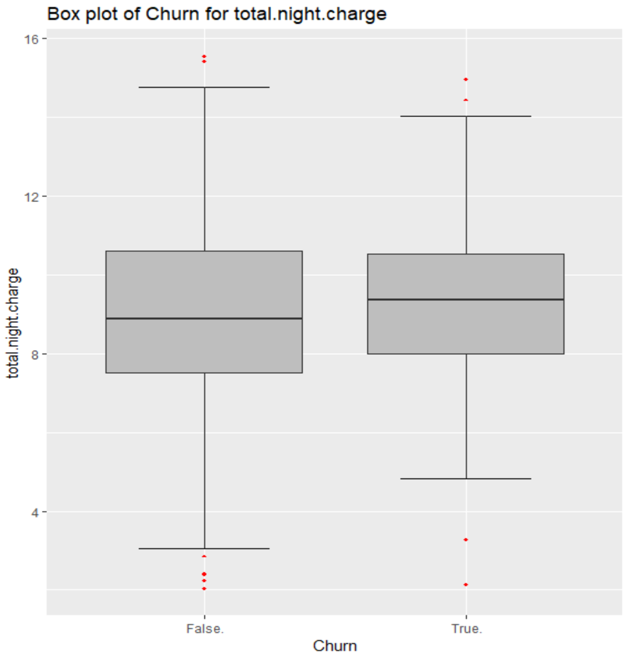
2.1 Pre-Processing

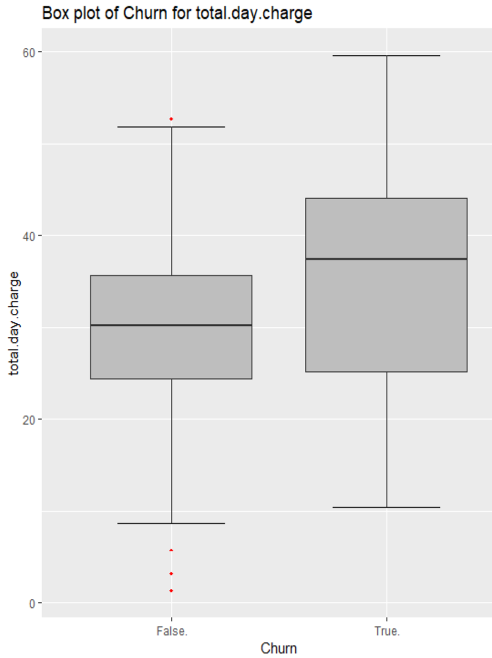
Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

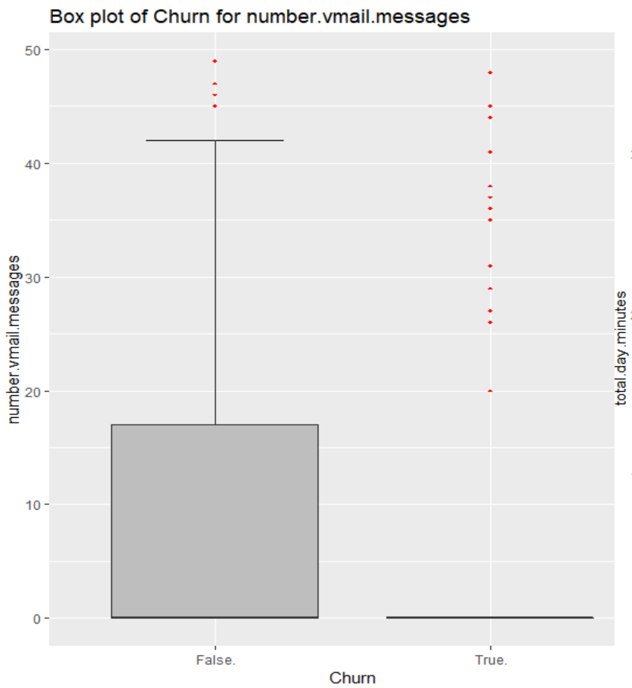
Here the dataset will be already given in probability like in test and train data. So, we will first try to look our train and test dataset for checking how many variables dataset contain and this dataset have balance or imbalance dataset. If the given dataset is balanced then used that dataset for further operations. but if the given dataset is imbalance then first you have to apply sampling technique. Sampling technique is used for equally distributed dataset that is in 70%-30% dataset or 60%-40% dataset. Once we apply sampling technique, dataset will be in proportion like in 70%-30% or 60%-40%.After that we will used that dataset for further operations.

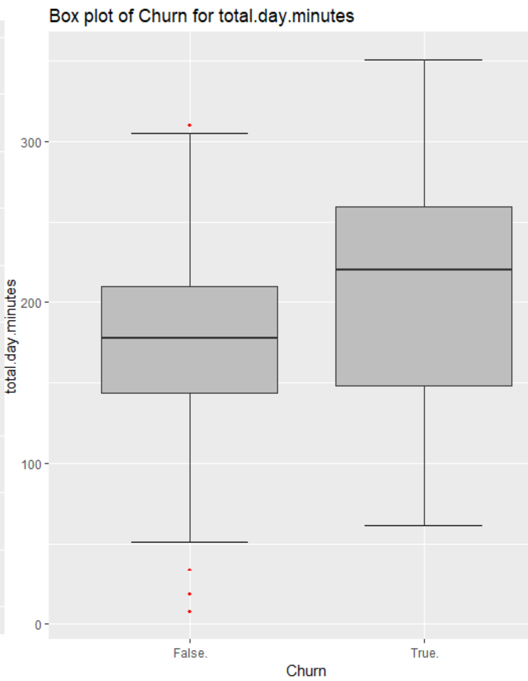
2.1.1 Outlier Analysis

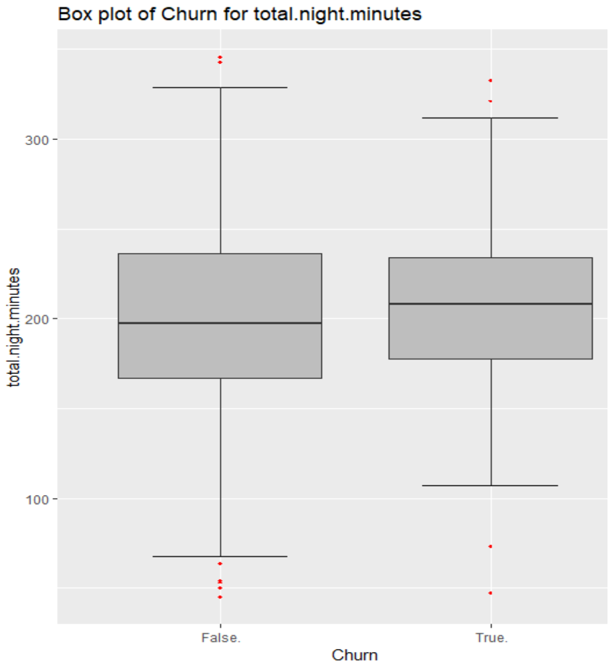
Outlier analysis is one of the technique to understand, clean and prepare data for building a predictive model. We can clearly observe from these probability distributions that most of the variables are skewed. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. We can see the effect of the skew in below figure.

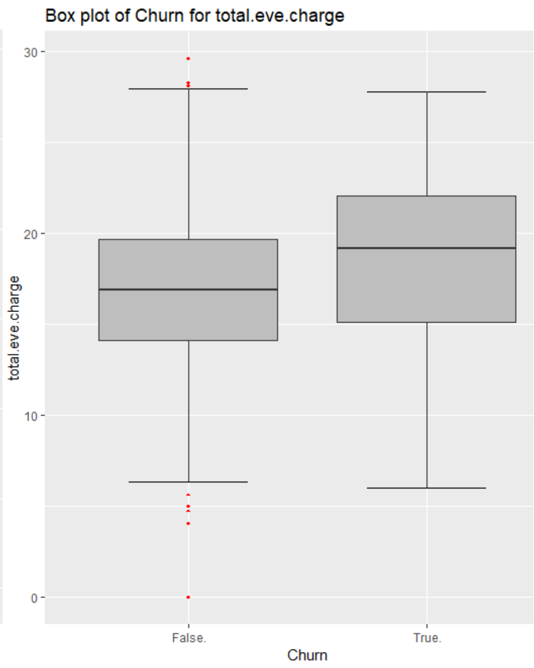

Box plot of Churn for total.night.charge
total.night.charge
16
12
8
4
False.
True.
Churn

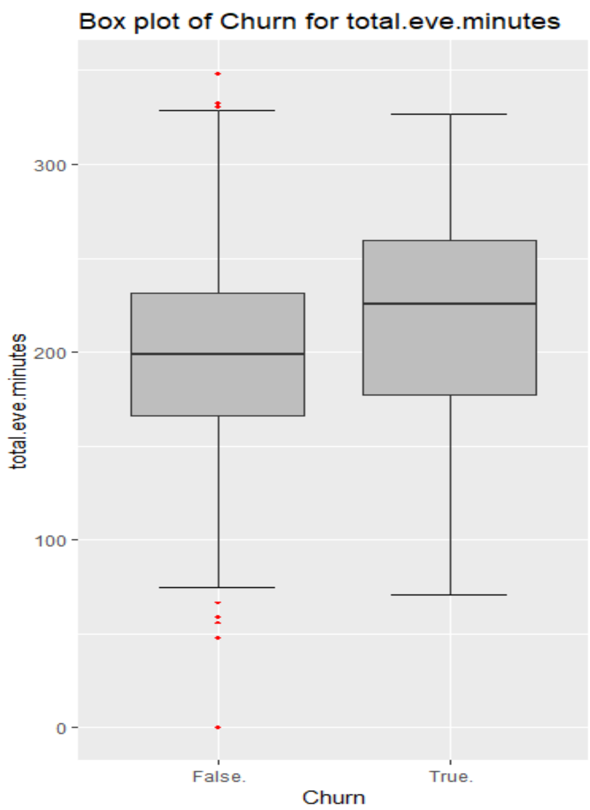

Box plot of Churn for total.day.charge
total.day.charge
60
40
20
0
False.
True.
Churn

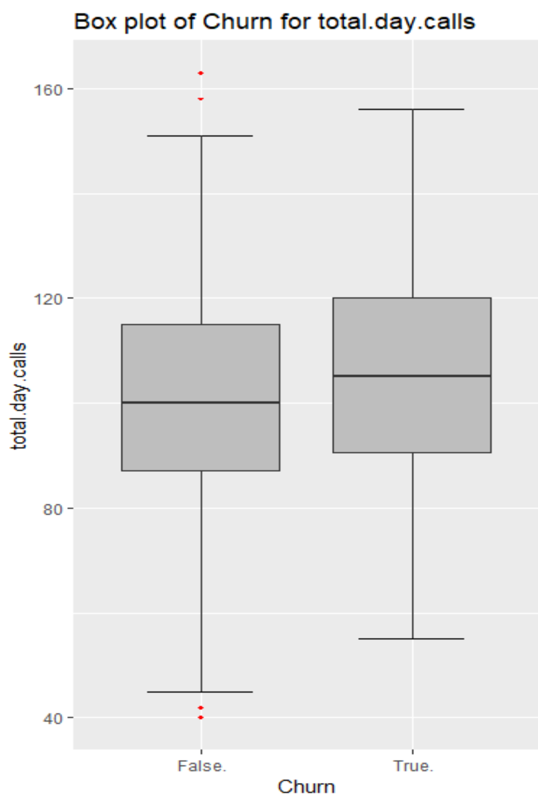

Box plot of Churn for number.vmail.messages
number.vmail.messages
50
40
30
20
10
0
False.
True.
Churn


Box plot of Churn for total.day.minutes
total.day.minutes
300
200
100
0
False.
True.
Churn

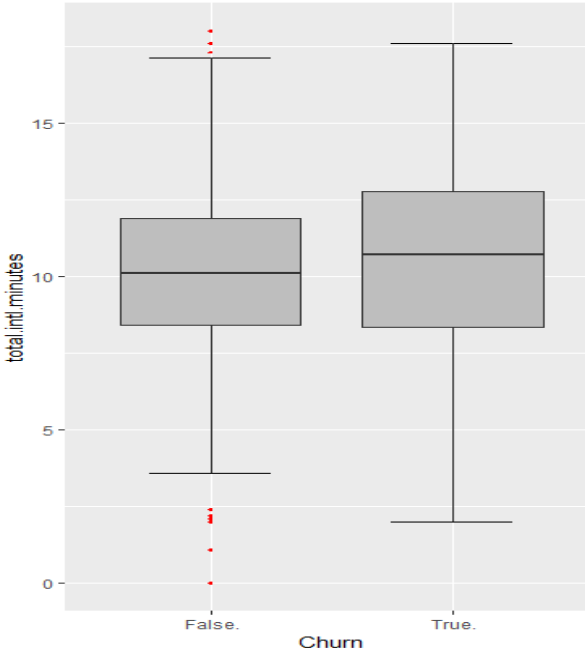

Box plot of Churn for total.night.minutes
300
200
100
total.night.minutes
False.
True.
Churn


Box plot of Churn for total.eve.charge
30
20
10
0
total.eve.charge
False.
True.
Churn

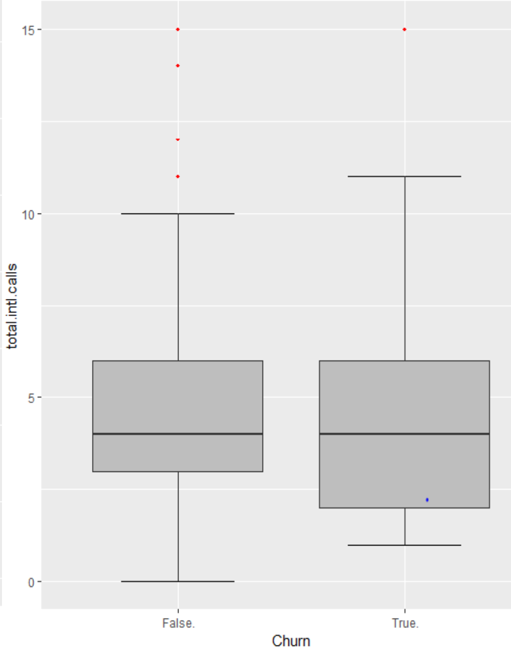

Box plot of Churn for total.eve.minutes
300
200
100
0
total.eve.minutes
False.
True.
Churn


Box plot of Churn for total.day.calls
160
120
80
40
total.day.calls
False.
True.
Churn

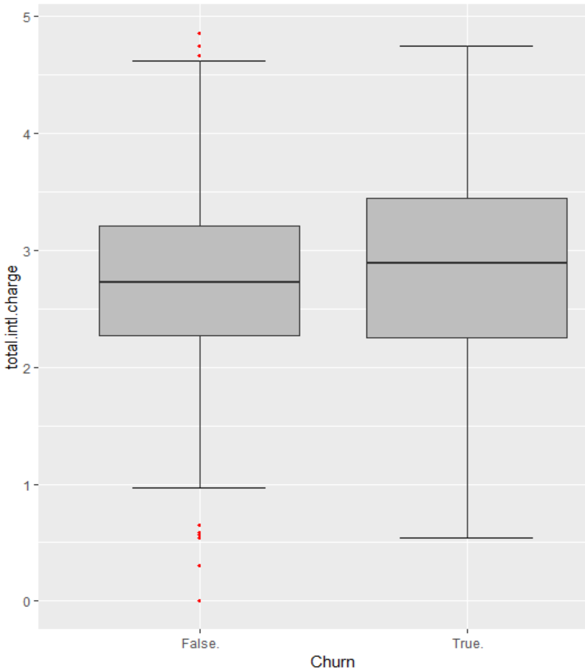
Box plot of Churn for total.intl.minutes



