# Churn Reduction

Sanjib Kumar Mishra

## Contents

# Chapter 1

## Introduction

### Customer Churn

Customer churn occurs when customers or subscribers stop doing business with a company or service, also known as customer attrition. It is also referred as loss of clients or customers. In telecommunication industries churn rates are particularly useful as the customer have multiple options from which to choose within a geographic location.

### 1.1 Problem Statement

The objective of this Case is to predict customer behavior. Here I got a public dataset that has customer usage pattern and if the customer has moved or not. I have to develop an algorithm to predict the churn score based on usage pattern.

### 1.2 Data

Dataset Details:

For this project I have train and test dataset. So, first step towards my project is to develop the model on train data, then validate the model with the test data.

The details of the train and test data are as follows;

**Train Data**

**dim(train)**

**Output:**
```
[1] 3333    21
```
Here my train data has 3333 observations and 21 variables.

**Test Data**

**dim(test)**

**Output:** `[1] 1667   21`
The test  data contains 1667 observations and 21 variables.

**Variables details:**

**colnames(train)**

```
 [1] "state"                         "account.length"
 [3] "area.code"                     "phone.number"
 [5] "international.plan"            "voice.mail.plan"
 [7] "number.vmail.messages"         "total.day.minutes"
 [9] "total.day.calls"               "total.day.charge"
[11] "total.eve.minutes"             "total.eve.calls"
[13] "total.eve.charge"              "total.night.minutes"
[15] "total.night.calls"             "total.night.charge"
[17] "total.intl.minutes"            "total.intl.calls"
[19] "total.intl.charge"             "number.customer.service.calls"
[21] "Churn"
```

- Out of 21 variables “Churn” variable is my target variable which having two class i.e. False and True.
- Here True means customer has moved and False means customer has not moved

**Exploratory Data Analysis**

Before doing the exploratory analysis let’s check the structure of the dataset.

#Structure of the dataset

**str(train)**

```
'data.frame':       3333 obs. of  21 variables:
 $ state                    : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50 ...
 $ account.length           : int  128 107 137 84 75 118 121 147 117 141 ...
 $ area.code                : int  415 415 415 408 415 510 510 415 408 415 ...
 $ phone.number             : Factor w/ 3333 levels " 327-1058"," 327-1319",..: 1927 1576 1118 1708 1
 $ international.plan       : Factor w/ 2 levels " no"," yes": 1 1 1 2 2 2 1 2 1 2 ...
 $ voice.mail.plan          : Factor w/ 2 levels " no"," yes": 2 2 1 1 1 1 2 1 1 2 ...
 $ number.vmail.messages    : int  25 26 0 0 0 0 24 0 0 37 ...
 $ total.day.minutes        : num  265 162 243 299 167 ...
 $ total.day.calls          : int  110 123 114 71 113 98 88 79 97 84 ...
 $ total.day.charge         : num  45.1 27.5 41.4 50.9 28.3 ...
 $ total.eve.minutes        : num  197.4 195.5 121.2 61.9 148.3 ...
 $ total.eve.calls          : int  99 103 110 88 122 101 108 94 80 111 ...
 $ total.eve.charge            : num  16.78 16.62 10.3 5.26 12.61 ...
 $ total.night.minutes         : num  245 254 163 197 187 ...
 $ total.night.calls           : int  91 103 104 89 121 118 118 96 90 97 ...
 $ total.night.charge          : num  11.01 11.45 7.32 8.86 8.41 ...
 $ total.intl.minutes          : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
 $ total.intl.calls            : int  3 3 5 7 3 6 7 6 4 5 ...
 $ total.intl.charge           : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
 $ number.customer.service.calls: int  1 1 0 2 3 0 3 0 1 0 ...
 $ Churn                       : Factor w/ 2 levels " False."," True.": 1 1 1 1 1 1 1 1 1 1 ...
.
```

Here I removed state,account.length, area.code and phone.number variables from the dataset as it will not Put any value on my project.

```
# Remove variables from the dataset
train=train[,-c(1,2,3,4)]
test= test[,-c(1,2,3,4)]
```

Now I have only 17 variables.

