Project Name
Churn reduction

Niket Ramesh Patil

12/3/2019

Contents

Chapter 1
Introduction

1.1 Problem Statement

Loss of customers to competition i.e. churn 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. The objective of this Case is to predict customer behavior.

1.2 Data

Here we have a public dataset in train and test that has customer usage pattern and if the customer has moved or not. Here we will develop an algorithm to predict the churn score based on usage pattern. The predictors provided are as follows:

- Account length
- State
- Area code
- Phone number
- 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
- Churn

Our aim is to develop a classification model which will predict if the customer left our service or not i.e. **he will move or not**.

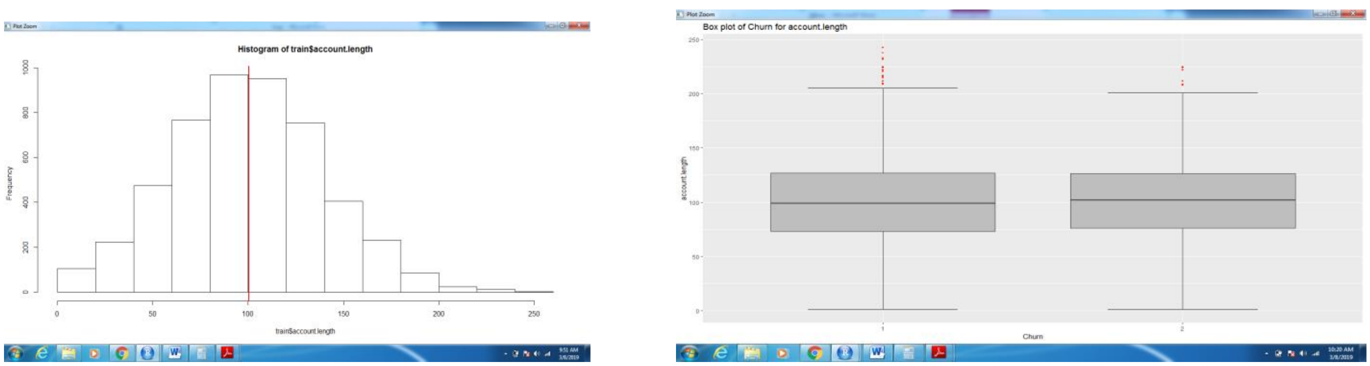
Chapter 2

Methodology

Here we have data set in the form of train and test. So, here we can work on data by two ways. 1st is we can apply preprocessing separately on both train and test data set and then apply machine learning on a top of it and 2nd is we simple merge both the data in one which contain information of both train and test data set and then we will go for the preprocessing and finally by 80 20 pattern we will apply machine learning on a top of it. So, here we have selected 2nd way to merge data into **New_train.csv** file.

2.1 Pre Processing

We will check the normality of each independent variable before applying the preprocessing on the top of the data. As we applied normality on the top of the New_train data set we come to know most of our data is normally distributed.



Normality and outlier output of account length before removing outlier

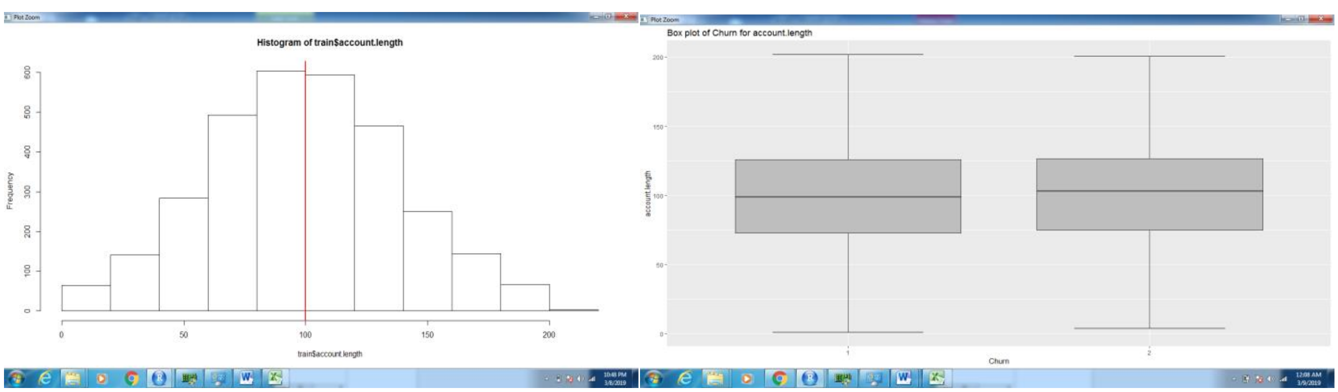


Fig 2.1 Normality and outlier output of account length after removing outlier

As we can see a lot of useful inferences can be made from these plots. First we have a lot of outliers and extreme values in each of the data set and our data is not completely normality distributed. When we applied the outlier on the top of the data we were successfully removed the most of them and from above we come to know most of our data is normally distributed. **Fig 2.1** tells us normality and outlier output of account length before and after removing outlier. We plotted rest of the predictors at end of the report.

2.1.2 Code to check normality and outliers:

- **Histogram**

```
mx = mean(bf$total.day.minutes)

hist(bf$total.day.minutes)

 abline(v = mx, col = "blue", lwd = 2)
```

- **Normalization**

```
numeric_index = sapply(bf,is.numeric)

numeric_data = train[,numeric_index]

cnames = colnames(numeric_data)

for (i in 1:length(cnames))

{

  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"), data = subset(bf))+

           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=cnames[i],x="Churn")+

           ggtitle(paste("Box plot of Churn for",cnames[i])))

}

gridExtra::grid.arrange(gn2,ncol=1)
```

2.1.1 Missing Value Analysis

When we applied missing value algorithm on the top of the data we come know the train data don't have any missing value. So, we will go for the further pre-processing techniques.

2.1.2 Boxplot Method

As we can see there is a plot **Fig.2.2** of independent variable verses target variable which has yes and no category. The red dots above and the below upper and lower phase which consider as outlier and we have to remove it by boxplot.stats() $out or replace with NA method . In this we applied boxplot code on the top of the data.