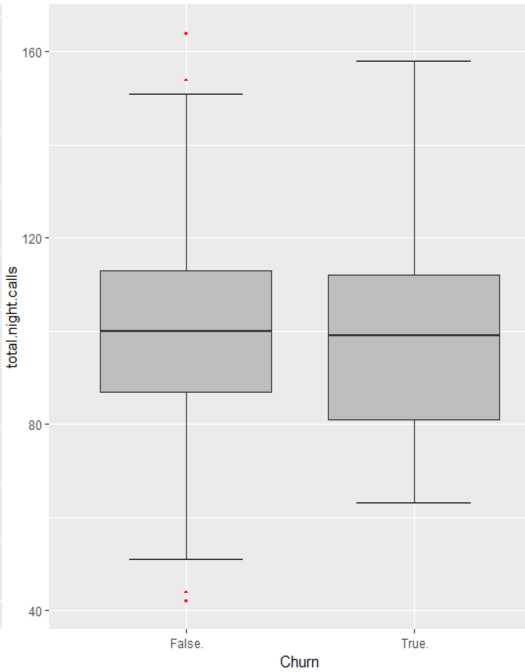
Box plot of Churn for total.intl.calls



Box plot of Churn for total.intl.charge



Box plot of Churn for total.night.calls



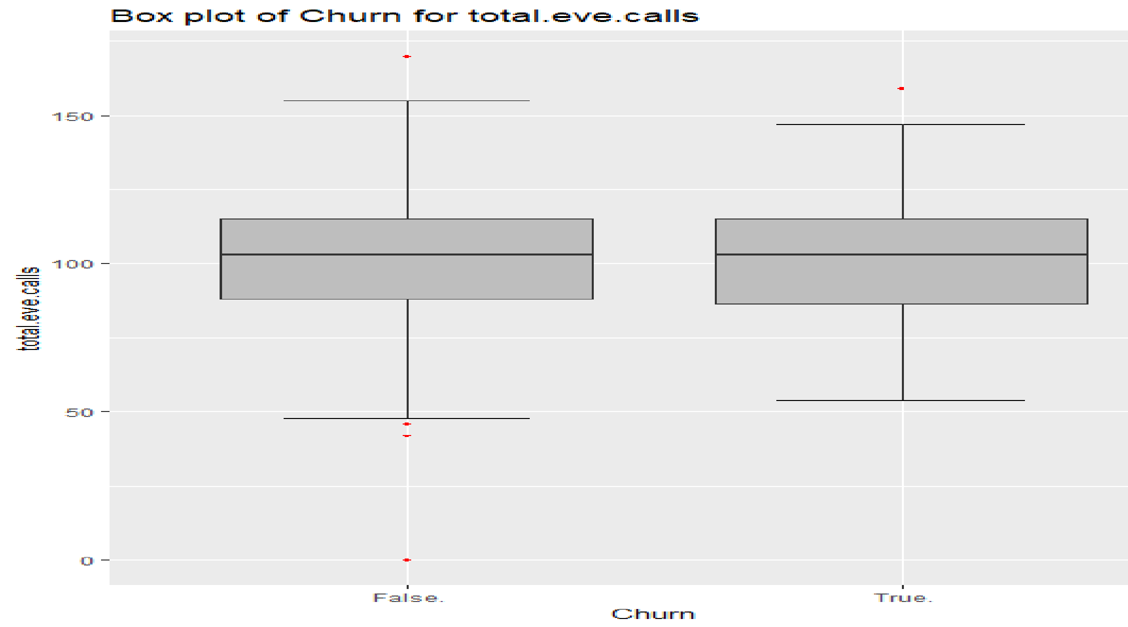
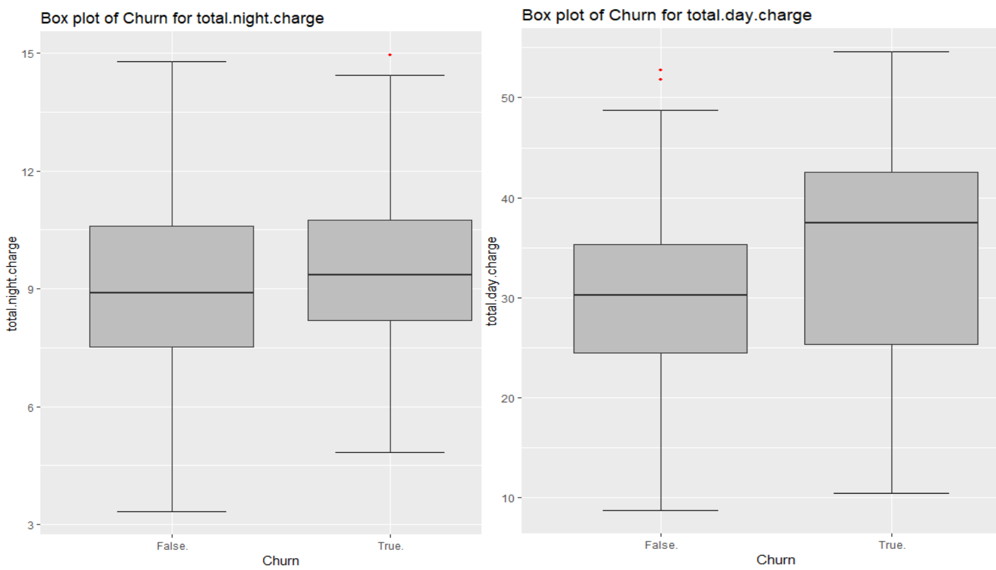
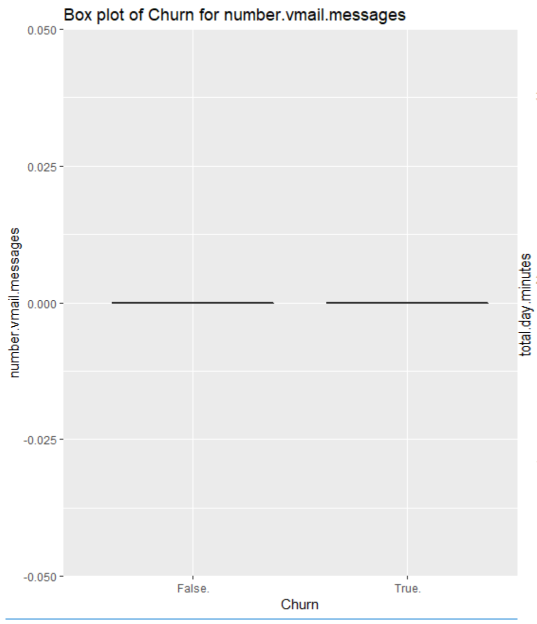


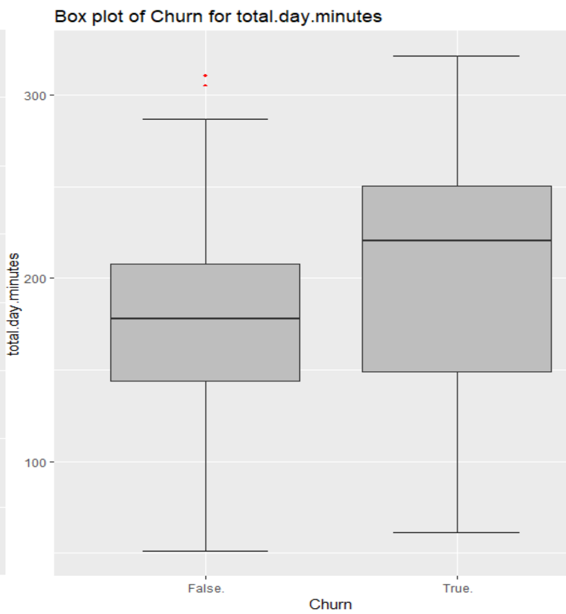
Figure 2.1 Churn Vs Predictor Boxplots(See R code in Appendix**)**

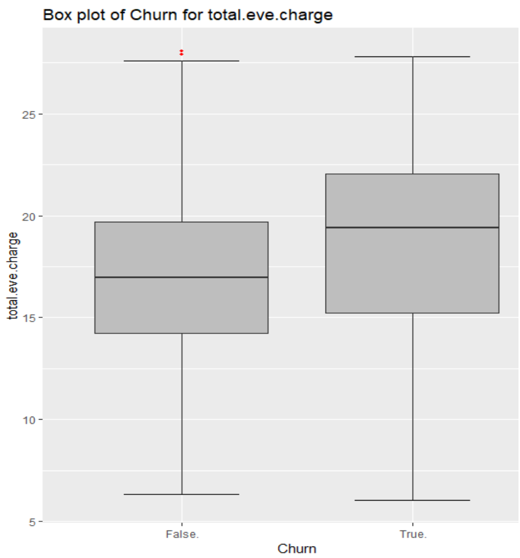
Here the red dots indicates the outliers or the values which is extreme on the given range. We visualize the outliers using boxplot. In figure 2.1 we have plotted the boxplots of the 13 predictor variables with respect to each Churn value. A lot of useful inferences can be made from these plots. First as you can see, we have a lot of outliers and extreme values in each of the data set. For that we first detect the outliers and then remove it from the boxplot also there are multiple methods to impute those outliers. Using different techniques like KNN, Mean, Median & Mode we remove outliers.

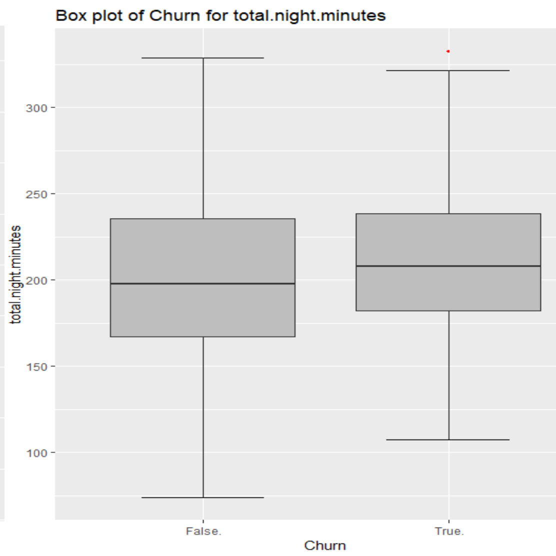
After removal of outliers the boxplot will be like this:

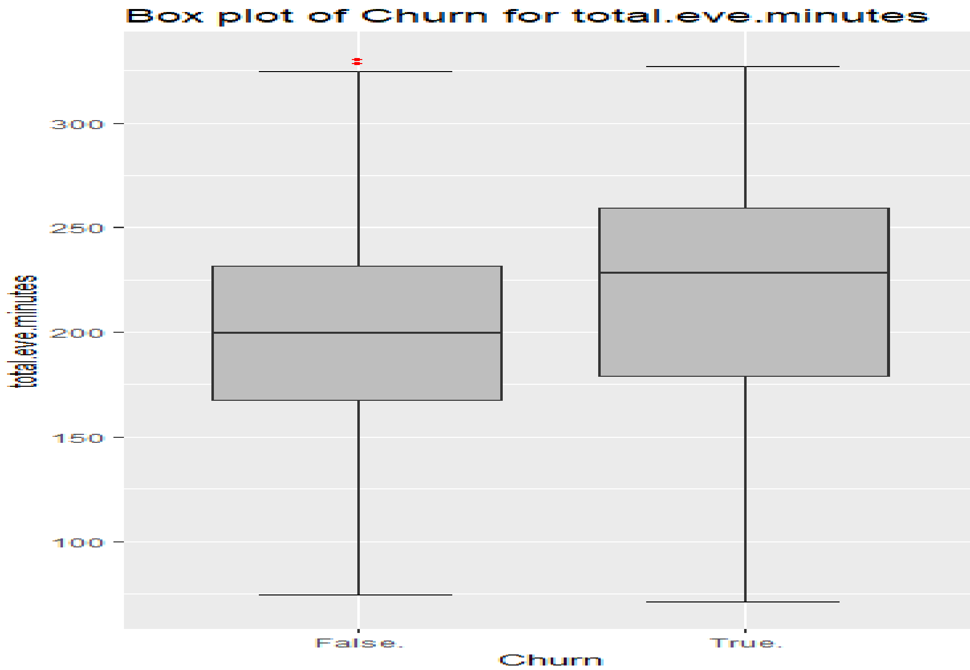


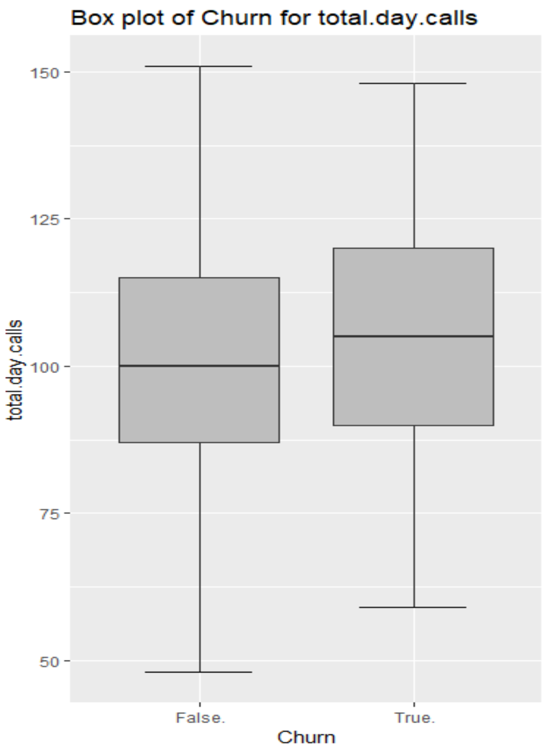

Box plot of Churn for number.vmail.messages
0.050
0.025
0.000
-0.025
-0.050
number.vmail.messages
False.
True.
Churn

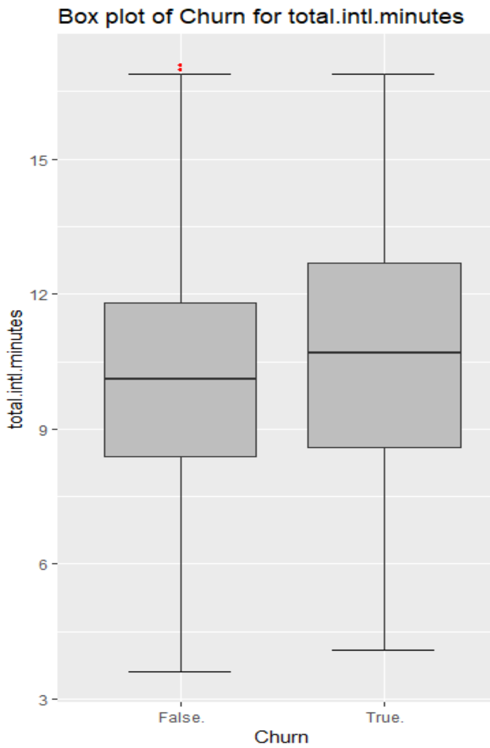

Box plot of Churn for total.day.minutes
300
200
100
total.day.minutes
False.
True.
Churn


Box plot of Churn for total.eve.charge
25
20
15
10
5
total.eve.charge
False.
True.
Churn


Box plot of Churn for total.night.minutes
300
250
200
150
100
total.night.minutes
False.
True.
Churn


Box plot of Churn for total.eve.minutes
300
250
200
150
100
total.eve.minutes
False.
True.
Churn


Box plot of Churn for total.day.calls
150
125
100
75
50
total.day.calls
False.
True.
Churn


Box plot of Churn for total.intl.minutes
15
12
9
6
3
total.intl.minutes
False.
True.
Churn