# Chapter 2

# Methodology

## 2.1 Pre Processing

Data preprocessing is a technique that involves transforming raw data into an understandable format. Real-world data is often **incomplete**, **inconsistent**, and/or **lacking** in certain **behaviors or trends**, and is likely to contain many **errors**. Data preprocessing is a proven method of resolving such issues. Data preprocessing **prepares raw data** for **further processing**.

### 2.1.1 Missing Value Analysis

**R code for Missing value analysis:**

```
sum(is.na(train))
[1] 0
```

I have Checked for the missing value in training dataset and found there is no missing value present in the dataset as shown above.

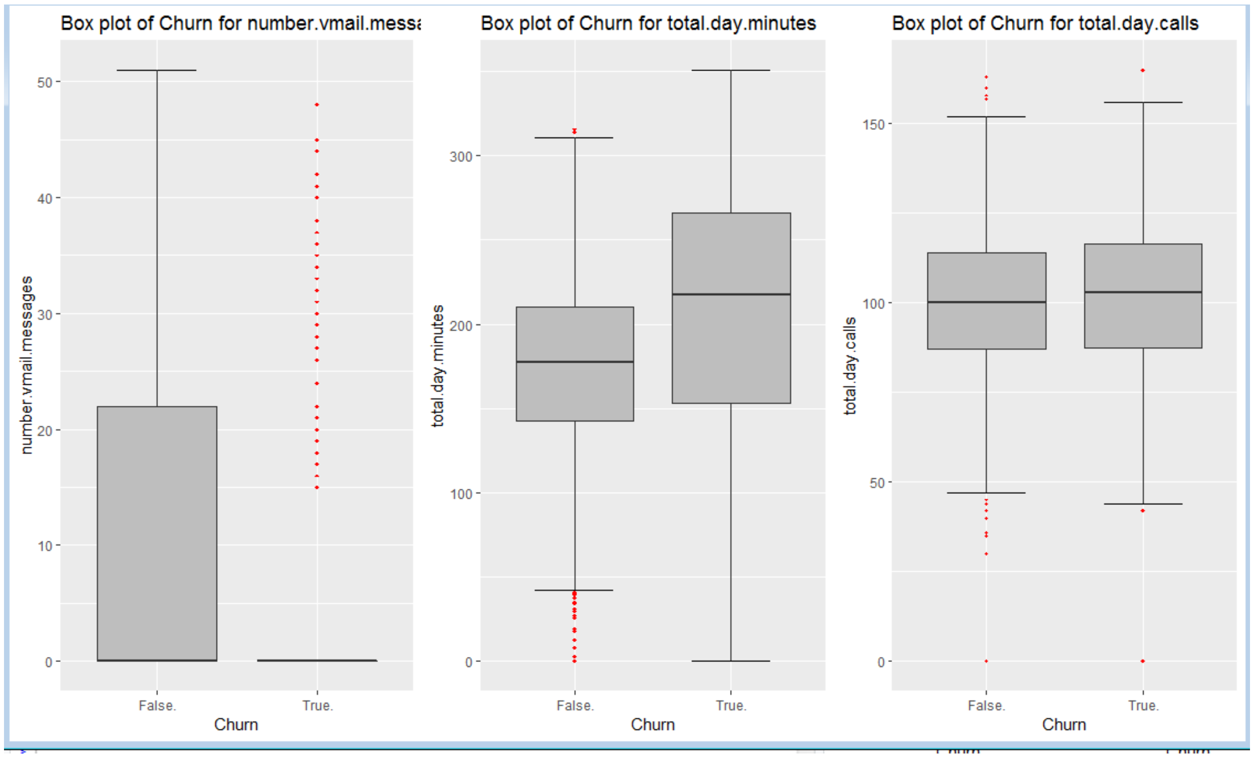
### 2.1.2 Outlier Analysis

Applied the box plot in all the numeric variables of the train dataset and found all variables having the outliers which will mislead the model. So, I replaced all outliers from the variables with NA and the impute by using KNNImputation.