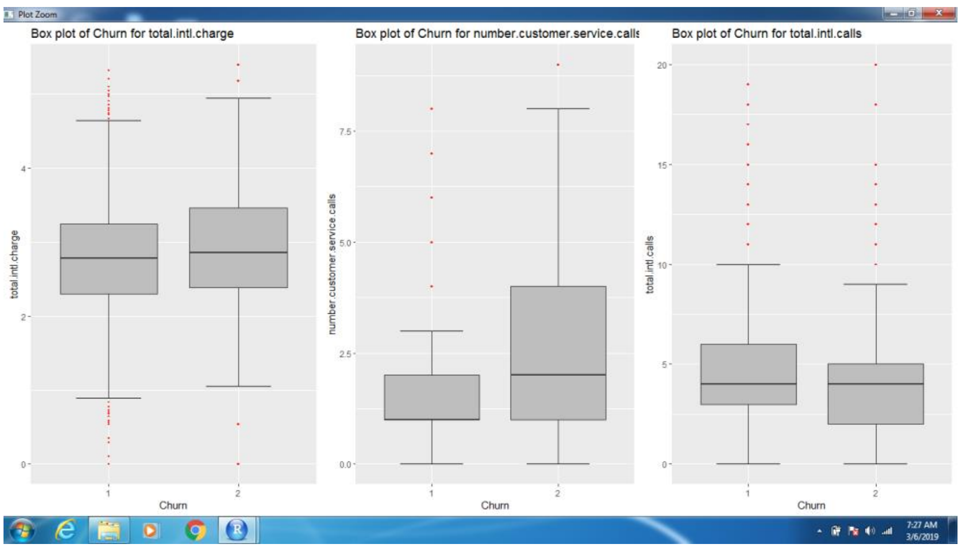


Fig.2.2 boxplot of predictors before removing outliers

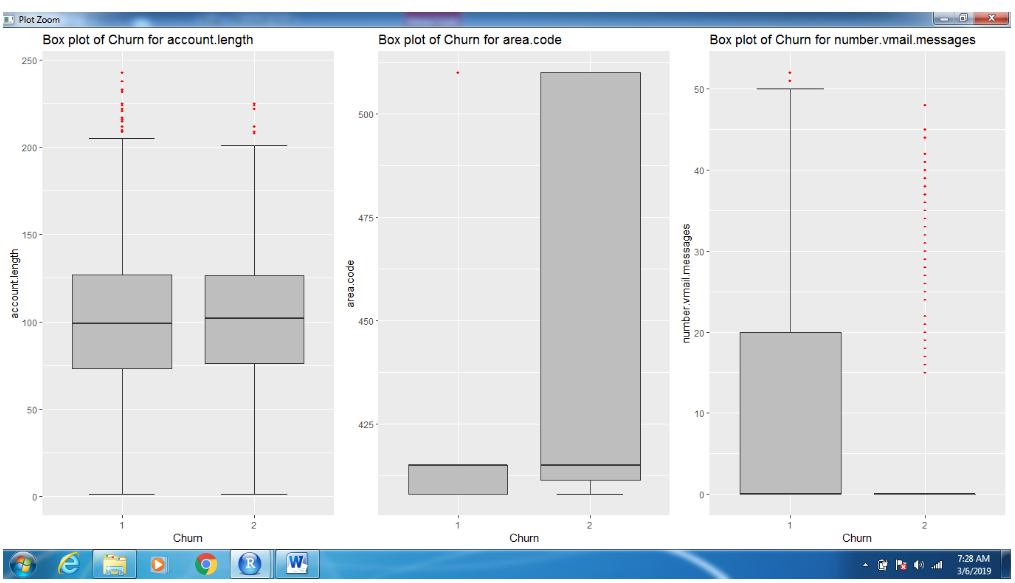


Fig.2.2 boxplot of predictors before removing outlier

2.1.3 Feature Selection

- **Correlation Plot on continuous variables**

As we know the dependency between independent and dependent variable should be high & the dependency between two independent variables should be low. So, on the behalf of same below we plotted a **correlation plot** that help us to identify which numeric or continues independent variable do not carries much information to explain the target variable. The **fig.2.3** depicts that we can consider all those features. We suppose to remove that variable, if it is positively correlated with one variable and negatively correlated with the other variable or if same variable is positively correlated with 2 or more variables, then we can remove that variable. So as per our below **Fig.2.3** here we are considering all those variables

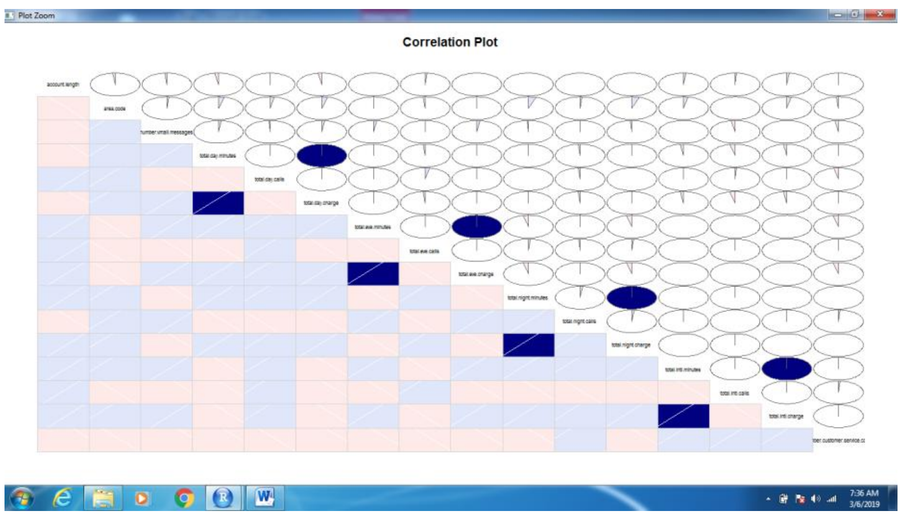


Fig.2.3 Correlation Plot to identify which variables are not carries much information

- **chi-square test (X^2)**

We will apply the chi-square test on categorical intendant variable for that we have to b uild the confusion matrix to find P value. If P value is less than 0.05 we select the null hypothesis else drop those variable (P>0.05) as shown below. P value of phone.number is NA as shown belo w.

$$\mathbf{X^2 = \sum (O - E)^2/E}$$ (where O is observed and E is expected value)

We can't conclude anything if result is NA i.e. X-squared is NA, the reason might be while computing if the denominator is zero it would result NA, thereby p value will be NA i.e. infinity. P should have some value, and then only we can get information out of it and it is possible if 0 frequencies is less than 20%. So, as per above methodology we drop this observation

[1] "state"

Pearson's Chi-squared test

data: table(factor_data$Churn, factor_data[, i])
X-squared = 72.847, df = 50, **p-value = 0.01913** # **P<0.05 it carries the information to explain target variable so will select it for further operation**

[1] "phone.number"

Pearson's Chi-squared test

data: table(factor_data$Churn, factor_data[, i])
X-squared = NaN, df = 4999, **p-value = NA** # **P =The denominator is 0 so that it gives NA. NA means it do not carry any meaningful information**

[1] "international.plan"

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

data: table(factor_data$Churn, factor_data[, i])
X-squared = 242.1, df = 1, **p-value < 2.2e-16** # **p<0.05**

[1] "voice.mail.plan"

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

data: table(factor_data$Churn, factor_data[, i])
X-squared = 35.574, df = 1, **p-value = 2.455e-09** # **p<0.05**

2.1.4 Feature Scaling

In feature selection pre-processing we plot the histogram to check the normality of cont inuous independent variables. If the variables are normally distributed then we go for standardiza tion else for normalization. In **Fig.2.4** we find the histogram of multiple continuous independent variables and most of them are normally distributed so we will go for standardization. The output of standardization will tell us how much Actual value is deviate from the mean

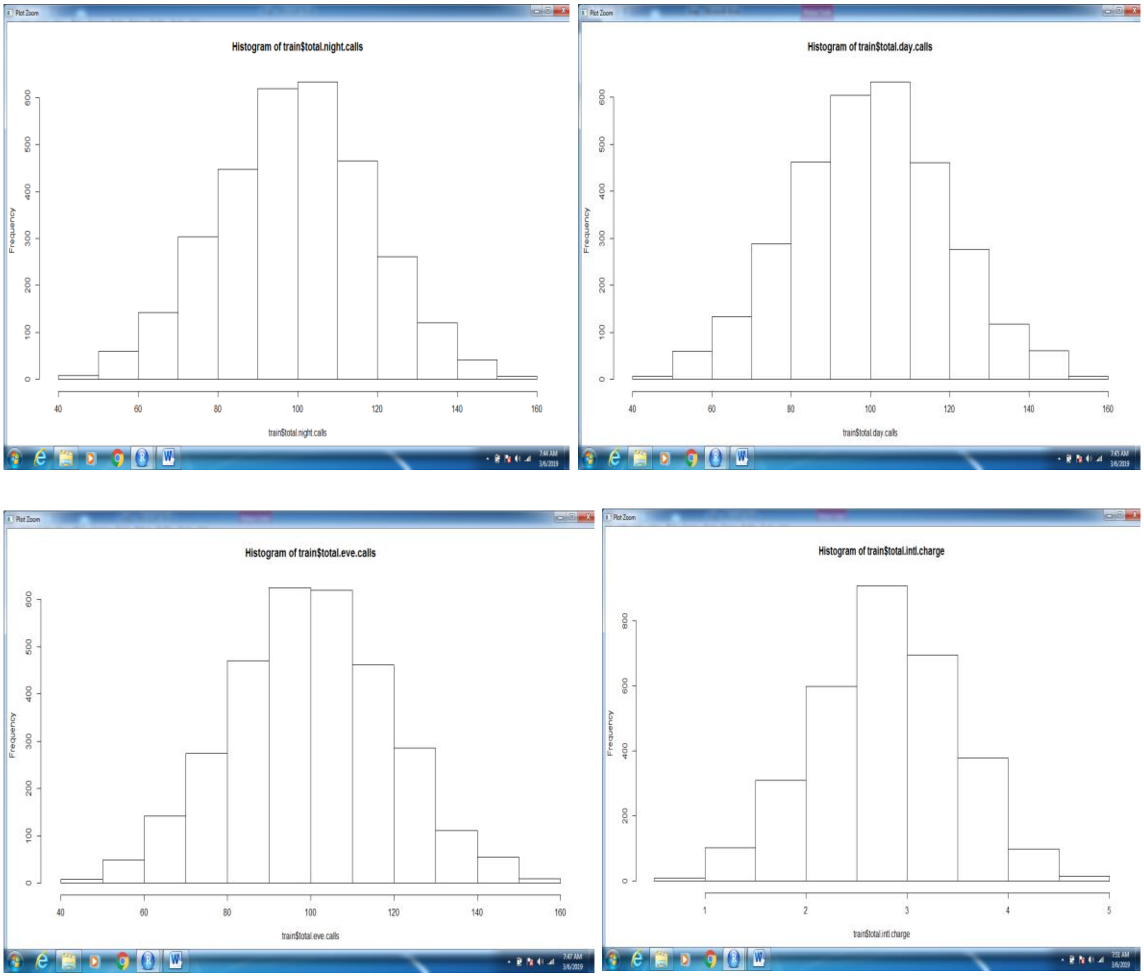


Fig.2.4 Normality Plot

2.2 Machine Learning Algorithm

As we know our train data has a target variable i.e. churn. So here we will apply supervise machine learning algorithm on the top of **New_train** data like decision tree, random forest, logistic regression, KNN, naïve Bayes. Every machine learning algorithm will give us different values of error matrix but here we will focus on **accuracy** and **FNR** of the model.