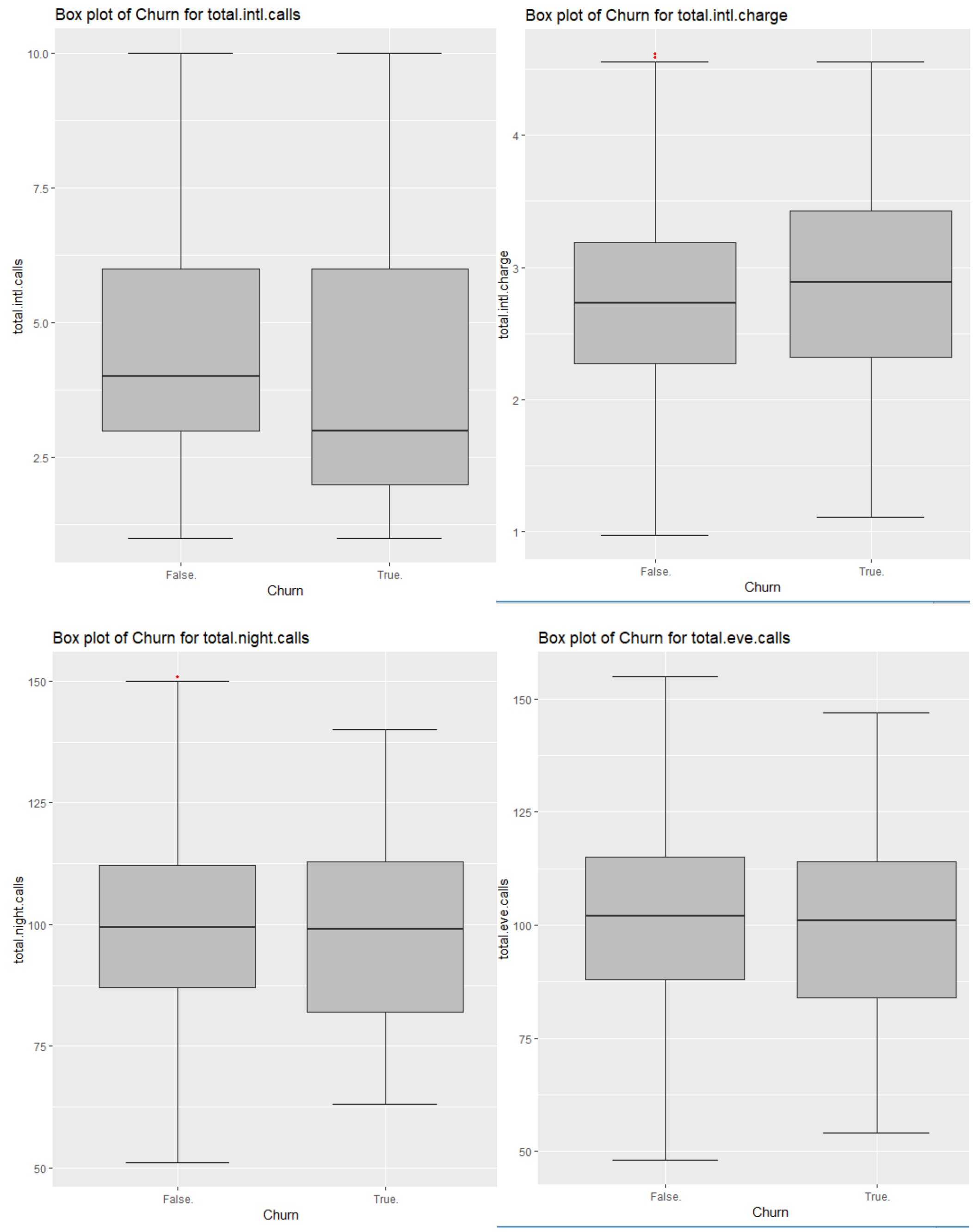


Figure 2.2 Churn Vs Predictor Boxplot After Outlier Removal.(See R code in Appendix**)**

2.1.2 Feature Selection

Before performing any type of modeling, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of churn prediction. There are several methods of doing that.

But before going for feature selection using different method, we first need to know domain knowledge. The dataset contains numerical as well as categorical variables. So, we apply the different analysis on different dataset. The correlation analysis is applied on only numerical data and chi-square test is apply on categorical data. In correlation plot analysis we check the variables which is positively correlated, negatively correlated and zero. The visualization tell about the variables are positively correlated or negatively correlated and according to negatively correlated variables we can drop.

Likewise, in chi-square test, if the p-value > 0.05 then reject the alternate hypothesis and variables are independent of each other, and if p-value < 0.05 then reject the null hypothesis and variables are dependent on each other.

Collect the variable whose p-value less than 0.05 and drop them in given dataset.

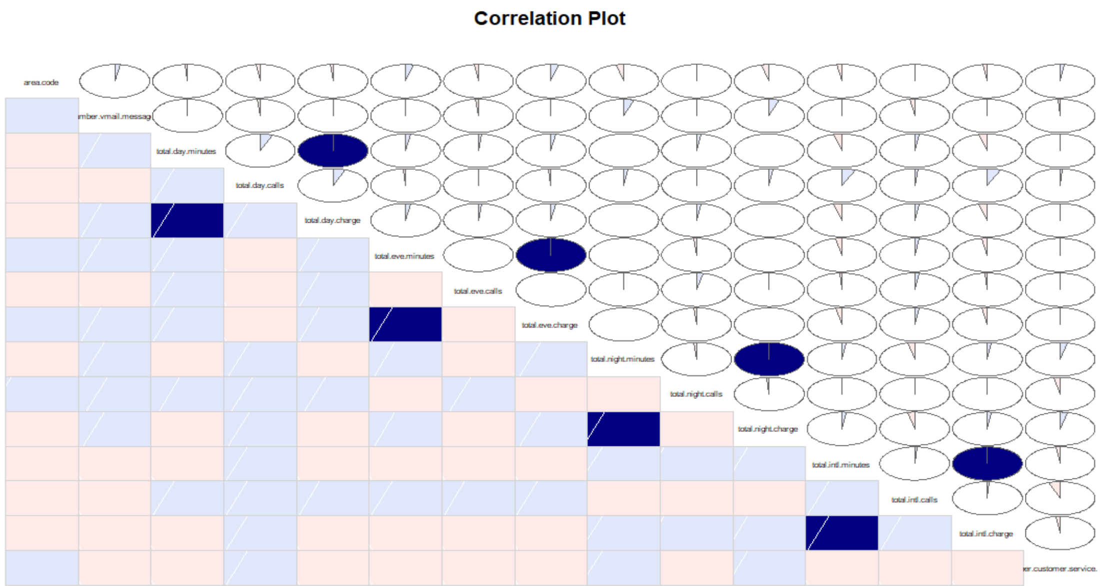


Figure 2.3 Visualization of Correlation Analysis(See R code in Appendix**)**

And Using the chi-square test on categorical variable where none of the variables are < than p-value means we cannot drop the variable through chi-square test.

Below we have used Random Forests to perform features selection.

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

Random Forest can be used to solve regression and classification problems. In regression problems, the dependent variable is continuous. In classification problems, the dependent variable is categorical.

So, In our given dataset the dependent (target) variable is categorical so we solve this problem using classification. There is no chance to solve the problem using regression because the dependent(target) variable is not categorical, it is continuous.

The Visualization of Random Forest is given below:

Random Forest :

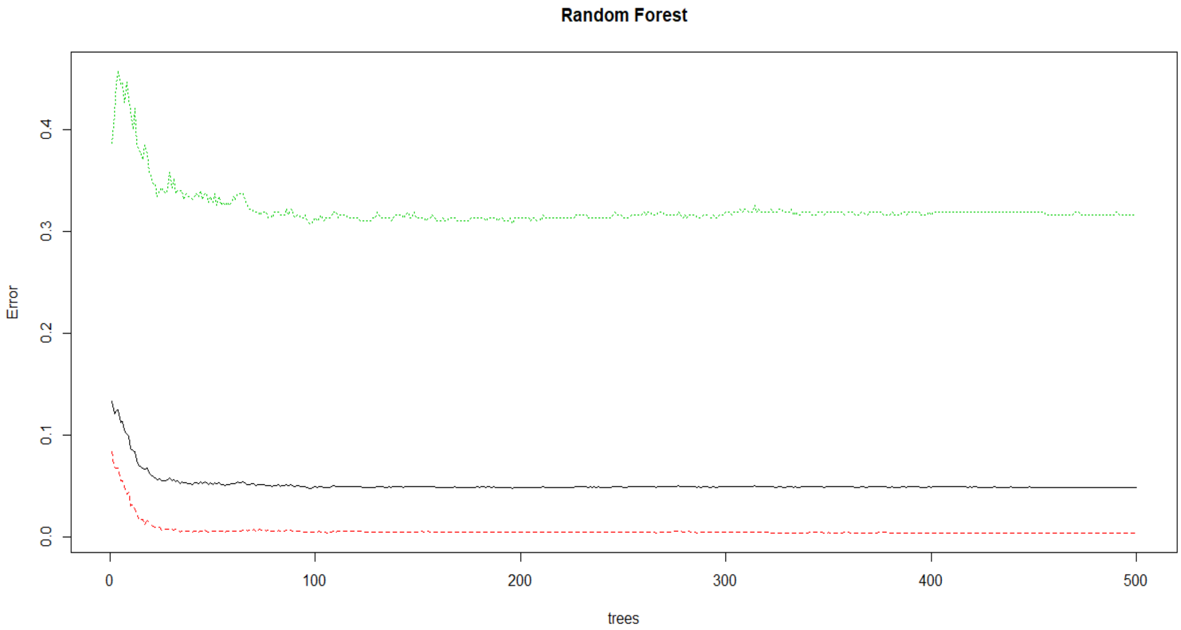


Figure 2.4 Visualization of Random Forest (See R code in Appendix**)**

The above visualization tell us about the number of trees with respective the error.

The first plot indicates the error for your different classes (colored) and out-of-bag samples (black) over the amount of trees. Classes are in the same order as the results you get from print(model), so will be red=False and green=True. You essentially see that the error seems to be lowest around 100 trees is the given example.

For the variable importance as **MeanDecreaseGini** :

MeanDecreaseGini

number.vmail.messages	28.83
total.day.minutes	84.27
total.day.calls	40.70
total.day.charge	86.41
total.eve.minutes	47.51
total.eve.calls	36.20
total.eve.charge	46.83
total.night.minutes	39.03
total.night.calls	33.01
total.night.charge	37.02

total.intl.minutes	33.58
total.intl.calls	26.83
total.intl.charge	33.21

Figure 2.5 Table for MeanDecreaseGini

The MeanDecreaseGini measures the Gini importance = how important the features are *over all splits* done in the tree/forest - whereas for each individual split the Gini importance indicates how much the Gini criterion = "unequality/heterogeneity" was reduced using this split. Because a classification tree essentially tries to built homogeneous groups of samples, so that one (homogeneous) class label can be predicted per group. So it makes sense to check how much features contributed to obtaining such homogeneous groups - which is the end is the MeanDecreaseGini = "variable importance" you see. So, as you can clearly see, **total.day.charge** and **total.day.minutes** contributed most to obtaining such splits, so they are considered more important.

2.2 Modeling

2.2.1 Model Selection

In our early stages of analysis during pre-processing we have come to understand that ,using correlation plot and chi-square test the result will be different. Therefore, we can neither combine the data sets nor use a single model for predicting variables. Hence, we need to analyse the data sets separately and generate separate models for data sets.

The dependent variable can fall in either of the four categories:

1. Nominal

2. Ordinal

3. Interval

4. Ratio

If the dependent variable, in our case Churn, is Nominal the only predictive analysis that we can perform is Classification, and if the dependent variable is Interval or Ratio the normal method is to do a Regression analysis, or classification after binning. But the dependent variable we are dealing with is Nominal, for which only classification can be done, because the Churn variable has categories.

You always start your model building from the simplest to more complex. Therefore, we use Logistic Regression.

2.2.2 Logistic Regression

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

```
> logit_model<-glm(Churn ~state + international.plan + voice.mail.plan, data=sample_set ,family = bi
> summary(logit_model)

Call:
glm(formula = Churn ~ state + international.plan + voice.mail.plan,
    family = binomial, data = sample_set)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4827  -0.5894  -0.4085  -0.2498   2.8286

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -3.45525    1.01881  -3.391 0.000695 ***
stateAL                  0.97282    1.14635   0.849 0.396087
stateAR                  1.89300    1.09260   1.733 0.083172 .
stateAZ                -13.95429  528.49604  -0.026 0.978935
stateCA                  2.69648    1.11795   2.412 0.015866 *
stateCO                  1.10533    1.15710   0.955 0.339450
stateCT                  1.80228    1.08614   1.659 0.097045 .
stateDC                  1.73842    1.13732   1.529 0.126382
stateDE                  1.19688    1.12393   1.065 0.286917
stateFL                  1.59459    1.10306   1.446 0.148287
stateGA                  1.48086    1.15611   1.281 0.200228
stateHI                -13.96128  643.51718  -0.022 0.982691
stateIA                  0.42258    1.44202   0.293 0.769486
stateID                  2.01704    1.08409   1.861 0.062802 .
stateIL                  0.23219    1.26694   0.183 0.854589
stateIN                  2.24281    1.06097   2.114 0.034522 *
stateKS                  1.72571    1.09676   1.573 0.115613
stateKY                  1.27487    1.13540   1.123 0.261507
stateLA                  1.32398    1.20480   1.099 0.271804
stateMA                  1.01342    1.16319   0.871 0.383620
stateMD                  1.99630    1.07129   1.863 0.062398 .
stateME                  1.79291    1.11185   1.613 0.106844
stateMI                  2.12797    1.07720   1.975 0.048215 *
stateMN                  1.75790    1.08798   1.616 0.106147

stateND                  0.48298    1.18516   0.408 0.683625
stateNE                  1.77991    1.11080   1.602 0.109074
stateNH                  1.48372    1.11652   1.329 0.183887
stateNJ                  2.64079    1.06603   2.477 0.013241 *
stateNM                  0.69975    1.18838   0.589 0.555979
stateNV                  1.78523    1.10112   1.621 0.104958
stateNY                  1.80515    1.08874   1.658 0.097316 .
stateOH                  0.99356    1.15123   0.863 0.388114
stateOK                  2.20664    1.10191   2.003 0.045225 *
stateOR                  1.20700    1.11343   1.084 0.278347
statePA                  1.25072    1.13965   1.097 0.272439
stateRI                 -0.09568    1.26464  -0.076 0.939691
stateSC                  2.79067    1.07801   2.589 0.009633 **
stateSD                  2.16140    1.09140   1.980 0.047660 *
stateTN                  1.08791    1.20514   0.903 0.366674
stateTX                  1.89345    1.08031   1.753 0.079653 .
stateUT                  1.54275    1.11720   1.381 0.167307
stateVA                  0.26841    1.19001   0.226 0.821547
stateVT                  1.73724    1.10329   1.575 0.115347
stateWA                  2.01236    1.09405   1.839 0.065861 .
stateWI                  0.95862    1.15280   0.832 0.405658
stateWV                  1.50431    1.09004   1.380 0.167571
stateWY                  1.24462    1.11599   1.115 0.264736
international.plan yes   1.90642    0.17427  10.939  < 2e-16 ***
voice.mail.plan yes     -1.00979    0.17247  -5.855 4.78e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1919.7  on 2332  degrees of freedom
Residual deviance: 1659.1  on 2280  degrees of freedom
AIC: 1765.1

Number of Fisher Scoring iterations: 16
```

As you can see the *probability value*, if p-value > 0.05 then we reject the alternate hypothesis that variables are independent of each other. That variables are StateNY StateTX, StateWA.And the *** indicates the if p-value < 0.05 then we reject null hypothesis that target variable does not depend on **international.plan** and **voice.mail.plan** of the predictor variable.

The Accuracy of the model will be less but the FNR will be greater. We can also display the prediction using plots.

Logistic Regression is part of a larger class of algorithm known as Generalized Linear Model(glm).

Using glm() we can predict the output. The following are the different different plots which identify the logistic regression model.

- **The Residuals vs Fitted plot** can help you see, for example, if there are curvilinear trends that you missed. But the fit of a logistic regression is curvilinear by nature, so you can have odd looking trends in the residuals with nothing amiss.
- **The Normal Q-Q plot** helps you detect if your residuals are normally distributed. But the deviance residuals don't have to be normally distributed for the model to be valid, so the normality / non-normality of the residuals doesn't necessarily tell you anything.
- **The Scale-Location plot** can help you identify heteroscedasticity. But logistic regression models are pretty much heteroscedastic by nature.
- **The Residuals vs Leverage** can help you identify possible outliers. But outliers in logistic regression don't necessarily manifest in the same way as in linear regression, so this plot may or may not be helpful in identifying them.