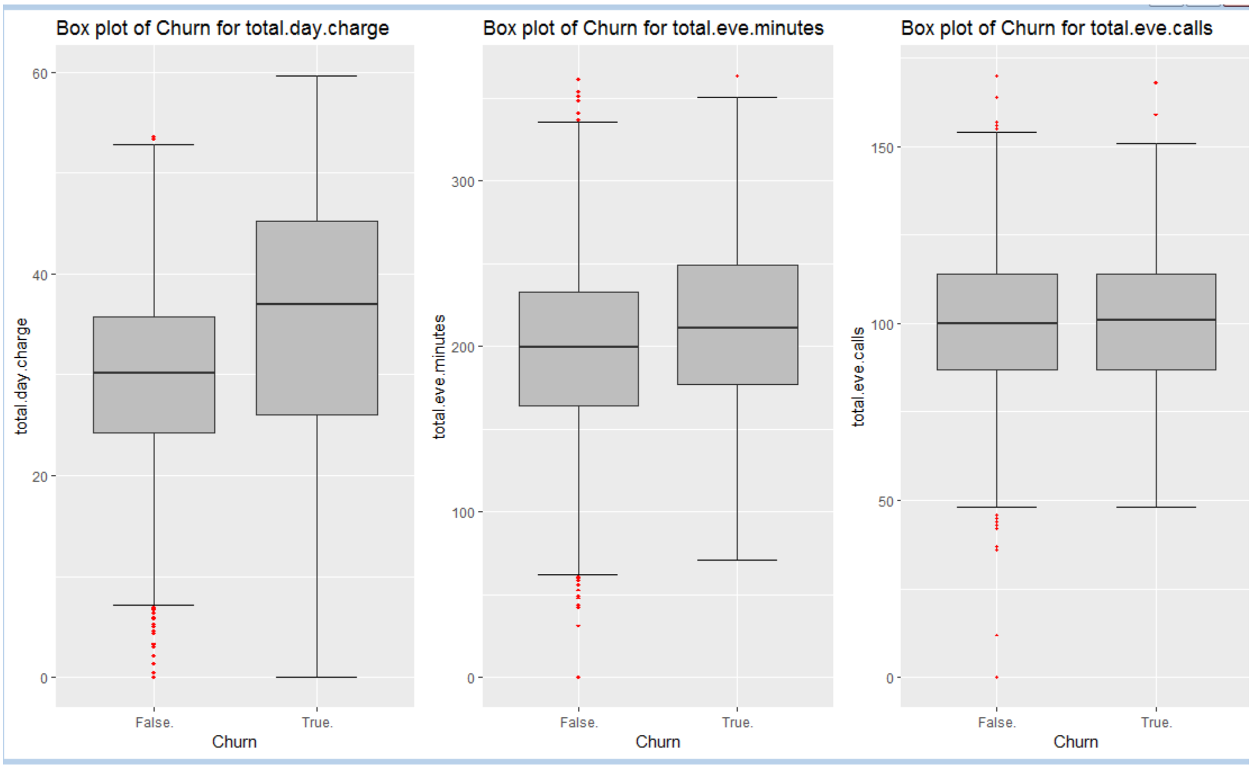
**R code for outlier :**

```
library(DMwR)
for(i in cnames)
{
val=train[,i][train[,i]%in% boxplot.stats(train[,i])$out]
 train[,i][train[,i]%in%val]=NA
}
train=knnImputation(train,k=3)
```