2.2.1 Decision Tree:

As we know our target variable is categorical so that we will apply decision tree for classification model. 1st we have to differentiate data into train and test 80:20 patterns and then apply C5.0 decision tree algorithm on the train data set. We will find multiple rule in which **lift** should be greater than **1**, **confidence** should be greater than **80%** and **support** should be greater than **20%** in below output **lift=1.4**, **confidence = 98.9%** so it carries meaningful important information to explain the target variable

Rule 73/2: (90.1, **lift 1.4**)
international.plan = 1
total.day.charge <= 0.7953073
total.eve.calls > -1.35342
total.eve.calls <= -0.8852951
-> class 1 [**0.989**]

Then we will apply the confusion matrix on the top of the data set and the output is below witch has multiple error matrix

Confusion Matrix and Statistics

```
C50_Predictions
     1   2
  1 555   1
  2  22  43
```

```
            Accuracy : 0.963
              95% CI : (0.9449, 0.9764)
 No Information Rate : 0.9291
 P-Value [Acc > NIR] : 0.0002566

               Kappa : 0.7695
Mcnemar's Test P-Value : 3.042e-05

         Sensitivity : 0.9619
         Specificity : 0.9773
      Pos Pred Value : 0.9982
      Neg Pred Value : 0.6615
          Prevalence : 0.9291
      Detection Rate : 0.8937
Detection Prevalence : 0.8953
   Balanced Accuracy : 0.9696
```

Conclusion = In the decision tree algorithm when we applied the predicted model of train data o n the top of test data set to predict variations in the target variable. We find multiple error matrix es but as we know our aim is to calculate the churn (loss of customer to campaign) so, here we c onsider accuracy and FNR as shown below

Accuracy = 96.3%

FNR = FN/FN+TP
= 33.84

From above predicted value we come to know the accuracy of the DT is good and it is acceptable and FNR is **33.84.** So, we will focus on another ML algorithm to improve our results more batter .

2.2.2 Random Forest:

As we know random forest works on collection of DT so, from our Train data set we here we bui ld 40 DT. Hence we come to know the RF is working batter than decision tree algorithm. FNR of RF is quit less as compare to DT FNR as shown below.

Confusion Matrix and Statistics

```
  RF_Predictions
     1   2
 1 554   2
 2 24  41

            Accuracy : 0.9581
              95% CI : (0.9393, 0.9725)
 No Information Rate : 0.9308
 P-Value [Acc > NIR] : 0.002802

               Kappa : 0.7374
 Mcnemar's Test P-Value : 3.814e-05

         Sensitivity : 0.9585
         Specificity : 0.9535
      Pos Pred Value : 0.9964
      Neg Pred Value : 0.6308
          Prevalence : 0.9308
      Detection Rate : 0.8921
Detection Prevalence : 0.8953
   Balanced Accuracy : 0.9560
```

Accuracy : 95.81%
FNR : 24/(24+41)
= 36.92

2.2.3 Logistic Regression:

As our target variable is categorical so that we will apply logistic regression on the top of it i nstead of linear regression. As we know our target variable has only two values (class) i.e. yes or no so here we select binomial family if P value is in between 0 to 0.5 we will assign it as class 0 else class 1 for that 1st we need to calculate p value i.e.

$$P = \frac{1}{1 + \mathrm{e}\hat{} - \mathrm{logit(p)}}$$

Logistic regression will calculate regression coefficient of each category present in a categorical variable. In our data set independent variable name as state have multiple categories as shown in below table

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-3.973e+00	1.090e+00	-3.646	0.000267	***
state2	6.289e-01	1.165e+00	0.540	0.589381	
state3	9.935e-01	1.216e+00	0.817	0.413842	
state4	3.762e-01	1.297e+00	0.290	0.771737	
state5	1.161e+00	1.267e+00	0.916	0.359432	
state6	-3.858e-01	1.358e+00	-0.284	0.776314	

……..

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1674.9 on 2487 degrees of freedom
Residual deviance: 1018.3 on 2419 degrees of freedom

- **Difference between Null deviance and Residual deviance should be very high**

- AIC: 1156.3 **(alkaline information criteria)**

Here we are working on the single model so AIC not comes into the picture but if we are workin g on the multiple model and those model have a same accuracy and FNR so, in such situation AI C play an important role to predict which model we should refer with the help of less value of AI C.

logit_Predictions

	0	1
1	544	12
2	42	23

Accuracy: 91.30%
FNR: 64.61

2.2.4 KNN Implementation:

As we know the KNN not store any pattern. It is works on distance formula and it directly calculates the distance between train and test of target data as shown below.

Accuracy: 0.8969404
> Conf_matrix

KNN_Predictions

```
        Predicted
observed   1   2
       1 553  61
       2   3   4
```

> 3/(3+4)
Accuracy : 89.69%
FNR : 42.85

2.2.5 Naive Bayes:

It is also a supervise machine learning algorithm which works on the basis of probability. It consider every single variable as independent variable so, it resolve the problem of multi-collinearty
Confusion Matrix and Statistics

```
        predicted
observed   1   2
       1 538  18
       2  30  35
```

Accuracy : 0.9227
95% CI : (0.8988, 0.9425)
No Information Rate : 0.9147
P-Value [Acc > NIR] : 0.2628

Kappa : 0.551
Mcnemar's Test P-Value : 0.1124

Sensitivity : 0.9472
Specificity : 0.6604
Pos Pred Value : 0.9676
Neg Pred Value : 0.5385
Prevalence : 0.9147

```
      Detection Rate : 0.8663
   Detection Prevalence : 0.8953
      Balanced Accuracy : 0.8038

       'Positive' Class : 1
```

Accuracy: 92.27%
FNR: 46.15

2.2.6 Code for the Best Model i.e. Decision Tree

#Divide data into train and test using stratified sampling method
```
set.seed(1234)
train.index = createDataPartition(train$Churn, p = .80, list = FALSE)# by sampling library
Train = train[ train.index,]
Test  = train[-train.index,]
```

##Decision tree for classification
```
#Develop Model on training data
final_model = C5.0(Churn ~., Train, trials = 50, rules = TRUE)
```