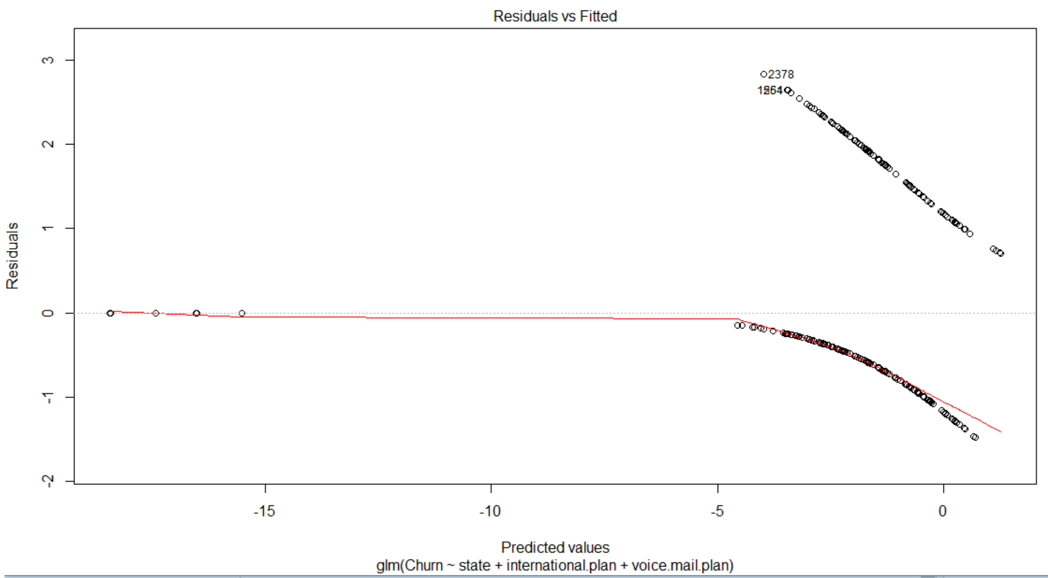


Figure : Residuals Vs Fitted Plot

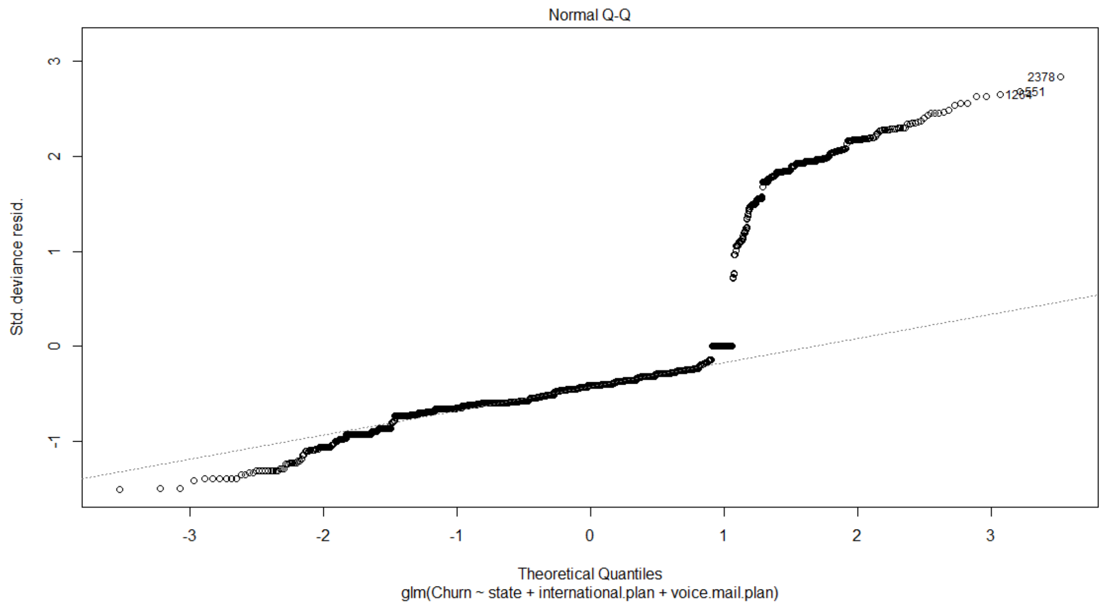


Figure :Normal Q-Q Plot

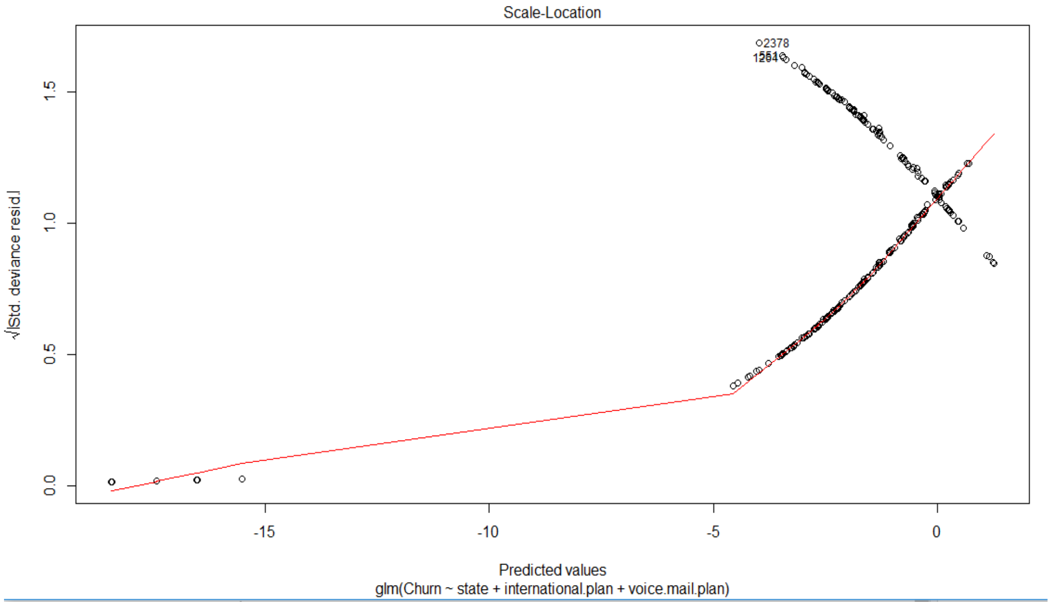


Figure : Scale – Location

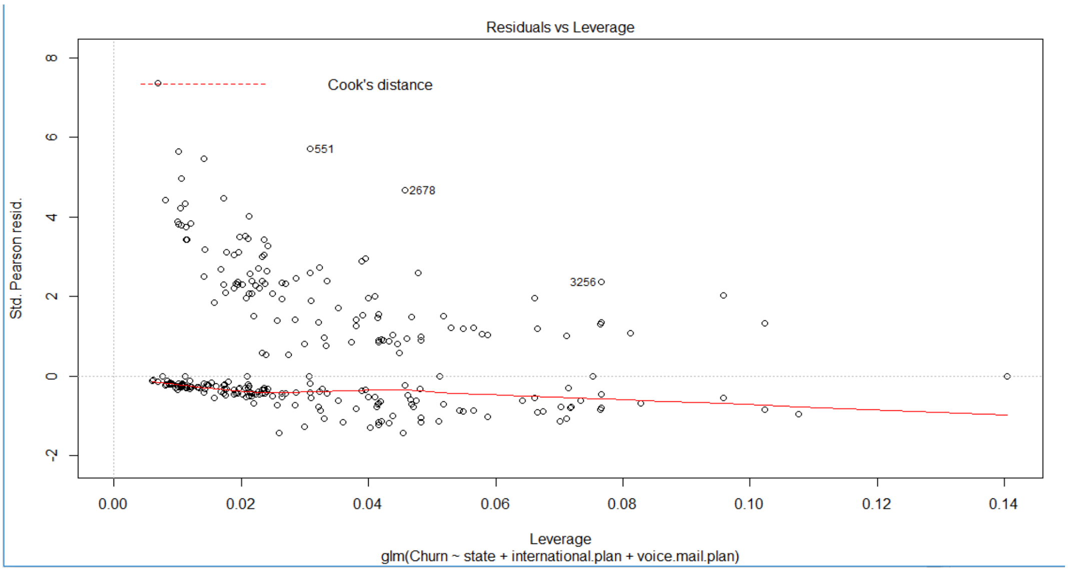


Figure : Residuals Vs Leverage

Figure 2.6 Logistic Regression Plots (See R code in Appendix**)**

2.2.3 Decision Tree / Classification Tree

Decision Tree is a supervised machine learning algorithm. Supervised means have a target variable. Decision tree used in a variety of ways to solve regression as well as classification model.

Classification trees, as the name implies are used to separate the dataset into classes belonging to the response variable. Usually the response variable has two classes:Yes or No (1 or 0). If the target variable has ***more*** than 2 categories, then a variant of the algorithm, called C4.5, is used. For binary splits however, the standard CART procedure is used. Thus classification trees are used when the response or target variable is categorical in nature.

Regression trees are needed when the response variable is numeric or continuous. For example, the predicted price of a consumer good. Thus, regression trees are applicable for *prediction* type of problems as opposed to *classification*.

So, here in given dataset the target variable (Churn) is present which contain two category that is TRUE and FALSE. So, we solve this problem using C5.0 and CART as well.

Here we not used Regression Tree because the target or response variable is not numeric or continuous. The difference between these two algorithms is not so far, accuracy is also near. The Visualization of both algorithms is different.C5.0 Model is multi split tree and CART has binary split tree.

So here we conclude that both algorithms give the high accuracy. No one is better than other, both give the results.

The Classification/Decision Tree will be shown:

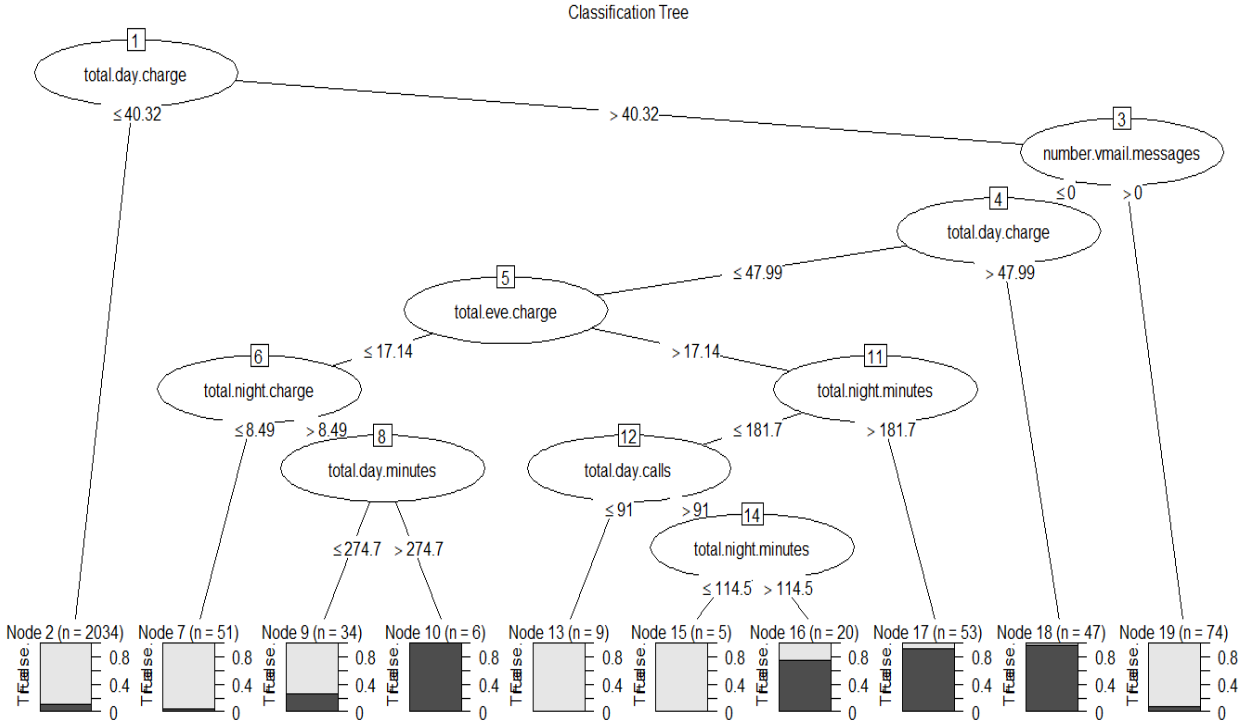


Figure 2.7 Decision tree using C5.0 Model (See R code in Appendix**)**

The Classification/Decision Tree will be shown:

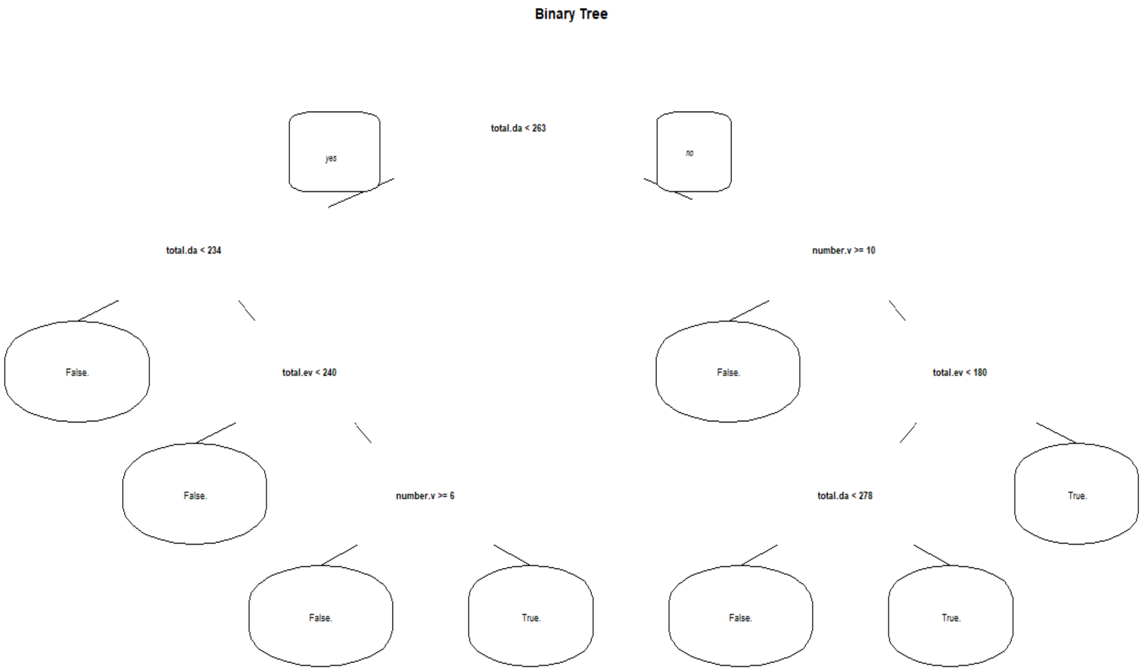


Figure 2.8 Decision Tree using CART Model(See R code in Appendix**)**

2.2.3 Regression

Using Regression for prediction analysis in this case is not interval or ratio ,though it can be done. The reason is, the values of the target variable, Churn is categorical that is the form of TRUE,FALSE or YES,NO. Though Regression predictions have been done on interval or ratio variables it is not a recommended approach, because the information stored in terms of categorical variables.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Confusion Matrix
2. False Negative Rate

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Confusion Matrix

Confusion Matrix is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous section.

One by one we can display Confusion Matrix of models.