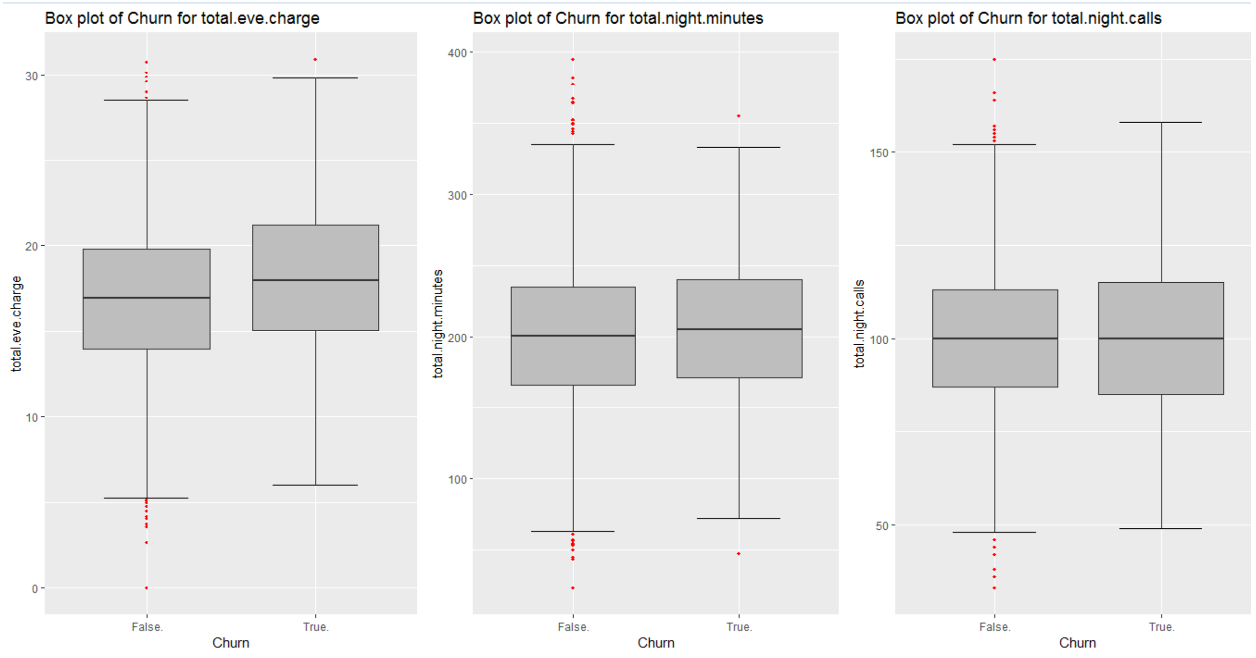
## Box plot for outliers



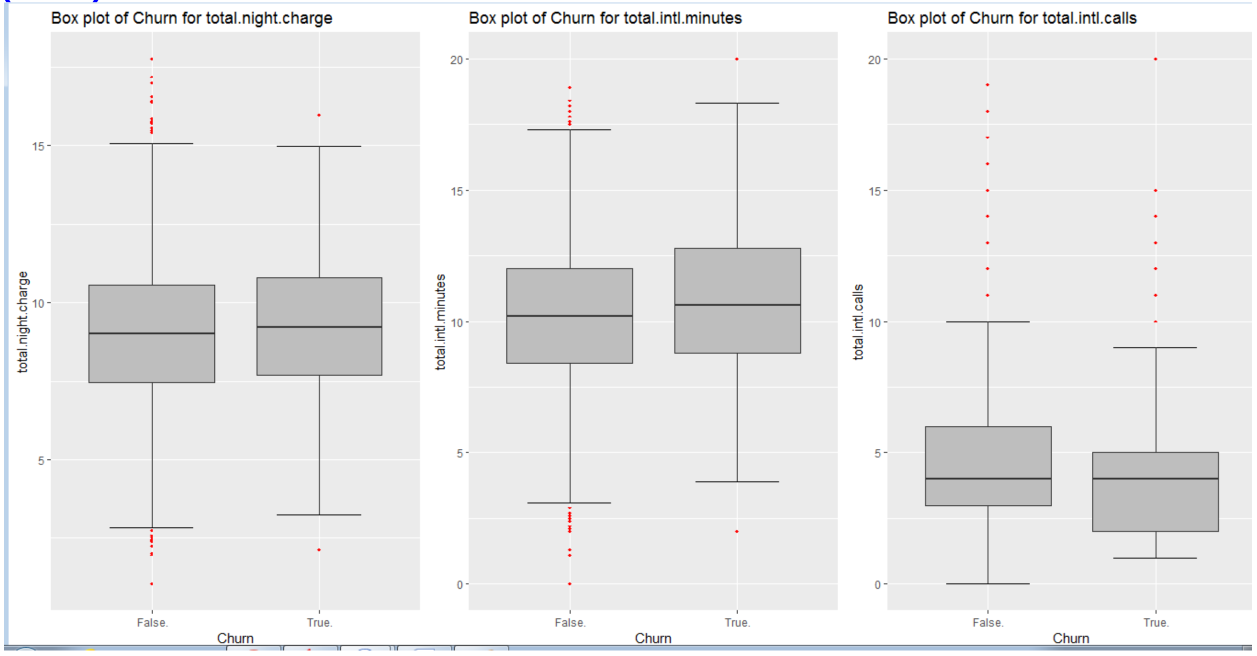
**(FIG -1)**



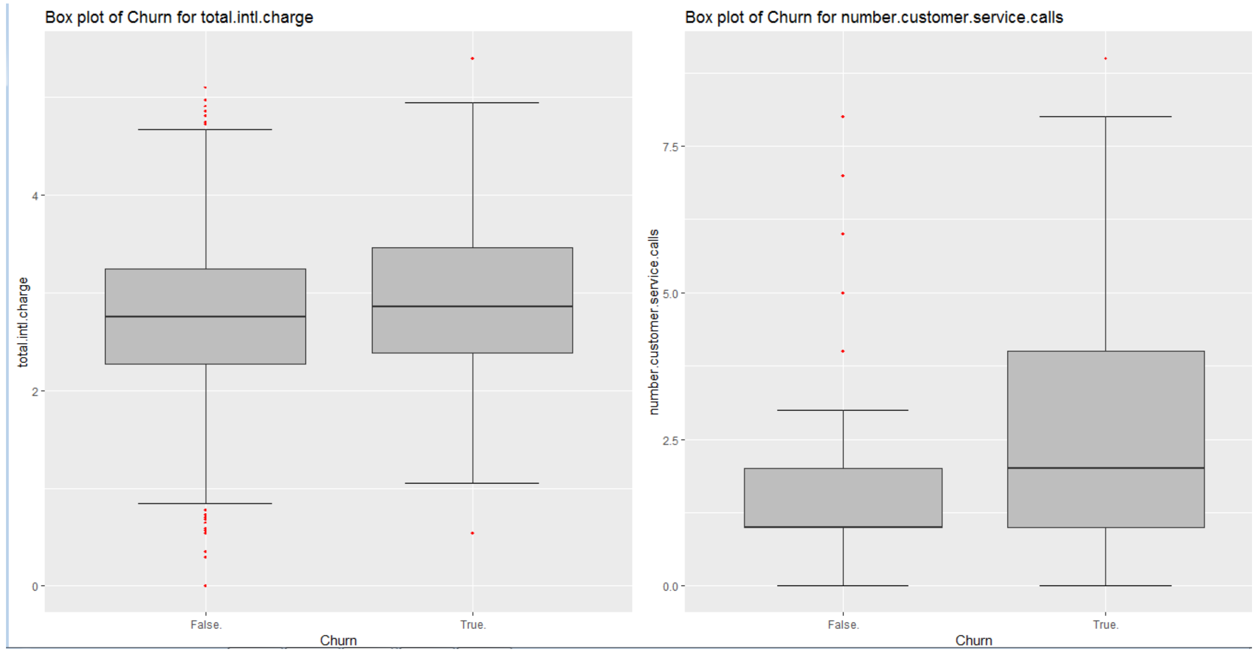
**(FIG-2)**



**(FIG-3)**



**(FIG-4)**

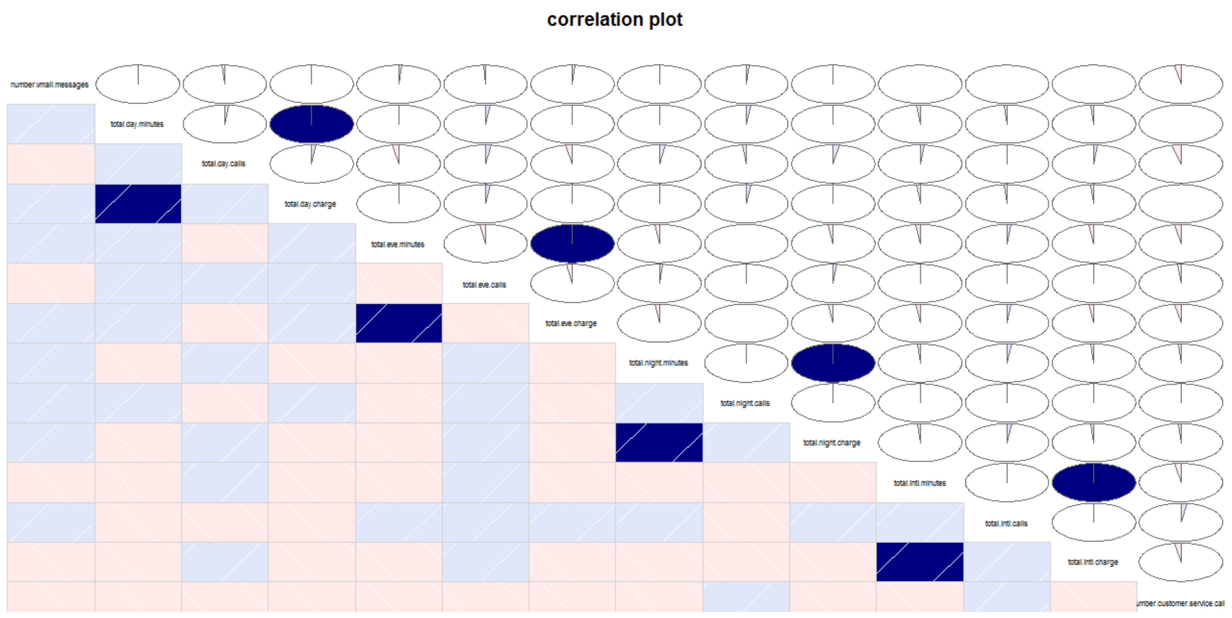

Box plot of Churn for total.intl.charge
total.intl.charge
0
2
4
False.
True.
Churn
Box plot of Churn for number.customer.service.calls
number.customer.service.calls
0.0
2.5
5.0
7.5
False.
True.
Churn

**(FIG-4)**

## 2.1.3 Feature Selection

- In feature selection method I checked the association between two variables.
- For continuous variables I used correlation analysis which tells the direction and strength of the linear relationship between two quantitative variables.
- I found total.day.minutes and total.day.charge, total.eve.minutes and total.eve.charge , total.night.minutes and total.night.charge, total.Intl.minutes and total.Intl.charge are highly correlated with each other .
- So I remove total.day.minutes, total.eve.minutes, total.night.minutes and total.Intl.minutes from the train and test dataset.
- For Categorical variables I used Chi-Square test which compare two variables in a contingency table to see if they are related or not.

**<u>Correlation analysis plot:</u>**



**#Removal of total.day.minutes, total.eve.minutes, total.night.minutes and total.Intl.minutes variables from train and test dataset.**

```
train=train[,-c(4,7,10,13)]
train=train[,-c(4,7,10,13)]
```

**Chi-Square Test**

```
> factor_index=sapply(train,is.factor)
> factor_data=train[,factor_index]
> for (i in 1:2)
+ {
+   print(names(factor_data)[i])
+   print(chisq.test(table(factor_data$Churn,factor_data[,i])))
+ }
```

**Output**

```
[1] "international.plan"

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

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

- In the above output I saw the P-value for the two variables is greater than 0.05 so I reject alternate hypothesis and select the null hypothesis i.e. the variables are independent with each other.

## 2.1.4 Class Imbalance

**table(train$Churn)**