#Summary of DT model
```
summary(final_model)
```

 # save the model to disk
```
saveRDS(final_model, "final_model.rds")
```

 # load the model
```
Best_model = readRDS("final_model.rds")
print(Best_model)
```

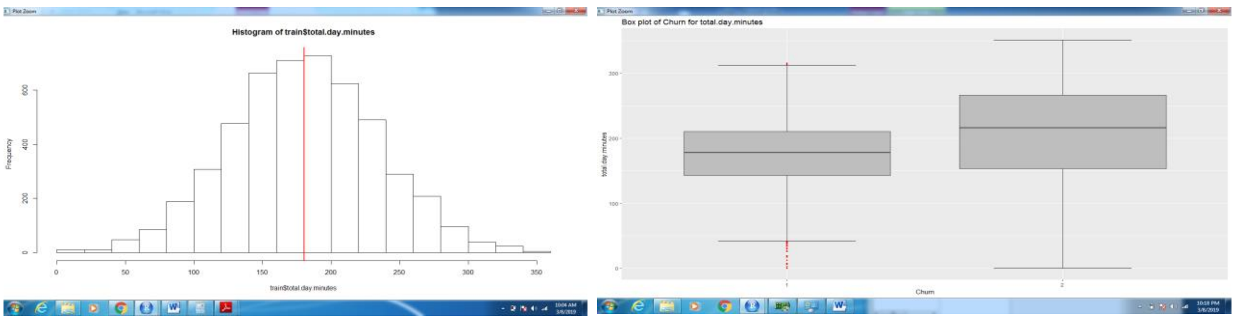
Chapter 3

Conclusion

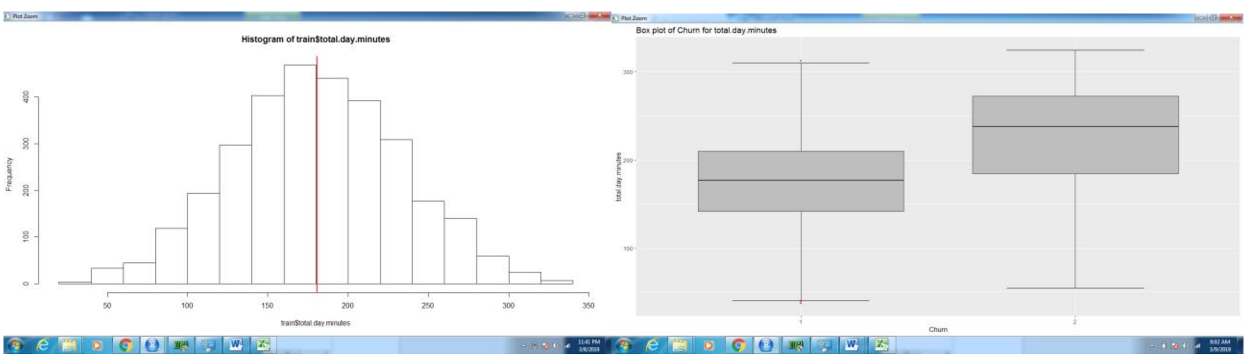
In this project the objective is to predict customer behavior. Here we applied multiple pre-pr ocessing techniques to design our data in proper outline and also activated numerous machines le arning to predict more systematic value of the target variable. So, in the model selection we can s ee the entire model gives us acceptable value of **accuracy** but as we are talking about **FNR we** di scovered that the Decision **Tree** give us best result as compare to other model. So, here we can s elect **Decision Tree** algorithm as our final model to predict customer behavior.

Appendix A

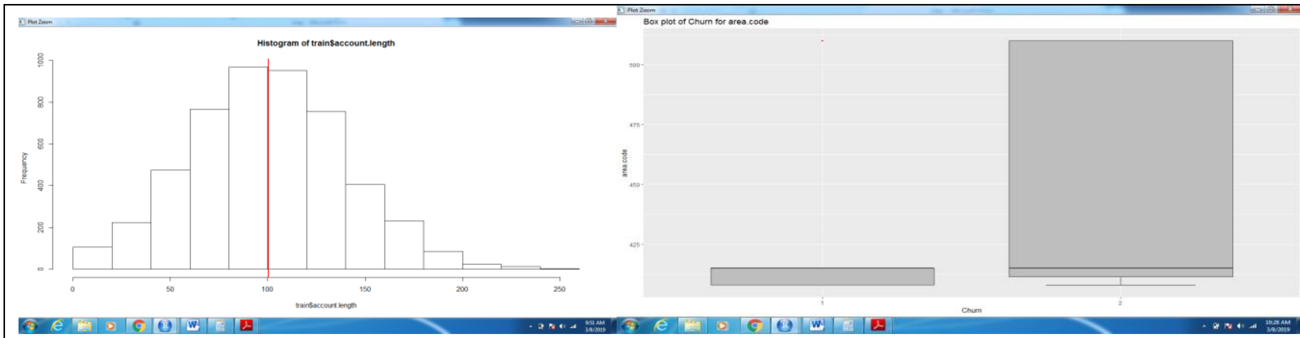
Extra Figures



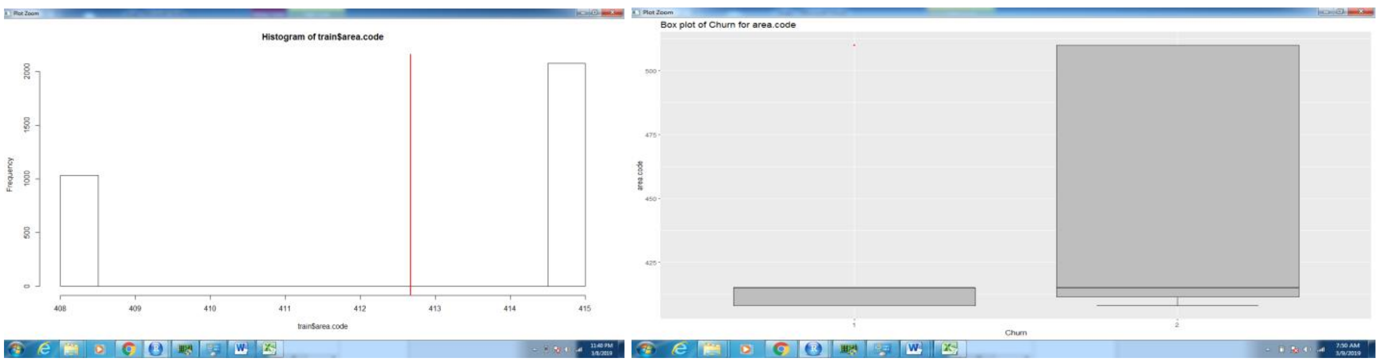
Normality and outlier output of total day minutes before removing outlier



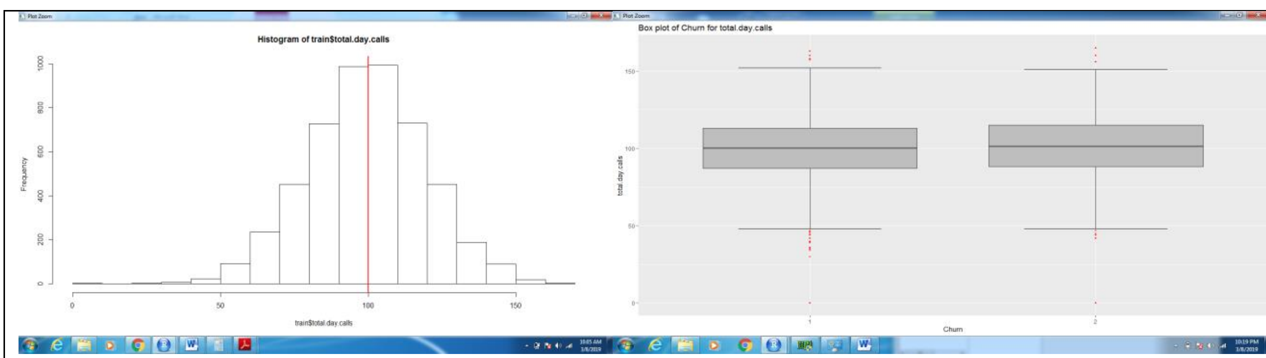
Normality and outlier output of total day minutes after removing outlier



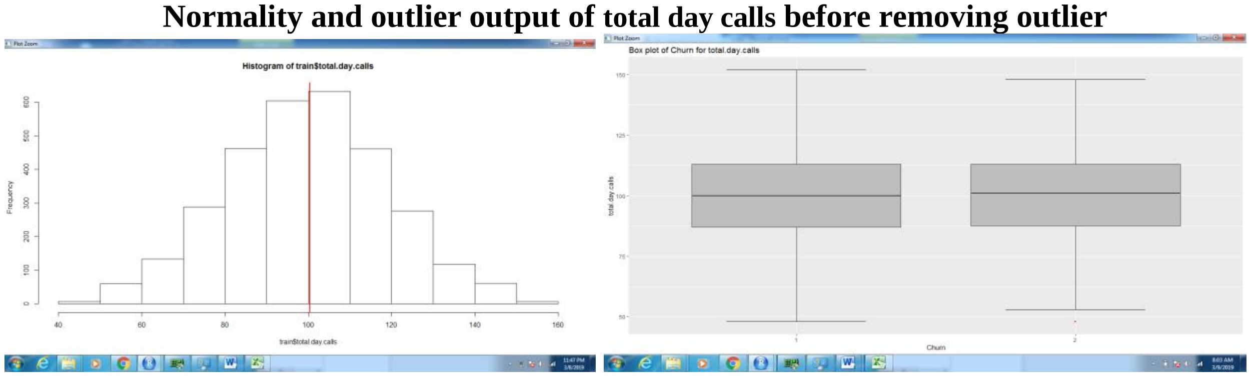
Normality and outlier output of area code before removing outlier



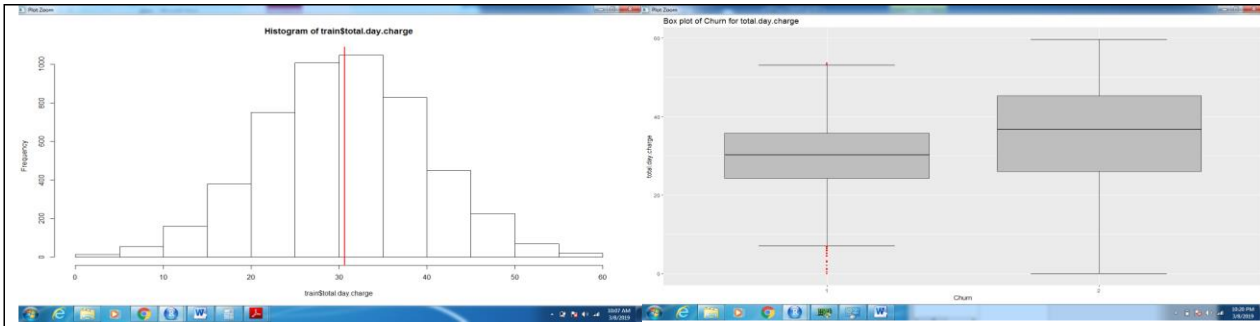
Normality and outlier output of area code After removing outlier