A) Decision Tree:

```
> confMatrix_C50=table(test$Churn,C50_predictors)
> confusionMatrix(confMatrix_C50)
Confusion Matrix and Statistics

         C50_predictors
           False.  True.
  False.    1422     21
  True.      142     82

               Accuracy : 0.9022
                 95% CI : (0.8869, 0.9161)
    No Information Rate : 0.9382
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4554
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9092
            Specificity : 0.7961
         Pos Pred Value : 0.9854
         Neg Pred Value : 0.3661
             Prevalence : 0.9382
         Detection Rate : 0.8530
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8527

       'Positive' Class :  False.
```

C5.0 Model :

Accuracy : 90.22 %

```
> confMatrix_cart=table(test$Churn,cart_predictors)
> confusionMatrix(confMatrix_cart)
Confusion Matrix and Statistics

         cart_predictors
           False.  True.
   False.    1428     15
   True.      151     73

               Accuracy : 0.9004
                 95% CI : (0.885, 0.9144)
    No Information Rate : 0.9472
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4243
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9044
            Specificity : 0.8295
         Pos Pred Value : 0.9896
         Neg Pred Value : 0.3259
             Prevalence : 0.9472
         Detection Rate : 0.8566
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8670

       'Positive' Class :  False.
```

CART Model :

Accuracy : 90.04 %

B) Random Forest:

```
> confMatrix_randomForest=table(test$Churn,randomForest_predictors)
> confusionMatrix(confMatrix_randomForest)
Confusion Matrix and Statistics

         randomForest_predictors
           False.  True.
   False.    1438      5
   True.      150     74

               Accuracy : 0.907
                 95% CI : (0.8921, 0.9205)
    No Information Rate : 0.9526
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4499
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9055
            Specificity : 0.9367
         Pos Pred Value : 0.9965
         Neg Pred Value : 0.3304
             Prevalence : 0.9526
         Detection Rate : 0.8626
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9211

       'Positive' Class :  False.
```

Accuracy = 90.7 %

C) Logistic Regression:

```
> confMatrix_logit=table(test$Churn,logit_predictions)
> confMatrix_logit
        logit_predictions
            0    1
  False. 1409   34
  True.   192   32
```

Accuracy = (TP + TN) / (TP + TN +FP +FN)

=(1409 + 32)/ 1667

=1441 / 1667

Accuracy = **86.44 %**

3.1.1 False Negative Rate :

A) Decision Tree:

The Confusion Matrix for C5.0 Model is:

```
Confusion Matrix and Statistics

         C50_predictors
          False.  True.
  False.    1422     21
  True.      142     82
```

The False Positive Rate for C5.0 Model is:

FNR = FN/(FN+TP)

=142 / (142 + 82)

FNR = 63.39%

The Confusion Matrix for CART Model is:

```
Confusion Matrix and Statistics

         cart_predictors
          False.  True.
  False.    1428     15
  True.      151     73
```

The False Positive Rate for CART Model is:

FNR = FN/(FN+TP)

=151 / (151 + 73)

FNR = 67.41%

B) Random Forest:

The Confusion Matrix is :

```
Confusion Matrix and Statistics

          randomForest_predictors
           False.  True.
   False.    1438      5
   True.      150     74
```

The False Negative Rate is :

FNR = FN/(FN+TP)

=150 / (150 + 74)

FNR = 66.96%

C) Logistic Regression:

The Confusion Matrix is :

```
> confMatrix_logit=table(test$Churn,logit_predictions)
> confMatrix_logit
        logit_predictions
            0    1
  False. 1409   34
  True.   192   32
```

The False Negative Rate is :

FNR = FN/(FN+TP)

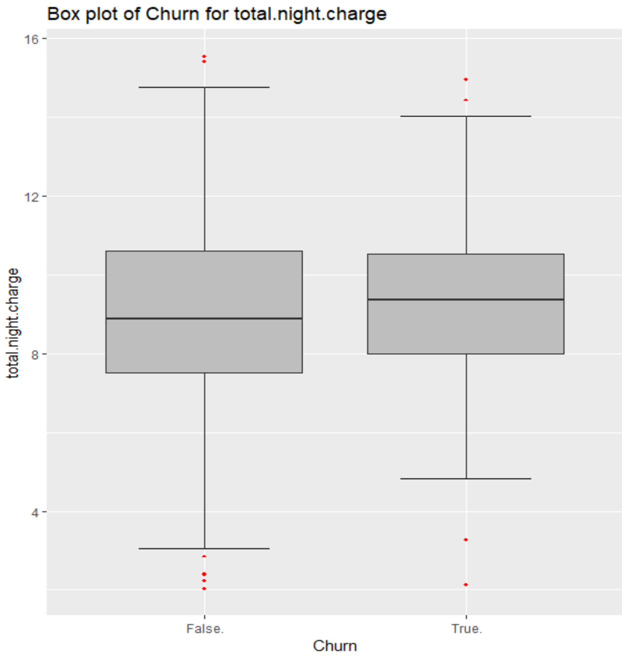
=192 / (192 + 32)

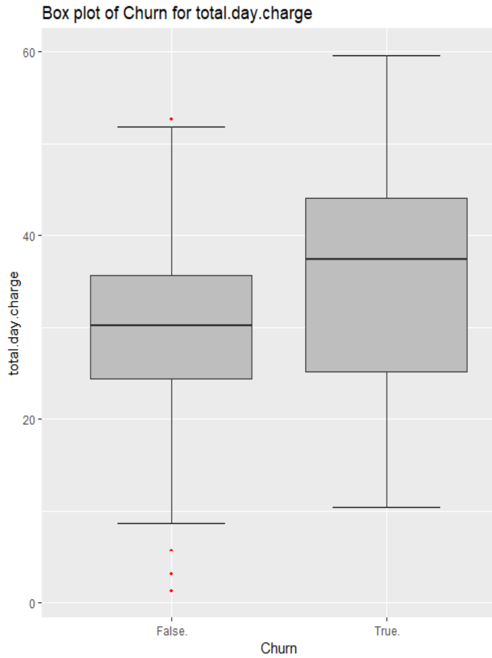
FNR = 85.71%

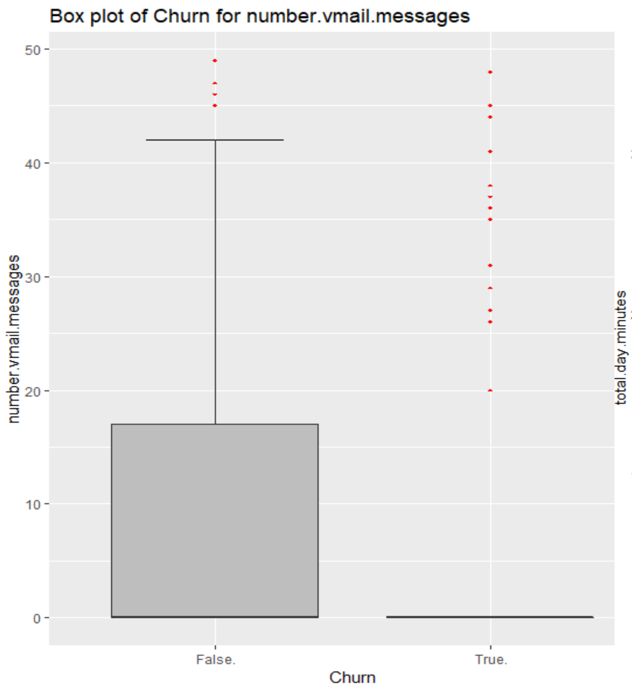
3.2 Model Selection

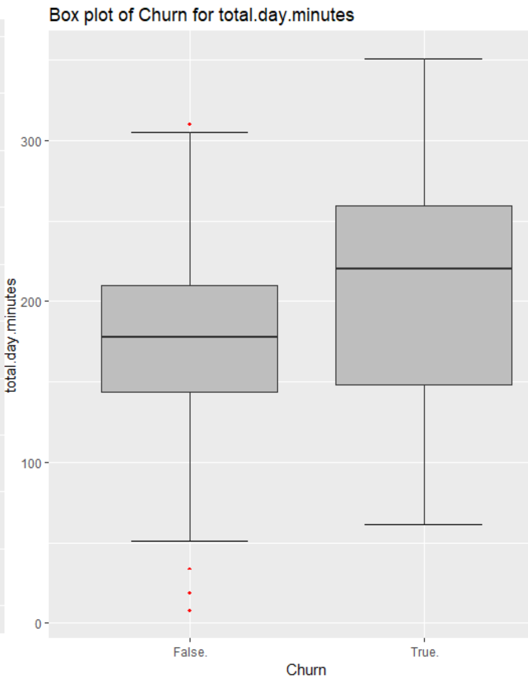
We can see that out of three models two models perform comparatively on average and therefore we can select either of the two models without any loss of information.

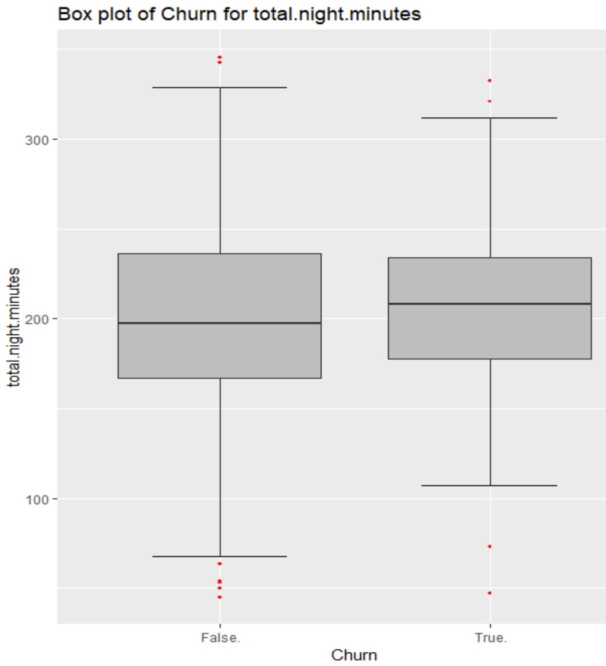
Appendix A - Extra Figures

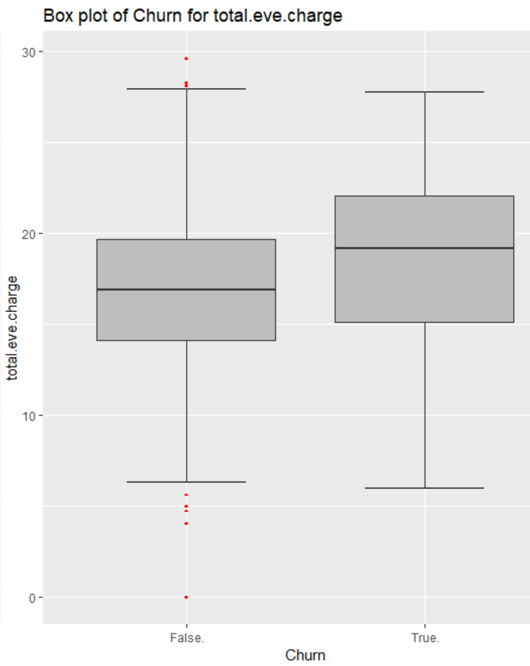

Box plot of Churn for total.night.charge
16
12
8
4
total.night.charge
False.
True.
Churn

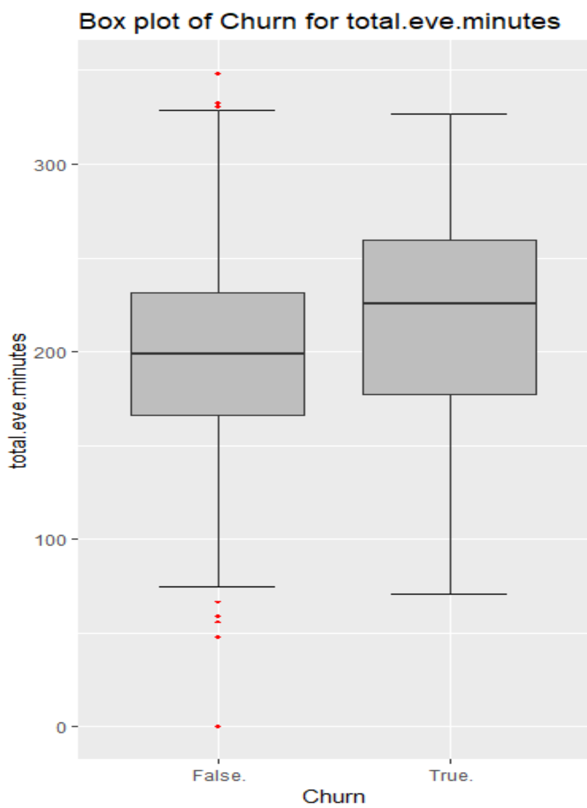

Box plot of Churn for total.day.charge
60
40
20
0
total.day.charge
False.
True.
Churn

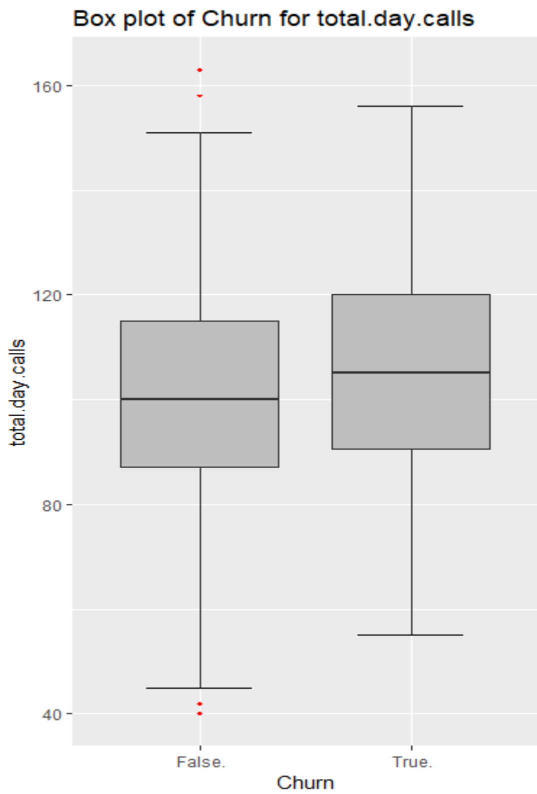

Box plot of Churn for number.vmail.messages
50
40
30
20
10
0
number.vmail.messages
False.
True.
Churn

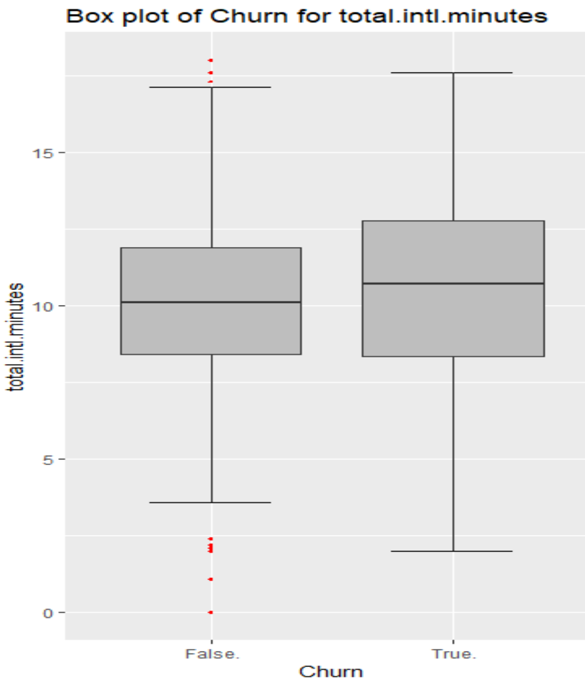

Box plot of Churn for total.day.minutes
300
200
100
0
total.day.minutes
False.
True.
Churn

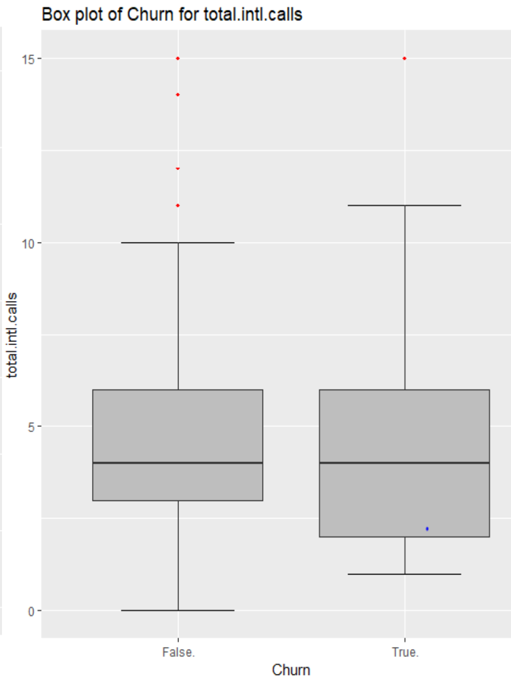

Box plot of Churn for total.night.minutes
total.night.minutes
300
200
100
False.
True.
Churn