```
 False.   True.
   2850     483
```

As shown above my target variable of train data is having two class i.e False and True but here the problem is out of 3333 observations one class is of 2850 observations and another one is only 483.
Here class imbalance occurs and the model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.

Standard classifier algorithms like Decision Tree and logistic Regression have a bias towards classes which have number of instances. They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored. Thus, there is a high probability of misclassification of the minority class as compared to the majority class.

## Approach to handling Imbalanced class

- **Under-Sampling**
  Under sampling aims to balance class distribution by randomly eliminating majority class examples . This is done until the majority and minority class instances are balanced.
  **R code for Undersampling :**

  ```
  library(ROSE)
  under = ovun.sample(Churn~.,data=train,method='under',N=966)$data
  ```

  Here my minority class is having 483 observations. So, to balanced the instance I take N = 966.

  **Balanced data after Under-Sampling:**

  ```
    table(under$Churn)

     False.   True.
      483     483
  ```

- **Over-Sampling**
  Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

  **R code for Over-Sampling:**

  ```
  Over = ovun.sample(Churn~.,data=train,method='over',N=5700)$data
  ```

  Here my majority class is having 2850 observations. So, to balanced the instances I take N = 5700.

**Balanced data after Over-Sampling:**

```
table(over$Churn)

 False.   True.
  2850    2850
```

- **Both Sampling**
  This method is a combination of both oversampling and undersampling methods. Using this method, the majority class is undersampled without replacement and the minority class is oversampled with replacement.

  **R code for Both Sampling**

  ```
  both=ovun.sample(Churn~.,data=train,method='both',p=0.5,seed=1234,
  N=3333)$data
  ```

  **Balanced data after Both Sampling:**

  ```
    table(both$Churn)

     False.   True.
     1663    1670
  ```

- **ROSE(Random Over Sampling Examples)**
  Rose sampling method generates data synthetically and provides a better estimate of original data.

  This method is used to avoid overfitting when adding exact replicas of minority instances to the main dataset.

  **R code for ROSE**

  ```
  rose=ROSE(Churn~.,data=train,N=3500,seed=1321)$data
  ```

  **Balanced data after ROSE:**

  ```
    table(rose$Churn)

     False.   True.
    1816    1684
  ```

## 2.2 Modeling

### 2.2.1 Model Selection

I am selecting the Classification model for Churn reduction project to predict the churn score based on usage pattern as my target variable is having binary class.
For Classification model I am going to apply the Logistic Regression, Decision tree, Random Forest and Naïve Bayes .

### 2.2.2 Model Evaluation

Model evaluation is the process of choosing between models, different model types and features. Better evaluation processes lead to better, more accurate models in applications.

**RandomForest**

**Random Forest on rose data**

```
> rf=randomForest(Churn~.,rose,importance=TRUE,ntree=500)
> confusionMatrix(predict(rf,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1256     78
    True.      187    146

               Accuracy : 0.841
                 95% CI : (0.8226, 0.8583)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 0.9982

                  Kappa : 0.4332
 Mcnemar's Test P-Value : 3.259e-11

            Sensitivity : 0.8704
            Specificity : 0.6518
         Pos Pred Value : 0.9415
         Neg Pred Value : 0.4384
             Prevalence : 0.8656
         Detection Rate : 0.7534
   Detection Prevalence : 0.8002
      Balanced Accuracy : 0.7611

       'Positive' Class :  False.
```

**<u>Random Forest on Under_Sampling</u>**

```
rf=randomForest(Churn~.,under,importance=TRUE,ntree=100)
> confusionMatrix(predict(rf,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1255     81
    True.      188    143

               Accuracy : 0.8386
                 95% CI : (0.8201, 0.856)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 0.9993

                  Kappa : 0.4228
 Mcnemar's Test P-Value : 1.027e-10

            Sensitivity : 0.8697
            Specificity : 0.6384
         Pos Pred Value : 0.9394
         Neg Pred Value : 0.4320
             Prevalence : 0.8656
         Detection Rate : 0.7528
   Detection Prevalence : 0.8014
      Balanced Accuracy : 0.7541