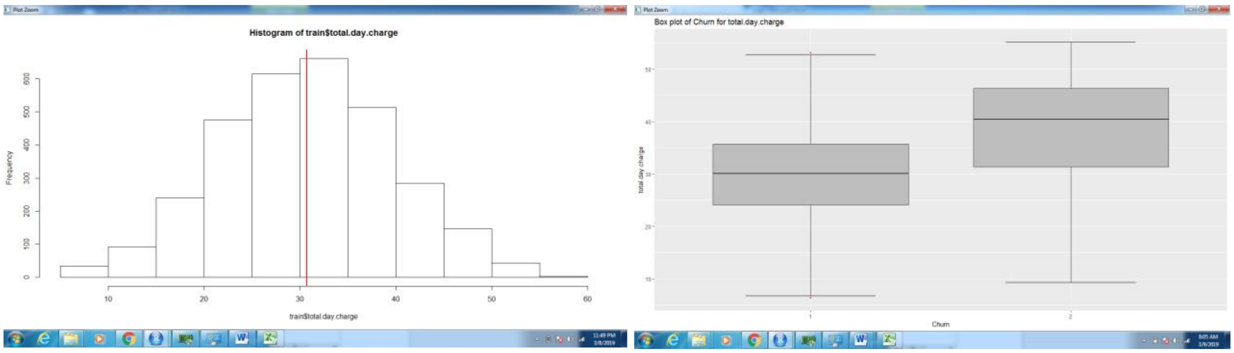
Normality and outlier output of total day calls before removing outlier



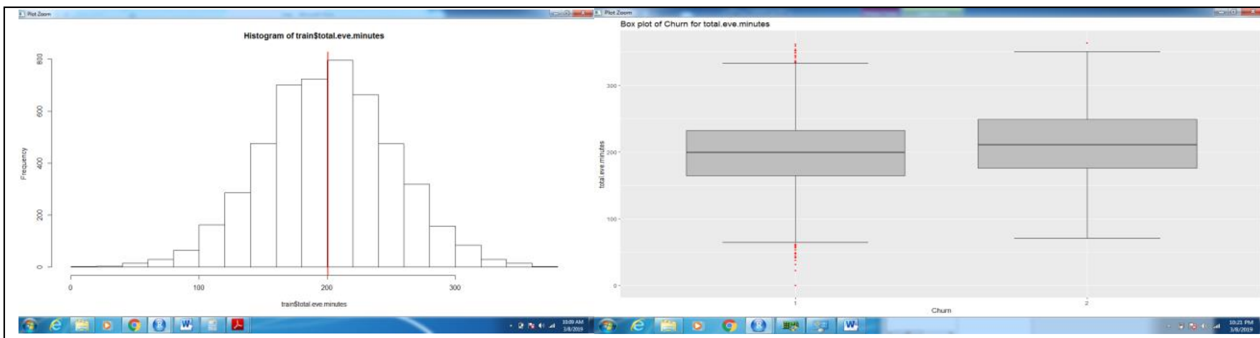
Normality and outlier output of total day calls after removing outlier



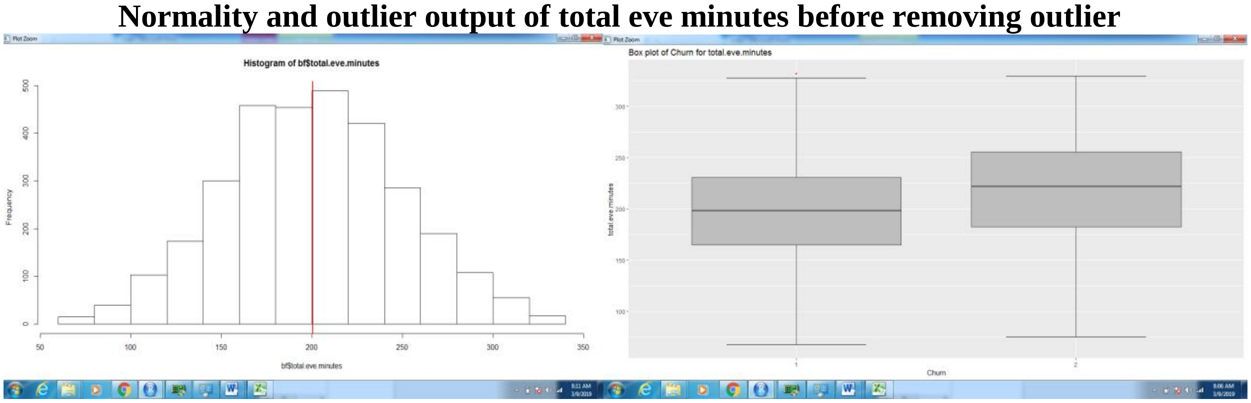
Normality and outlier output of total day charge before removing outlier



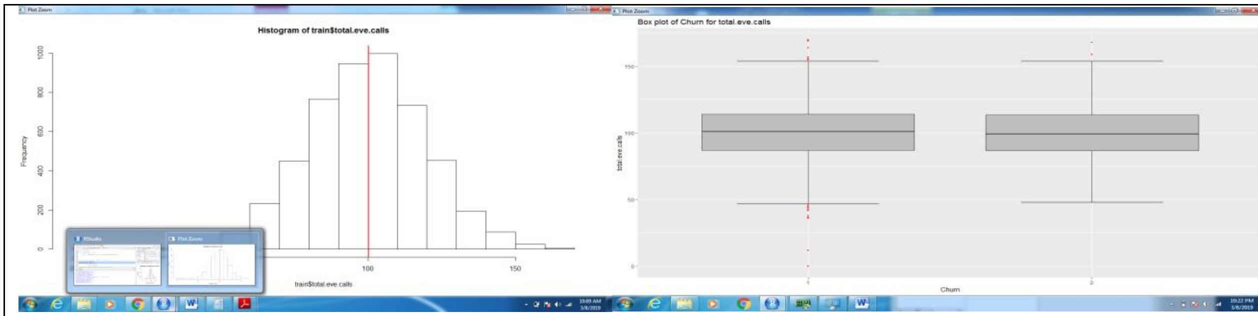
Normality and outlier output of total day charge after removing outlier



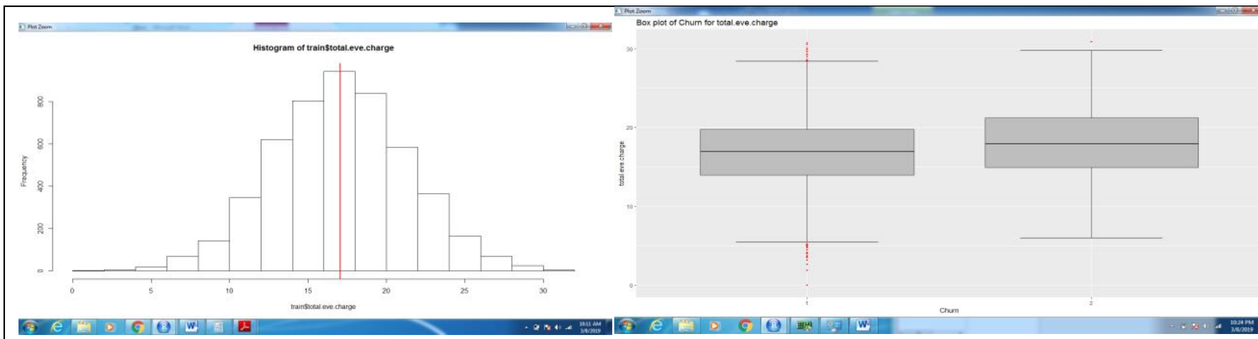
Normality and outlier output of total eve minutes before removing outlier