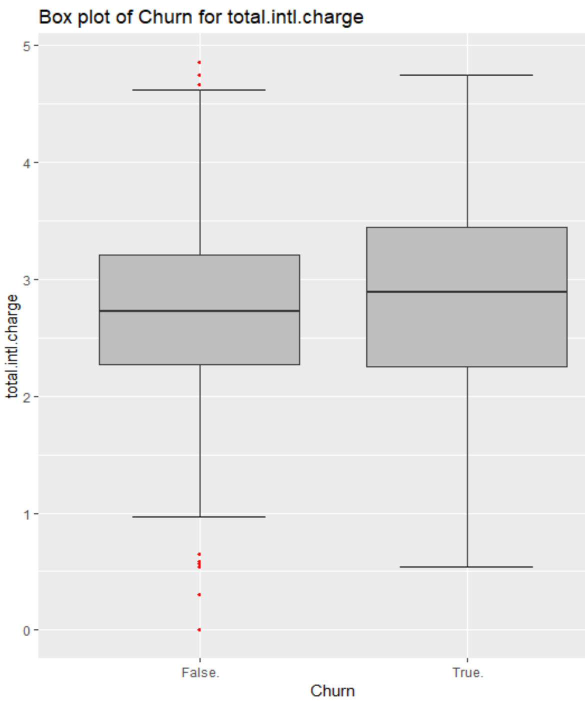

Box plot of Churn for total.eve.charge
total.eve.charge
30
20
10
0
False.
True.
Churn

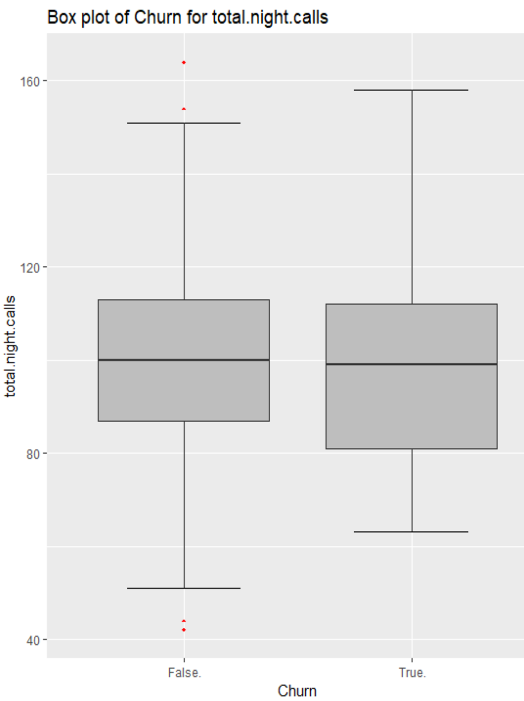

Box plot of Churn for total.eve.minutes
total.eve.minutes
300
200
100
0
False.
True.
Churn


Box plot of Churn for total.day.calls
total.day.calls
160
120
80
40
False.
True.
Churn


Box plot of Churn for total.intl.minutes
total.intl.minutes
0
5
10
15
False.
True.
Churn


Box plot of Churn for total.intl.calls
total.intl.calls
0
5
10
15
False.
True.
Churn


Box plot of Churn for total.intl.charge
total.intl.charge
0
1
2
3
4
5
False.
True.
Churn


Box plot of Churn for total.night.calls
total.night.calls
40
80
120
160
False.
True.
Churn

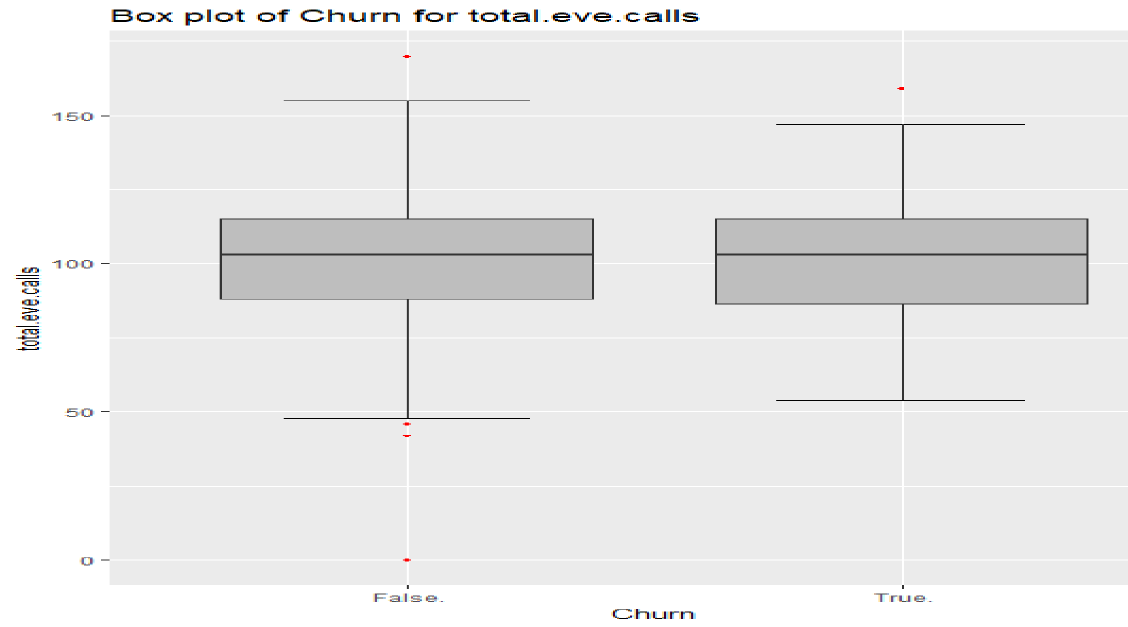
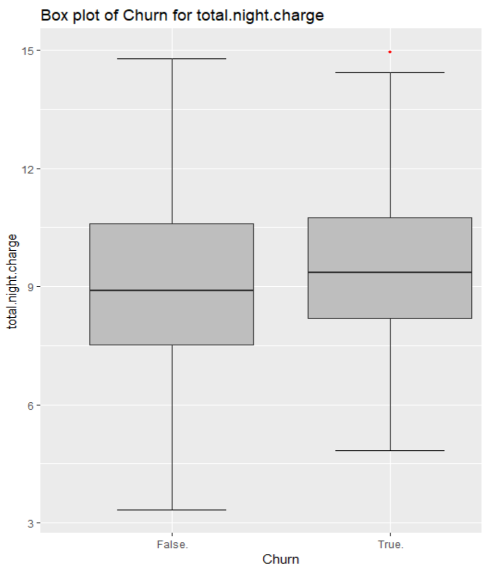
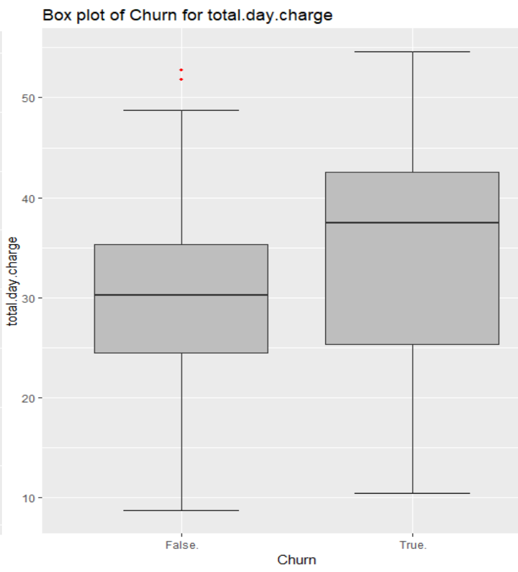
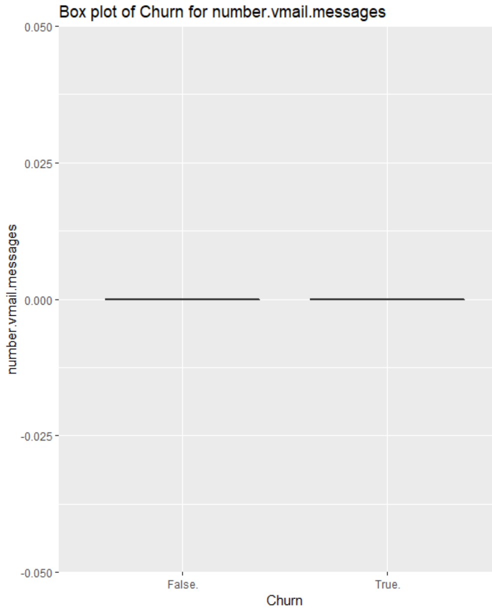


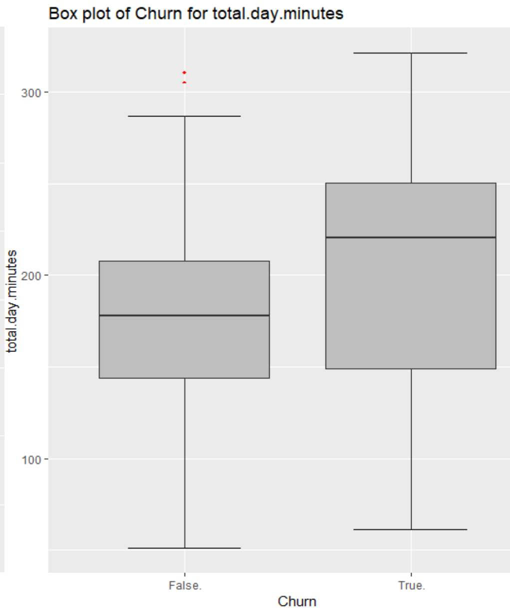
Figure 2.1 Churn Vs Predictor Boxplots (See R code in Appendix)

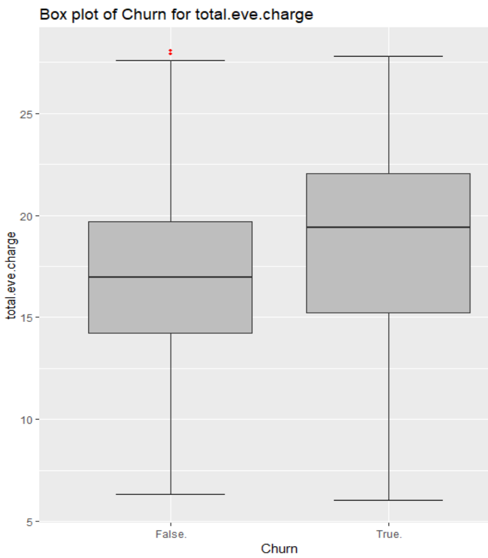
After removal of outliers the boxplot will be like this:

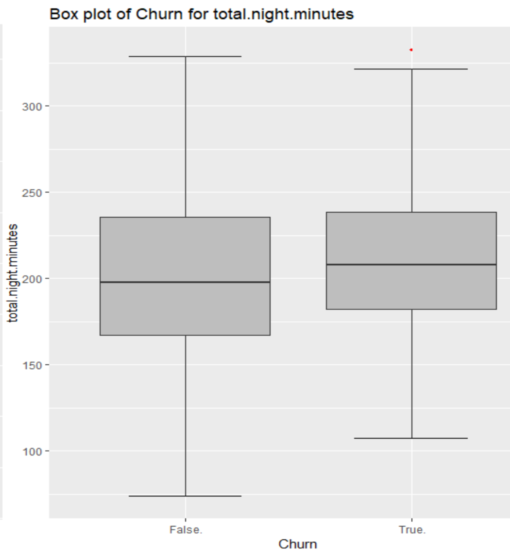


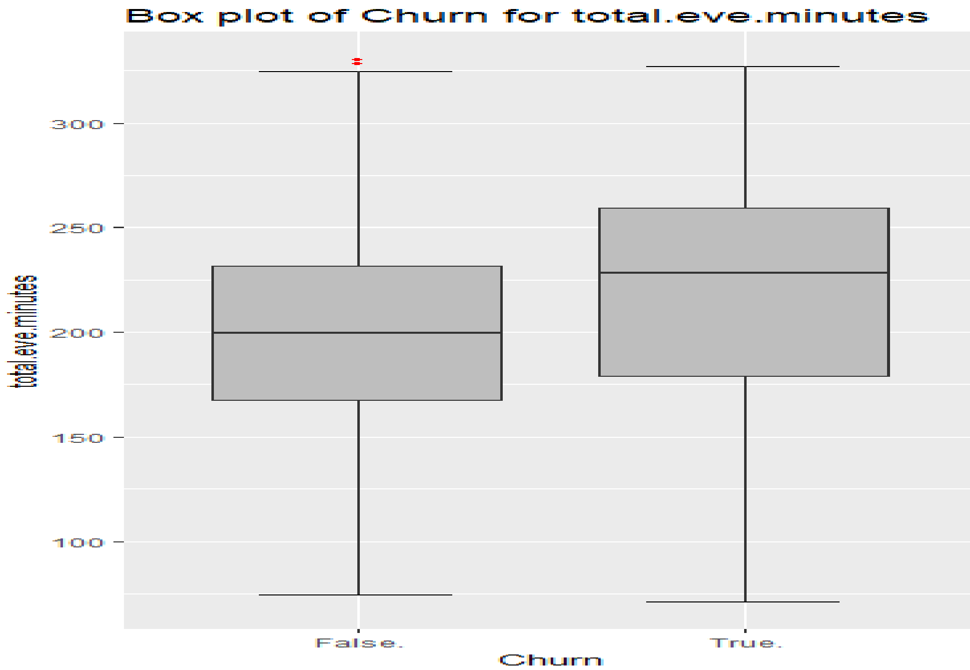


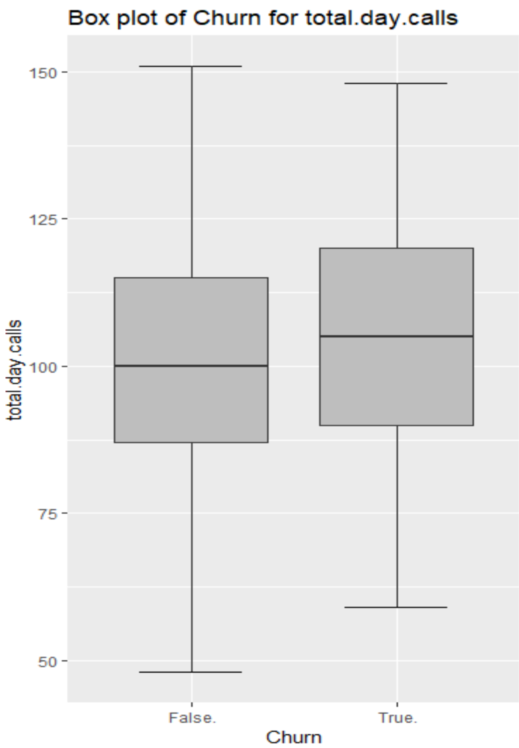

Box plot of Churn for number.vmail.messages
number.vmail.messages
0.050
0.025
0.000
-0.025
-0.050
False.
True.
Churn