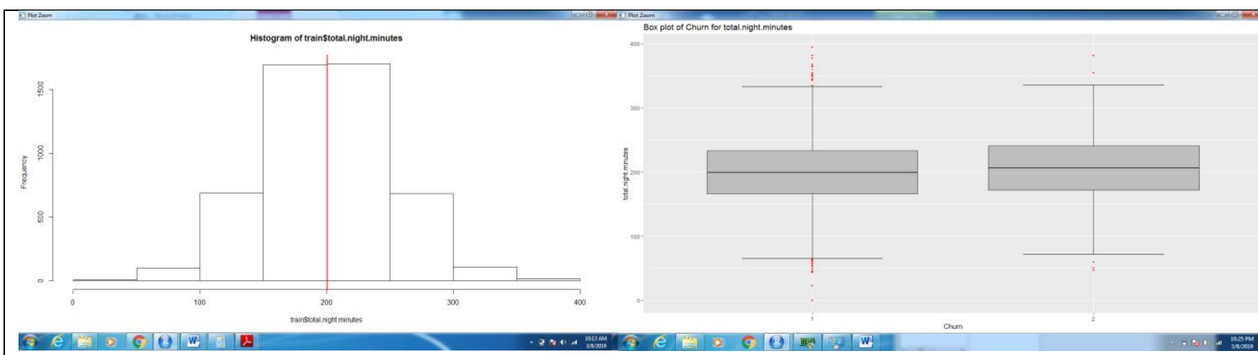
Normality and outlier output of total eve minutes after removing outlier



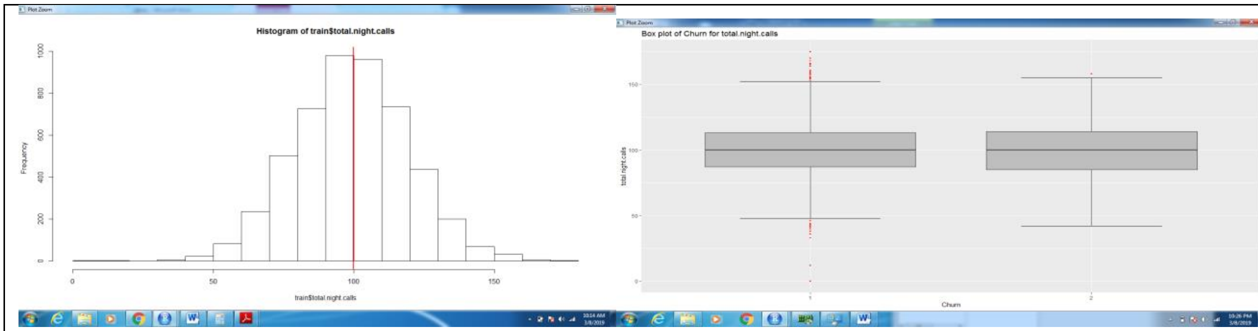
Normality and outlier output of total eve calls before removing outlier



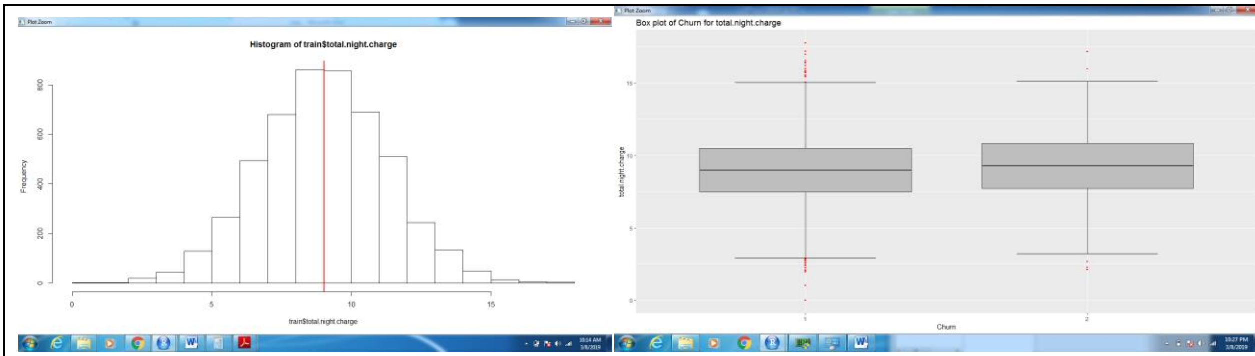
Normality and outlier output of total eve charge before removing outlier



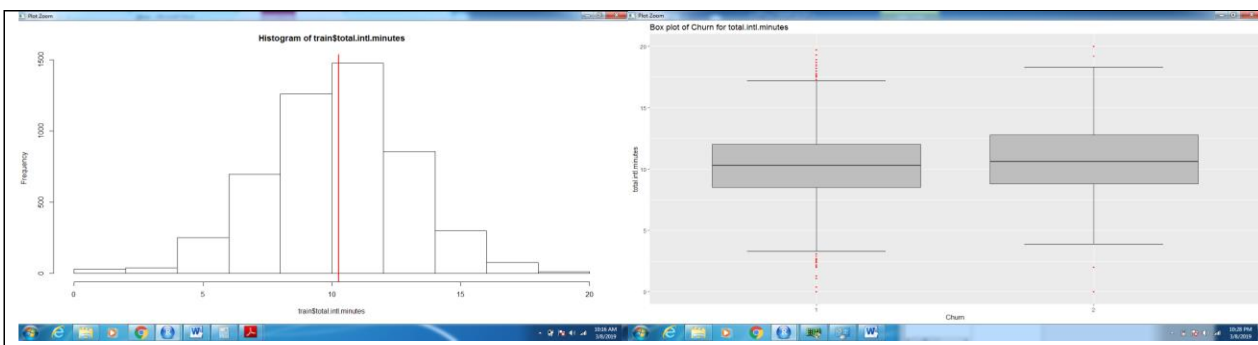
Normality and outlier output of total night minutes before removing outlier



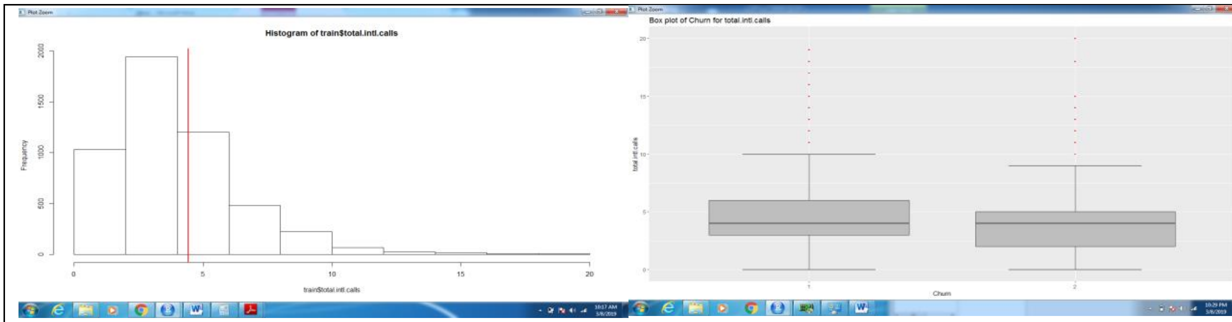
Normality and outlier output of total night calls before removing outlier



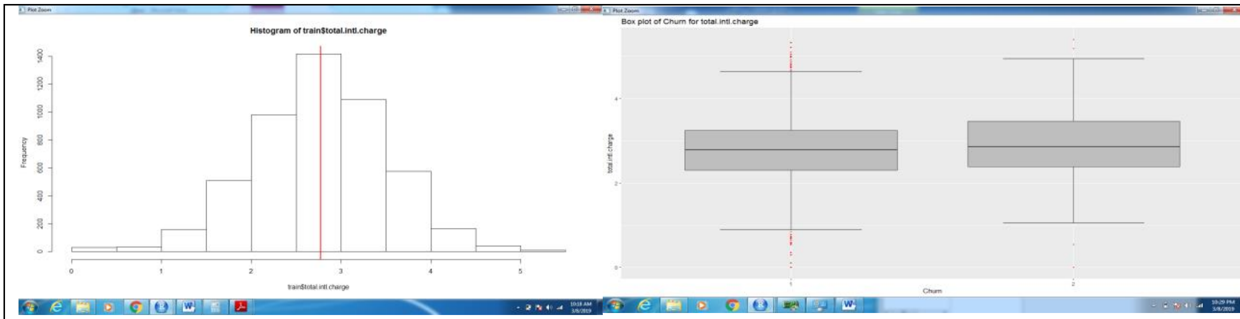
Normality and outlier output of total night charge before removing outlier



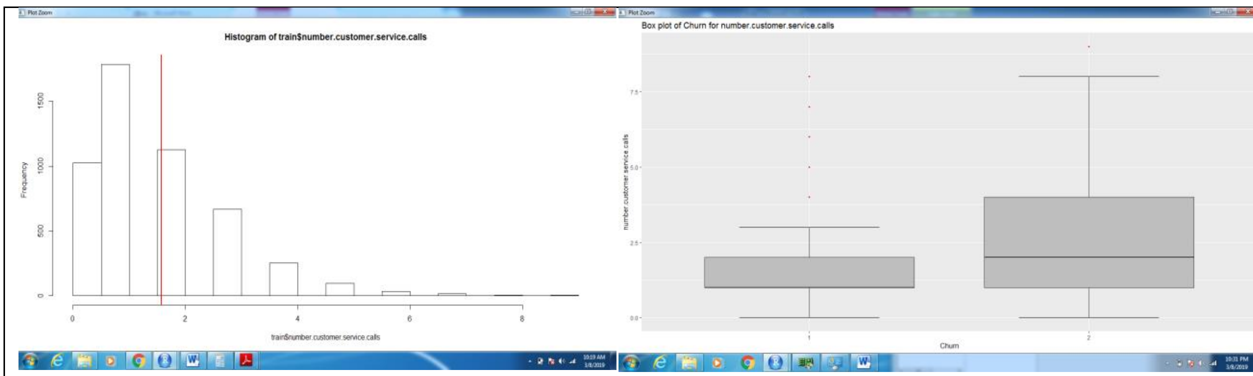
Normality and outlier output of total intl minutes before removing outlier



Normality and outlier output of total intl calls before removing outlier



Normality and outlier output of total intl charge before removing outlier



Normality and outlier output of number customer service calls before removing outlier

Appendix B

1. R Code

```
rm(list=ls())

setwd("C:/Users/User/Desktop/Project 1/Store")
#getwd()

#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

#install.packages(x)
lapply(x, require, character.only = TRUE)


## Read the data
train = read.csv("New_train.csv")

## As per the misssing value analysis process we come know there is no any missing value
in above data set
missing_val = data.frame(apply(train,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] =  "Missing_percentage"
missing_val$train = (missing_val$Missing_percentage/nrow(train)) * 100
missing_val = missing_val[order(-missing_val$Missing_percentage),]
row.names(missing_val) = NULL
missing_val = missing_val[,c(2,1)]


##Data Manupulation; convert string categories into factor numeric

for(i in 1:ncol(train)){

  if(class(train[,i]) == 'factor'){

      train[,i] = factor(train[,i], labels=(1:length(levels(factor(train[,i])))))

  }
}
```

Outlies analysis

numeric_index = sapply(train,is.numeric) #as we know the outlier are only applied on numeric data

numeric_data = train[,numeric_index]

cnames = colnames(numeric_data)

```
 for (i in 1:length(cnames))
 {
   assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"), data = subset(train))+
           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=cnames[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",cnames[i])))
 }
```

Plotting plots together

gridExtra::grid.arrange(gn1,ncol=1)

df = train # store data before removeing outlier.

 # #Remove outliers using boxplot method

```
#loop to remove from all variables
 for(i in cnames){

   val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
   print(length(val))
   train = train[which(!train[,i] %in% val),]
 }
```

bf= train # store data of outlier analysis

Feature Selection

```
## Correlation Plot
corrgram(train[,numeric_index], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
```

```
## Chi-squared Test of Independence
factor_index = sapply(train,is.factor)
factor_data = train[,factor_index]

for (i in 1:4)
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn ,factor_data[,i]))) # there should be high dependancy between target and indedant veriable
}

## Dimension Reduction
train = subset(train, select = -c(phone.number))
gf = train

## feature scaling
#Normality check


#Histograme
mx = mean(train$total.day.charge)
hist(train$total.day.charge)
abline(v = mx, col = "Red", lwd = 2)

#Normalisation
cnames =
c("account.length","area.code","total.day.calls","total.day.charge","number.vmail.messages",
          "total.eve.calls","total.eve.charge","total.night.calls","total.night.charge",
          "total.intl.calls","total.intl.charge","number.customer.service.calls",
          "total.day.minutes","total.eve.minutes","total.night.minutes","total.intl.minutes")

# #Standardisation
 for(i in cnames){
   print(i)
   train[,i] = (train[,i] - mean(train[,i]))/ sd(train[,i])
 }

pf= train # output train data set


##################################Model Development##########################

#Clean the environment
library(DataCombine) # it is use to remove all the files except the original one
rmExcept("train")
```

#Divide data into train and test using stratified sampling method

```
set.seed(1234)
train.index = createDataPartition(train$Churn, p = .80, list = FALSE)# by sampling library
Train = train[ train.index,]
Test  = train[-train.index,]
```

##Decision tree for classification
#Develop Model on training data

```
C50_model = C5.0(Churn ~., Train, trials = 50, rules = TRUE)
```

#Summary of DT model

```
summary(C50_model)
```

#write rules into disk

#Lets predict for test cases

```
C50_Predictions = predict(C50_model, Test[,-20], type = "class")
```

##Evaluate the performance of classification model

```
ConfMatrix_C50 = table(Test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)
```

#False Negative rate

```
FNR = FN/FN+TP
```

#Accuracy: 96.3%
#FNR:33.84

###Random Forest

```
RF_model = randomForest(Churn ~ ., Train, importance = TRUE, ntree = 60)
```

#Presdict test data using random forest model

```
RF_Predictions = predict(RF_model, Test[,-20])
```

##Evaluate the performance of classification model

```
ConfMatrix_RF = table(Test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
```

```
#False Negative rate
#FNR = FN/FN+TP
```

#Accuracy = 95.81
#FNR = 36.92

#Logistic Regression
logit_model = glm(Churn ~ ., data = Train, family = "binomial") # binomial bcoz our target variable is in form of yes or no if it is more than this we can go for maltinomial

#summary of the model
summary(logit_model)

#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = Test, type = "response") # response in the form of probablity 0 and 1

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model
ConfMatrix_RF = table(Test$Churn, logit_Predictions)

#False Negative rate
#FNR = FN/FN+TP

#Accuracy:91.30
#FNR: 64.61

##KNN Implementation
library(class)

#Predict test data
KNN_Predictions = knn(Train[, 1:19], Test[, 1:19], Train$Churn, k = 7)

#Confusion matrix
Conf_matrix = table(KNN_Predictions, Test$Churn)

#Accuracy
sum(diag(Conf_matrix))/nrow(Test)

#False Negative rate
#FNR = FN/FN+TP

#Accuracy = 89.69
#FNR = 42.85

#naive Bayes
library(e1071)

#Develop model
NB_model = naiveBayes(Churn ~ ., data = Train)

#predict on test cases #raw
NB_Predictions = predict(NB_model, Test[,1:19], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = Test[,20], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)

#Accuracy: 92.47%
#FNR: 46.15

#Here We selet Decision tree as our finel model.
saveRDS(C50_model,"train.rds")

#Best_model = readRDS("train.rds")

2. Python Code

DATA PREPROSSESING

#Load libraries

```
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
from sklearn import model_selection
import pickle
```

#Set working directory

```
os.chdir("C:/Users/User/Desktop/Project 1/Store")
```

#Load data

```
train = pd.read_csv("New_train.csv")
```

Missing Value Analysis

#Create dataframe with missing percentage

```
missing_val = pd.DataFrame(train.isnull().sum())
```

#Reset index

```
missing_val = missing_val.reset_index()
```

#Rename variable

```
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})
```

#Calculate percentage

```
missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(marketing_train))*100
```

#descending order

```
    missing_val = missing_val.sort_values('Missing_percentage', ascending
= False).reset_index(drop = True)
```

###There is no any missing value in our data set###

```
    #Convert into proper datatypes
    for i in lis:
       train.loc[:,i] = train.loc[:,i].round()
       train.loc[:,i] = train.loc[:,i].astype('object')
```

Outlier Analysis"

```
    df = train.copy()
    #train = df.copy()

    # #Plot boxplot to visualize Outliers

     %matplotlib inline
     plt.boxplot(train['custAge'])

    #save numeric names

    cnames =
["area.code","total.day.calls","total.day.charge","number.vmail.messages",
"total.eve.calls","total.eve.charge","total.night.calls","total.night.char
ge","total.intl.calls","total.intl.charge","number.customer.service.calls"
,"total.day.minutes","total.eve.minutes","total.night.minutes","total.intl
.minutes",“account length”]

       #Detect and delete outliers from data
     for i in cnames:
         print(i)
         q75, q25 = np.percentile(train.loc[:,i], [75 ,25])
         iqr = q75 - q25

         min = q25 - (iqr*1.5)
         max = q75 + (iqr*1.5)
         print(min)
         print(max)

train = train.drop(train[train.loc[:,i] < min].index)
train = train.drop(train[train.loc[:,i] > max].index)

    #Detect and replace with NA

    #Extract quartiles
     q75, q25 = np.percentile(train[' total.eve.minutes '], [75 ,25])

    #Calculate IQR
     iqr = q75 - q25

    #Calculate inner and outer fence
     minimum = q25 - (iqr*1.5)
```

```
 maximum = q75 + (iqr*1.5)

#Replace with NA
 train.loc[train[' total.eve.minutes '] < minimum,:'custAge'] = np.nan
 train.loc[train[' total.eve.minutes '] > maximum,:'custAge'] = np.nan

#Calculate missing value
missing_val = pd.DataFrame(train.isnull().sum())
```

```
#Impute with KNN
   train = pd.DataFrame(KNN(k = 3).complete(train), columns =
train.columns)
```

```
## Feature Selection
```

```
#Correlation analysis
#Correlation plot
df_corr = train.loc[:,cnames]

#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220, 10, as_cmap=True),
                square=True, ax=ax)
```

```
#Chisquare test of independence

#Save categorical variables
 cat_names = ["state"," international plan"," voice mail
plan","Churn"]
```

```
#loop for chi square values
 for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(train['churn'],
train[i]))
    print(p)

 train = train.drop(['phone.number'], axis=1)
```

```
# Feature Scaling
```

```
df = train.copy()
#train = df.copy()
```