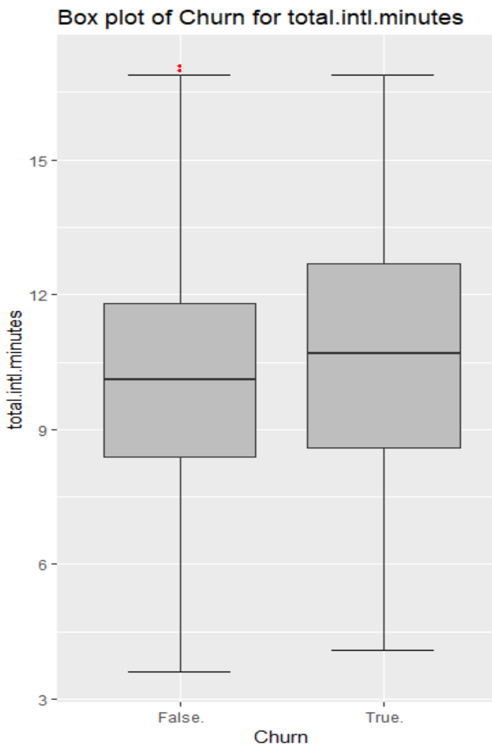

Box plot of Churn for total.day.minutes
total.day.minutes
300
200
100
False.
True.
Churn


Box plot of Churn for total.eve.charge
total.eve.charge
25
20
15
10
5
False.
True.
Churn


Box plot of Churn for total.night.minutes
total.night.minutes
300
250
200
150
100
False.
True.
Churn


Box plot of Churn for total.eve.minutes
300
250
200
150
100
total.eve.minutes
False.
True.
Churn


Box plot of Churn for total.day.calls
150
125
100
75
50
total.day.calls
False.
True.
Churn


Box plot of Churn for total.intl.minutes
15
12
9
6
3
total.intl.minutes
False.
True.
Churn

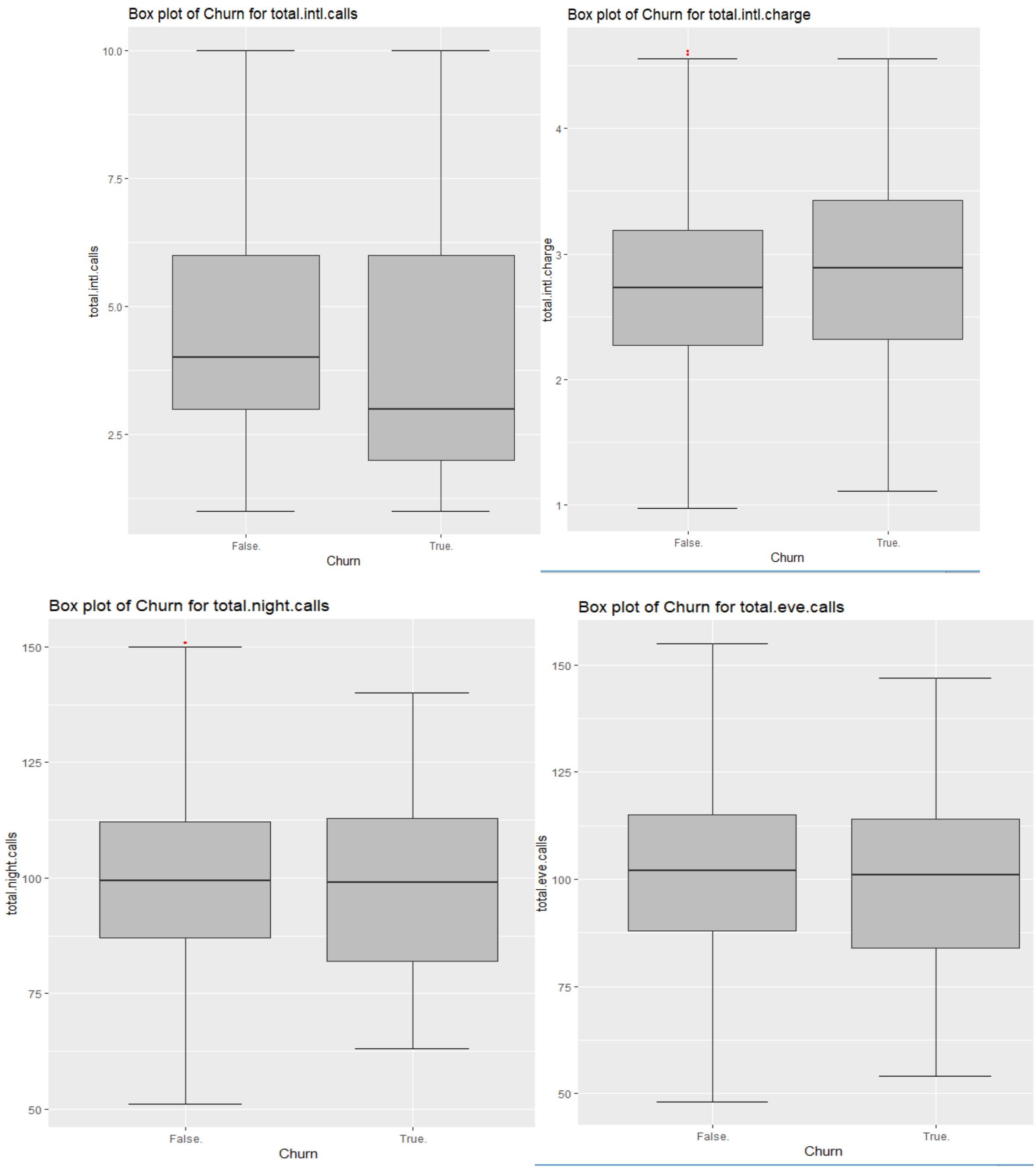


Figure 2.2 Churn Vs Predictor Boxplot After Outlier Removal.(See R code in Appendix**)**

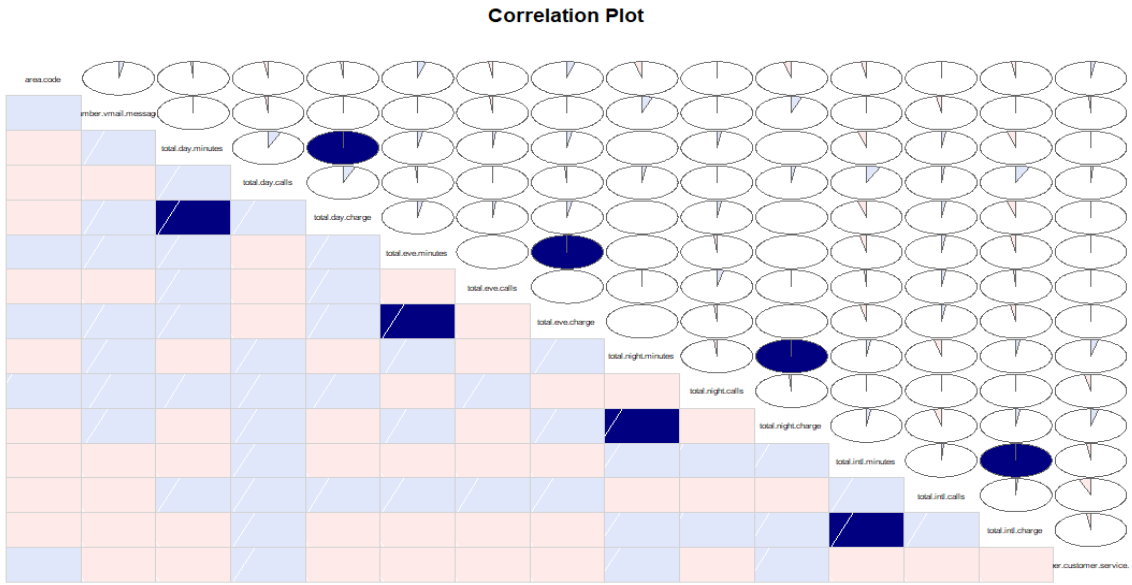


Figure 2.3 Visualization of Correlation Analysis(See R code in Appendix**)**

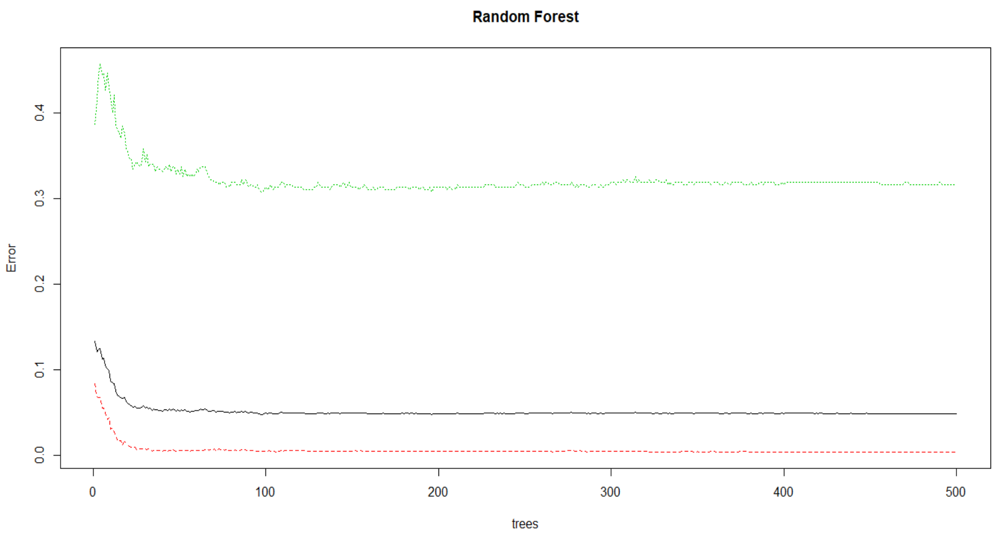


Figure 2.4 Visualization of Random Forest(See R code in Appendix**)**

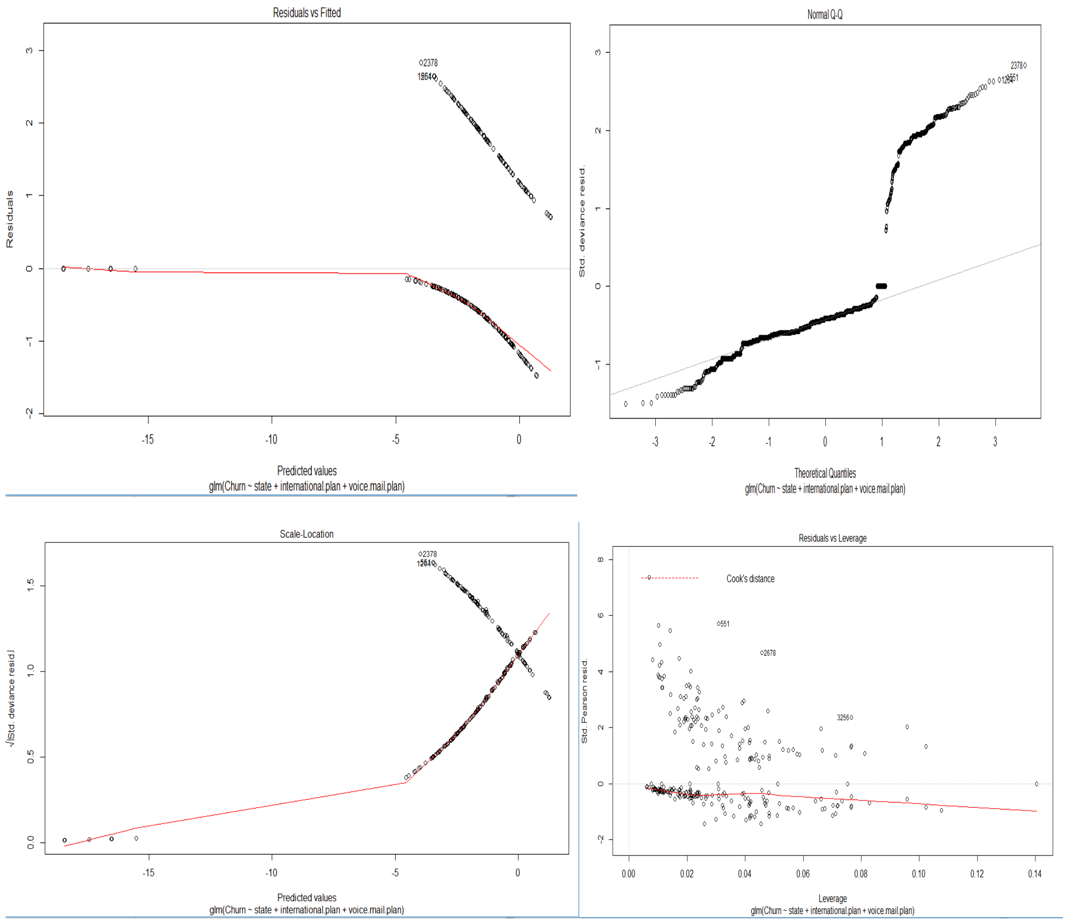


Figure 2.6 Logistic Regression Plots(See R code in Appendix**)**

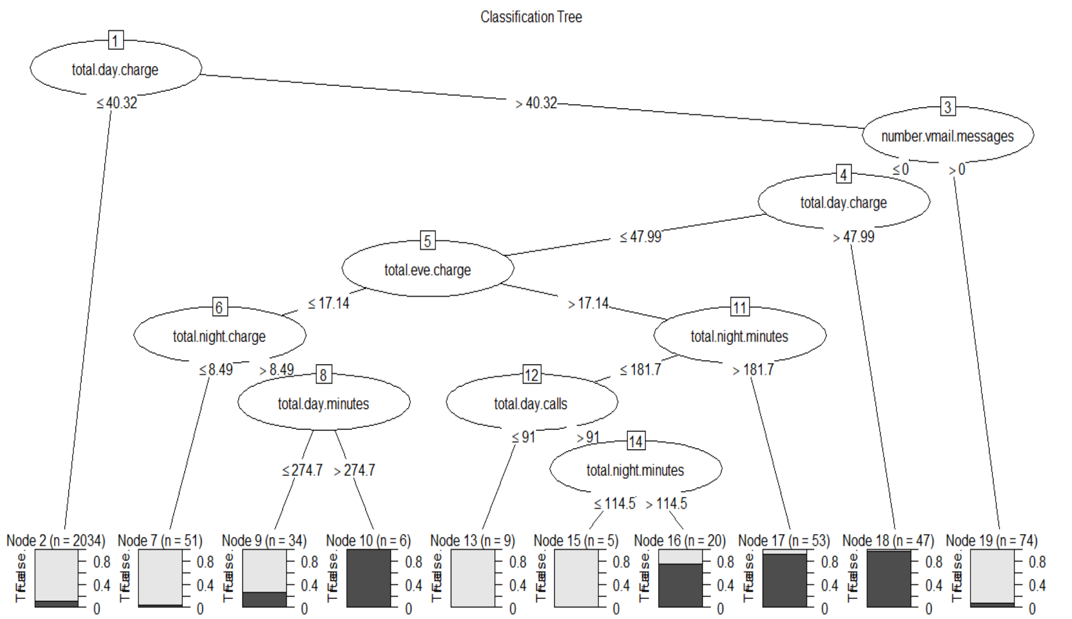


Figure 2.7 Decision tree using C5.0 Model (See R code in Appendix**)**

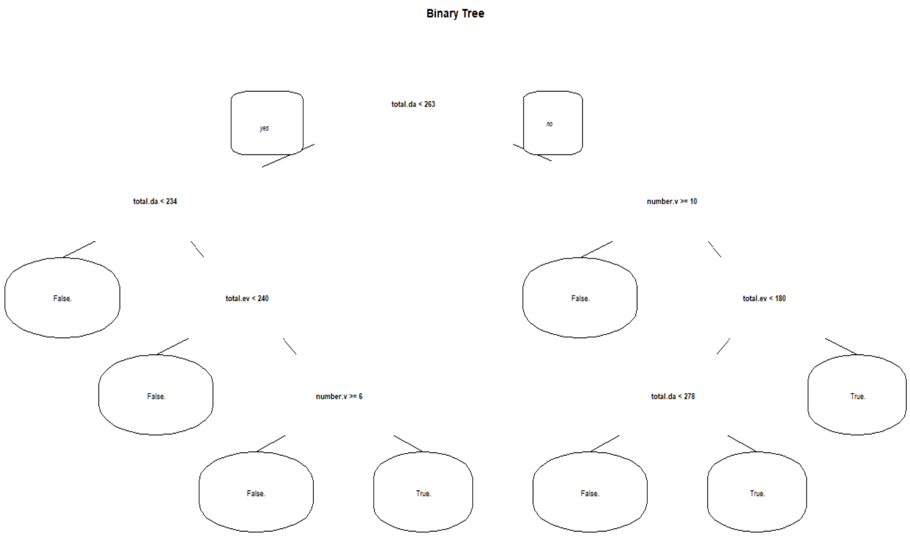


Figure 2.8 Decision Tree using CART Model (See R code in Appendix**)**

Appendix B - R Code

Boxplots for all predictors variables (Fig. 2.1)

```
#Plotting the Boxplot
install.packages("caret")
library(caret)

for(i in 1:length(new_cnames))
{
  assign(paste0("bx",i),ggplot(aes_string(y=(new_cnames[i]),x="Churn"),data=subset(new_set))+
           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=new_cnames[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",new_cnames[i])))
  print(i)
}

#Plotting Plots Together
gridExtra::grid.arrange(bx10,bx4,ncol=2)
gridExtra::grid.arrange(bx1,bx2,ncol=2)
gridExtra::grid.arrange(bx8,bx7,ncol=2)
gridExtra::grid.arrange(bx5,bx3,bx11,ncol=3)
gridExtra::grid.arrange(bx12,bx13,ncol=2)
gridExtra::grid.arrange(bx9,bx6,ncol=2)
```