#Normality check

```
%matplotlib inline
plt.hist(train['campaign'], bins='auto')
```

#Nomalisation

```
for i in cnames:
    print(i)
```

 # #Standarisation

```
 for i in cnames:
    print(i)
    train[i] = (train[i] - train[i].mean())/train[i].std()


marketing_train = pd.to_csv("marketing_train_Model.csv")
```

. # Model Development
DATA MINEING

#Import Libraries for decision tree

#replace target categories with Yes or No

```
train['churn'] = train['churn'].replace(0, 'No')
train['churn'] = train['churn'].replace(1, 'Yes')
```

#Divide data into train and test

```
X = train.values[:, 0:19]
Y = train.values[:,19]

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size =
0.2)
```

#Decision Tree

```
C50_model =
tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)

#predict new test cases
C50_Predictions = C50_model.predict(X_test)
```

#Create dot file to visualise tree

```
 dotfile = open("pt.dot", 'w')
```

```
     df = tree.export_graphviz(C50_model, out_file=dotfile, feature_names
= train.columns)
```

#build confusion matrix

```
     from sklearn.metrics import confusion_matrix

     CM = confusion_matrix(y_test, y_pred)
     CM = pd.crosstab(y_test, C50_Predictions)

    #let us save TP, TN, FP, FN
    TN = CM.iloc[0,0]
    FN = CM.iloc[1,0]
    TP = CM.iloc[1,1]
    FP = CM.iloc[0,1]

    #check accuracy of model
    #accuracy_score(y_test, y_pred)*100
    ((TP+TN)*100)/(TP+TN+FP+FN)

    #False Negative rate
    (FN*100)/(FN+TP)


# Fit the model
model = C50_model()
model.fit(X, Y)

# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

#Random Forest

```
    from sklearn.ensemble import RandomForestClassifier

    RF_model = RandomForestClassifier(n_estimators = 20).fit(X_train,
y_train)

    RF_Predictions = RF_model.predict(X_test)

    #build confusion matrix
    from sklearn.metrics import confusion_matrix
    CM = confusion_matrix(y_test, y_pred)
    CM = pd.crosstab(y_test, RF_Predictions)

    #let us save TP, TN, FP, FN
    TN = CM.iloc[0,0]
    FN = CM.iloc[1,0]
    TP = CM.iloc[1,1]
    FP = CM.iloc[0,1]

    #check accuracy of model
    accuracy_score(y_test, y_pred)*100
```

```
((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
(FN*100)/(FN+TP)
```

Let us prepare data for logistic regression

```
#replace target categories with Yes or No
train['churn'] = train['churn'].replace('No', 0)
train['churn'] = train['churn'].replace('Yes', 1)

#Create logistic data. Save target variable first
train_logit = pd.DataFrame(train['churn'])

#Add continous variables
train_logit = train_logit.join(train[cnames])

#Create dummies for categorical variables
cat_names = ["state"," international plan"," voice mail plan","Churn"]

for i in cat_names:
   temp = pd.get_dummies(train[i], prefix = i)
   train_logit = train_logit.join(temp)

Sample_Index = np.random.rand(len(train_logit)) < 0.8

train = train_logit[Sample_Index]
test =  train_logit[~Sample_Index]

#select column indexes for independent variables
train_cols = train.columns[1:30]
```

#Built Logistic Regression

```
import statsmodels.api as sm

logit = sm.Logit(train['churn'], train[train_cols]).fit()

logit.summary()

#Predict test data
test['Actual_prob'] = logit.predict(test[train_cols])

test['ActualVal'] = 1
test.loc[test.Actual_prob < 0.5, 'ActualVal'] = 0


#Build confusion matrix
CM = pd.crosstab(test['churn'], test['ActualVal'])

#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
```

```
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

#check accuracy of model
accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)

(FN*100)/(FN+TP)
```

#KNN implementation

```
from sklearn.neighbors import KNeighborsClassifier

KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(X_train,
y_train)

#predict test cases
KNN_Predictions = KNN_model.predict(X_test)

#build confusion matrix
CM = pd.crosstab(y_test, KNN_Predictions)

#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]

#check accuracy of model
#accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
(FN*100)/(FN+TP)
```

#Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

#Naive Bayes implementation
NB_model = GaussianNB().fit(X_train, y_train)


#predict test cases
NB_Predictions = NB_model.predict(X_test)


#Build confusion matrix
CM = pd.crosstab(y_test, NB_Predictions)
```

- **Code for Bar Plot**

```
#load libraries
library("ggplot2")
library("scales")
library("psych")
library("gplots")

ggplot(train, aes_string(x = train$Churn)) +
geom_bar(stat="count",fill =  "Pink") + theme_bw() +
xlab("Churn Prediction of Customer Behavior ") +  theme(text=element_text(size=11))


train$Churn = ifelse(train$Churn ==1,'No','yes')
write.csv(train, "train_finel.csv")
```

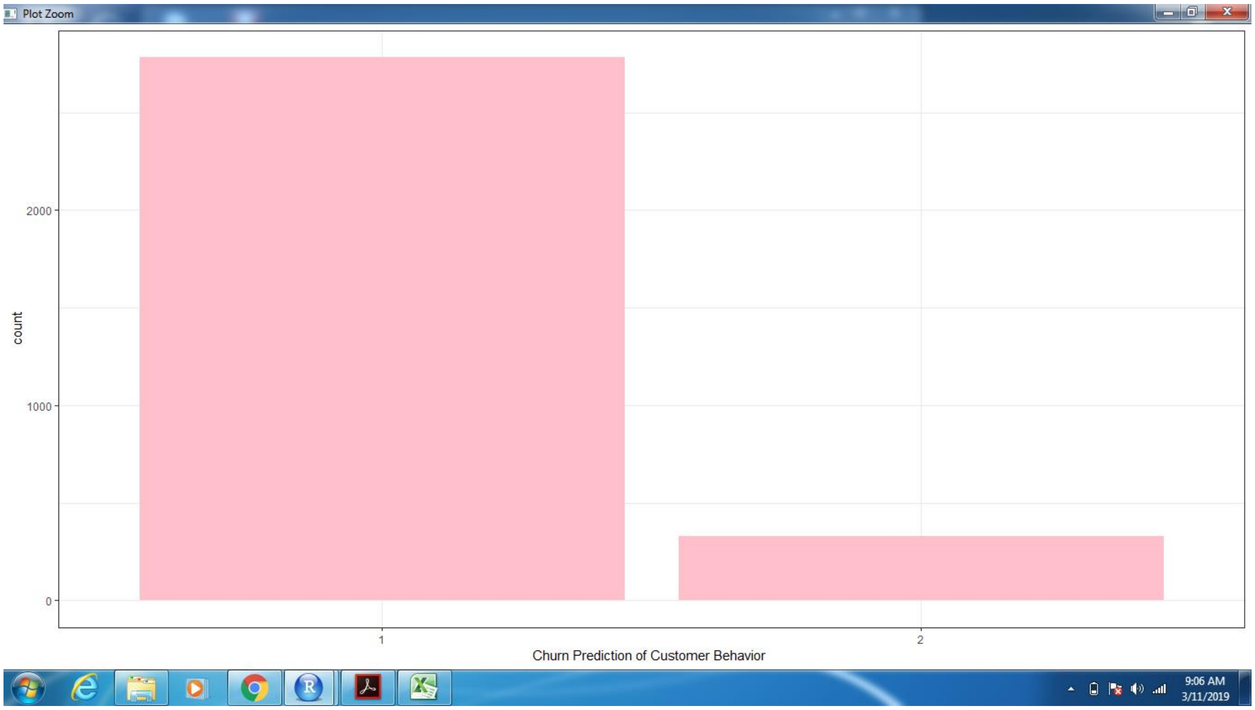


Fig. Bar Plot for Customer Behavior Where 1 Represent No and 2 as Yes