Correlation Plot (Fig. 2.3)

```
#Load the Package:
install.packages("corrgram")
library(corrgram)

#Correlation Plot
corrgram(sample_set[,sample_numeric_index],order=F,
         upper.panel = panel.pie ,text.panel = panel.txt,method="color",main="Correlation Plot")
```

Random Forest Plot (Fig. 2.4)

```
#Plot the Random Forest
plot(RF_model , main = "Random Forest")
```

Logistic Regression Plot (Fig. 2.6)

```
#Plot the Logistic Regression
plot(logit_model)
```

Decision Tree Using C5.0 and CART (Fig. 2.7 & Fig. 2.8)

```
#Plot the decision tree
plot(C50_model, main = "Classification Tree")

#Plot the tree using prp command defined in rpart.plot package
prp(cart_model, main = "Binary Tree")
```

Complete R File

```
#Remove The Variable & Values In Environment
rm(list=ls())

#Set The Directory
setwd("F:/Eddwisor/Task Program/Projects")

#Get The Directory
getwd()

#------Load the Train dataset-------#
train=read.csv("Train_data.csv",encoding = 'ISO-8859-1')

#-----Check Dimension-------#
dim(train)             #rows=3333 and column=21

#Get the Column Names
names(train)

#Get the structure of the dataset
str(train)
head(train,4)

#----------------------------------------------------------------------

  #No need of account length for churn reduction
  #get the index of column account length and remove it from the dataset

  account_length_index = match("account.length",names(train))
  account_length_index
  train=train[-account_length_index]
#----------------------------------------------------------------------

#After Removing The columns Check The train Dataset
  str(train)
#Original dataset have 8 num variables, 7 int and 5 factor
#
```

```
#Seperate The Numeric & categorical variables In Actual Train Dataset

#Seperate Numeric Variables :
numeric_index=sapply(train,is.numeric)        #It shows only numeric value.
numeric_index
numeric_data=train[,numeric_index]            #Display only numeric data.
numeric_data
cnames=colnames(numeric_data)                 #Numeric column names are stored in variable cnames.
cnames

#Seperate Categorical Variables :
cat_index=sapply(train,is.factor)             #It shows only categorical value.
cat_index
cat_data=train[,cat_index]                    #Display only categorical data.
cat_data
cat_col_names=colnames(cat_data)              #Catgorical column names are stored in variable cat_col_cnames.
cat_col_names

#Now we have two subset of train dataset
# 1. numeric_data which contains only numerical variables data
# 2. cat_data which contains only categorical variables data

#Check Dimension For Actual Dataset
dim(train)                                    #rows=3333  & col=20

#Check Dimension For Numeric Data
dim(numeric_data)                             #rows=3333  & col=15

#Check Dimension For Categorical Data
dim(cat_data)                                 #rows=3333  & col=5


#Churn is our target variable which is categorical
#Let's check out if there exits target imbalance class problem

imbal_class=table(train$Churn)
imbal_class                                   #False : 2850 $ True : 483
barplot(table(train$Churn))
#It can be seen here that 2850 elements are False and 483 elements are True
#Thus we can say that target imbalance problem exists here
#---------------------------------------------------------------------

#calculating event rate
483/3333                                      #14.49 %

#----------------------------------------------------------------------
#To solve these imbalance problem we used here simple sampling method
set.seed(1234)
sample_set =train[sample(nrow(train),size=2333,replace = TRUE),]
sample_set

barplot(table(sample(nrow(train),size=2333,replace=TRUE)))
#----------------------------------------------------------------------
#calculate number of observations in the sample
(3333*70)/100                                 #2333

#After Applying Sampling Method The Result Will Be Given:
imbal_class=table(sample_set$Churn)
imbal_class                                   #So,Sample set contain FALSE : 1998 & TRUE : 335
barplot(table(sample_set$Churn))
#---------------------------------------------------------------------
#Calculate Churn event rate
335/2333                                      #14.35%

#14.35 % rate of churning on a sample set of 2333 observations
#Now from here onwards we are having sample_set as our dataset to apply further processes over it.
```

```
#Onwards We are Using sample_set:
#check The sample_set column names
names(sample_set)

#Seperate The Numeric & categorical variables In sample_set Dataset

#Seperate Numerical Variables:
sample_numeric_index=sapply(sample_set,is.numeric)          #It shows only numeric value.
sample_numeric_index
sample_numeric_data=sample_set[,sample_numeric_index]       #Display only numeric data.
sample_numeric_data
cnames=colnames(sample_numeric_data)                        #Numeric column names are stored in variable cnam
cnames

#Seperate Categorical Variables:
sample_cat_index=sapply(sample_set,is.factor)               #It shows only categorical value.
sample_cat_index
sample_cat_data=sample_set[,sample_cat_index]               #Display only categorical data.
sample_cat_data
sample_cat_col_names=colnames(sample_cat_data)              #Catgorical column names are stores in variable s
sample_cat_col_names

#Check Dimension Of Numerical Variables
 dim(sample_numeric_data)                                   #rows : 2333 & col : 15

#Check Dimension of Categorical Variables
 dim(sample_cat_data)                                       #rows : 2333 & col : 5
```

```
#Bulid the Other Dataset For Performing Operations
set.seed(1234)
new_set =sample_set[sample(nrow(sample_set),size=1000,replace = TRUE),]
new_set

#Dimension of New Dataset i.e new_set
dim(new_set)                                     #rows : 1000 & col : 20

#Check The Column Name
names(new_set)

#For Outlier Analysis We don't need of categorical variables
#as well some numerical variable so we drop that variable.

#First Drop state:
state_index=match("state",names(new_set))
state_index
new_set=new_set[-state_index]

#Second Drop area_code:
area_code_index = match("area.code",names(new_set))
area_code_index
new_set=new_set[-area_code_index]

#Third Drop phone_number:
phone_number_index=match("phone.number",names(new_set))
phone_number_index
new_set=new_set[-phone_number_index]

#Fourth Drop international_plan:
international_plan_index=match("international.plan",names(new_set))
international_plan_index
new_set=new_set[-international_plan_index]
```

```
#Fifth Drop voice_mail_plan:
voice_mail_plan_index=match("voice.mail.plan",names(new_set))
voice_mail_plan_index
new_set=new_set[-voice_mail_plan_index]

#Sixth Drop number.customer.service.calls:
number_customer_service_calls_index= match("number.customer.service.calls",names(new_set))
number_customer_service_calls_index
new_set=new_set[-number_customer_service_calls_index]

#After Drop The variables Check the columns in new_Set
names(new_set)

#Then Seperate Numerical & categorical Variables:

#Seperate Numerical Variables:
new_numeric_index=sapply(new_set,is.numeric)            #It shows only numeric value.
new_numeric_index
new_numeric_data=new_set[,new_numeric_index]            #Display only numeric data.
new_numeric_data
new_cnames=colnames(new_numeric_data)                   #Numeric column names are stored in variable new_cnames
new_cnames

#Seperate Categorical Variables:
new_cat_index=sapply(new_set,is.factor)                 #It shows only categorical value.
new_cat_index
new_cat_data=new_set[,new_cat_index]                    #Display only categorical data.
new_cat_data
new_cat_col_names=colnames(new_cat_data)                #Catgorical column names are stores in variable new_cat
new_cat_col_names
```

```
################Check For Missing Values Column Wise###############

sum(is.na(new_set))        # no any missing value is available

#Plotting the Boxplot
install.packages("caret")
library(caret)

for(i in 1:length(new_cnames))
{
  assign(paste0("bx",i),ggplot(aes_string(y=(new_cnames[i]),x="Churn"),data=subset(new_set))+
           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=new_cnames[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",new_cnames[i])))
  print(i)
}

#Plotting Plots Together
gridExtra::grid.arrange(bx10,bx4,ncol=2)
gridExtra::grid.arrange(bx1,bx2,ncol=2)
gridExtra::grid.arrange(bx8,bx7,ncol=2)
gridExtra::grid.arrange(bx5,bx3,bx11,ncol=3)
gridExtra::grid.arrange(bx12,bx13,ncol=2)
gridExtra::grid.arrange(bx9,bx6,ncol=2)

#Make a backup dataframe
df=new_set
dim(new_set)                                              #rows : 1000 & col : 14
dim(df)                                                   #rows : 1000 & col : 14
```

```
#Detect total number of outliers present in all of the numerical variables

for(i in new_cnames)
{
  print(i)
  val=new_set[,i][new_set[,i] %in% boxplot.stats(new_set[,i])$out]   #0
  new_set=new_set[which(!new_set[,i]%in% val),]                #889
}

#Check missing values
outliers= apply(new_set, 2, function(x) sum(is.na(x)))
outliers
sum(is.na(new_set))                                            # sum(is.na(new_set)) = 0

#Replace all outliers with NA and then impute

for(i in new_cnames)
{
  val=new_set[,i][new_set[,i] %in% boxplot.stats(new_set[,i])$out]
  new_set[,i][new_set[,i] %in% val]=NA
}

#After applying imputation check outliers
#Check Outliers
outliers= apply(new_set, 2, function(x) sum(is.na(x)))
outliers

sum(is.na(new_set))                                            # 218
```

```
#For Removing Outliers we used imputed methods:

# 1.KNN imputation
install.packages("DMwR")
library(DMwR)
install.packages("VIM")
library("VIM")
new_set=kNN(new_set, k=3)
sum(is.na(new_set))

# 2.Mean
new_set_mean=mean(new_cnames)
sum(is.na(new_set_mean))

# 3.Median
new_set_median=median(new_cnames)
sum(is.na(new_set_median))

# 4.Mode                                       #Mode is apply on categorical variable
new_set_mode=mode(new_cat_data)
sum(is.na(new_set_mode))
```

```r
##########################Feature Selection##############################

#Feature Selection is apply on sample_set data
#Feature Selection is solve by correlation as well as Chi-square Test.Correlation is apply only on Numeric variable
# & Chi-square Test is apply on categorical variables.

#Load the Package:
install.packages("corrgram")
library(corrgram)

#Correlation Plot
corrgram(sample_set[,sample_numeric_index],order=F,
         upper.panel = panel.pie ,text.panel = panel.txt,method="color",main="Correlation Plot")


#Chi-Square Test For Categorical Variable
factor_index=sapply(sample_set, is.factor)            #It shows categorical variable.
factor_index

factor_data=sample_set[,factor_index]                 #It shows Categorical Data.
factor_data

factor_cnames=colnames(factor_data)                   #It shows Categorical Column Names.
factor_cnames

#Here 5 are the categorical variables.
#(independent variable:4 & dependent variable:1)
for(i in 1:4)
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}
#Dimension Detection
train_deleted=subset(sample_set,select=-c(phone.number))

############################Decision Tree##################################
#For Decision Tree we used sample_set dataset.

#For Decision Tree model we need to load C50 library
install.packages("C50")
library(C50)

#Develop C50 model on sample_set dataset
C50_model=C5.0(Churn ~number.vmail.messages + 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 +
                 total.intl.minutes + total.intl.calls +total.intl.charge  , data = sample_set)
summary(C50_model)

#For prediction we need test dataset.
#Load the test dataset
test=read.csv("Test_data.csv",encoding = 'ISO-8859-1')

#Predict the data for test dataset
C50_predictors=predict(C50_model,newdata = test,type="class")
C50_predictors

library(caret)
#Calculate the Performance of Model using C50 model

confMatrix_C50=table(test$Churn,C50_predictors)    #Accuracy = 90.22%
confusionMatrix(confMatrix_C50)

#Calculate False Negative Rate
FNR=FN/(FN+TP)         #142/142+82 = 63.39%

#Plot the decision tree
plot(C50_model, main = "Classification Tree")
```

```r
#For Binary Tree model we need to load rpart library
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)

#Develop CART model on sample_set dataset
cart_model = rpart(Churn ~number.vmail.messages + total.day.minutes + total.day.calls + total.day.charge
summary(cart_model)

#For prediction we need test dataset.
#Load the test dataset
test=read.csv("Test_data.csv",encoding = 'ISO-8859-1')

#Predict the data for test dataset
cart_predictors=predict(cart_model,newdata = test,type="class")
cart_predictors

#Calculate the Performance of Model using CART model
confMatrix_Cart=table(test$Churn,cart_predictors)     #Accuracy = 90.04%
confusionMatrix(confMatrix_Cart)

#Calculate False Negative Rate
FNR=FN/(FN+TP)         #151/151+73 = 67.41%

#Plot the tree using prp command defined in rpart.plot package
prp(cart_model, main = "Binary Tree")

#Decision Tree Using C50 Model
 # Accuracy : 90.22%
 # FNR : 63.39%

#Decision Tree Using CART Model
 # Accuracy : 90.04%
 # FNR : 67.41%
```

```r
############################Random Forest#####################################
#For Random Forest we used sample_set Datasets.

#Load The Package
library(randomForest)

#Create the Forest
RF_model<-randomForest(Churn ~ number.vmail.messages + total.day.minutes + total.day.calls + total.day.charge

# View the forest results.
print(RF_model)

# Importance of each predictor.
out.importance <- round(importance(RF_model), 2)
print(out.importance )

#Predict the data for test dataset
randomForest_predictors=predict(RF_model,newdata = test,type="class")
randomForest_predictors

#Calculate the Performance of Model

confMatrix_randomForest=table(test$Churn,randomForest_predictors)     #Accuracy = 90.7%
confusionMatrix(confMatrix_randomForest)

#Calculate False Negative Rate
FNR=FN/(FN+TP)         #150/150+74 = 66.96%

#Plot the Random Forest
plot(RF_model , main = "Random Forest")

#Using Random Forest:
# Accuracy : 90.7%
# FNR : 66.96%
```

```r
###############Logistic Regression###################
#For Logistic Regression We used sample_set dataset.
#Logistic model is apply on categorical data.
require(ISLR)
#Create a Logistic Model
logit_model<-glm(Churn ~state + international.plan + voice.mail.plan, data=sample_set ,family = binomial)
summary(logit_model)

#Predict the dataset for test data
logit_predictions=predict(logit_model,newdata = test,type="response")

#Convert into probabilities
logit_predictions=ifelse(logit_predictions>0.5, 1,0)
logit_predictions

#Calculate the performance of the model
confMatrix_logit=table(test$Churn,logit_predictions)  #Accuracy = (1409 + 32)/1667 = 86.44%
confMatrix_logit

#Calculate False Negative Rate
FNR=FN/(FN+TP)        #192/(192+32) = 85.71%

#Plot the Logistic Regression
plot(logit_model)

#Using Logistic  Regression:
# Accuracy : 86.44%
# FNR : 85.71%

#Comparing within three models like Decision Tree,Random Forest and Logistic Regression,
#Three out of two model give the average result.
#Therefore we can select either of two models without loss of information.
```