       'Positive' Class :  False.
```

**<u>RandomForest on Over_Sampling</u>**

```
> rf=randomForest(Churn~.,over,importance=TRUE,ntree=100)
> confusionMatrix(predict(rf,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1437    100
    True.        6    124

               Accuracy : 0.9364
                 95% CI : (0.9236, 0.9476)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6678
```

```
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9958
            Specificity : 0.5536
         Pos Pred Value : 0.9349
         Neg Pred Value : 0.9538
             Prevalence : 0.8656
         Detection Rate : 0.8620
   Detection Prevalence : 0.9220
      Balanced Accuracy : 0.7747

       'Positive' Class :  False.
```

**RandomForest on Both_Sampling**

```
> rf=randomForest(Churn~.,both,importance=TRUE,ntree=100)
> confusionMatrix(predict(rf,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1418     93
    True.       25    131

               Accuracy : 0.9292
                 95% CI : (0.9158, 0.9411)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.651
 Mcnemar's Test P-Value : 6.922e-10

            Sensitivity : 0.9827
            Specificity : 0.5848
         Pos Pred Value : 0.9385
         Neg Pred Value : 0.8397
             Prevalence : 0.8656
         Detection Rate : 0.8506
   Detection Prevalence : 0.9064
      Balanced Accuracy : 0.7837

       'Positive' Class :  False.
```

In the above four model of RandomForest I found rose data gave the better Specificity than other three. So, for other model building I used rose data.

**Decision Tree on rose data**

```
> library(C50)
> mo=C5.0(Churn~.,rose,trails=100,rules=TRUE)
> confusionMatrix(predict(mo,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1321     82
    True.      122    142

               Accuracy : 0.8776
                 95% CI : (0.8609, 0.893)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 0.079374

                  Kappa : 0.5109
 Mcnemar's Test P-Value : 0.006323

            Sensitivity : 0.9155
            Specificity : 0.6339
         Pos Pred Value : 0.9416
         Neg Pred Value : 0.5379
             Prevalence : 0.8656
         Detection Rate : 0.7924
   Detection Prevalence : 0.8416
      Balanced Accuracy : 0.7747

       'Positive' Class :  False.
```

**Naïve Bayes on rose data**

```
> nb=naiveBayes(Churn~.,data=rose)
> confusionMatrix(predict(nb,test),test$Churn)
Confusion Matrix and Statistics

          Reference
Prediction  False.  True.
    False.    1178     89
    True.      265    135

               Accuracy : 0.7876
                 95% CI : (0.7672, 0.8071)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3146
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8164
            Specificity : 0.6027
         Pos Pred Value : 0.9298
         Neg Pred Value : 0.3375
             Prevalence : 0.8656
         Detection Rate : 0.7067
   Detection Prevalence : 0.7600
      Balanced Accuracy : 0.7095

       'Positive' Class :  False.
```

**Logistic Regression on ROSE**

```
> lgrose=glm(Churn~.,data=rose,family="binomial")
> prediction=predict(lgrose,newdata=test,type="response")
> prediction=ifelse(prediction>0.5,1,0)
> confmatrix=table(test$Churn,prediction)
> confmatrix
         prediction
            0     1
  False. 1192  251
  True.    76  148
```

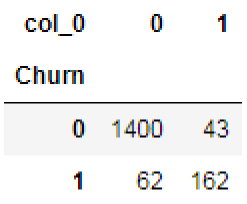
In the above confusionmatrix
TP= 1192
TN=148
FP=76
FN=251

```
> Accuracy=(TN+TP)/(TN+TP+FN+FP)
> Accuracy
[1] 0.8038392
> Specificity=TN/(TN+FP)
> Specificity
[1] 0.6607143
> Sensitivity=TP/(TP+FN)
> Sensitivity
 [1] 0.8260568
```

## Random Forest in python

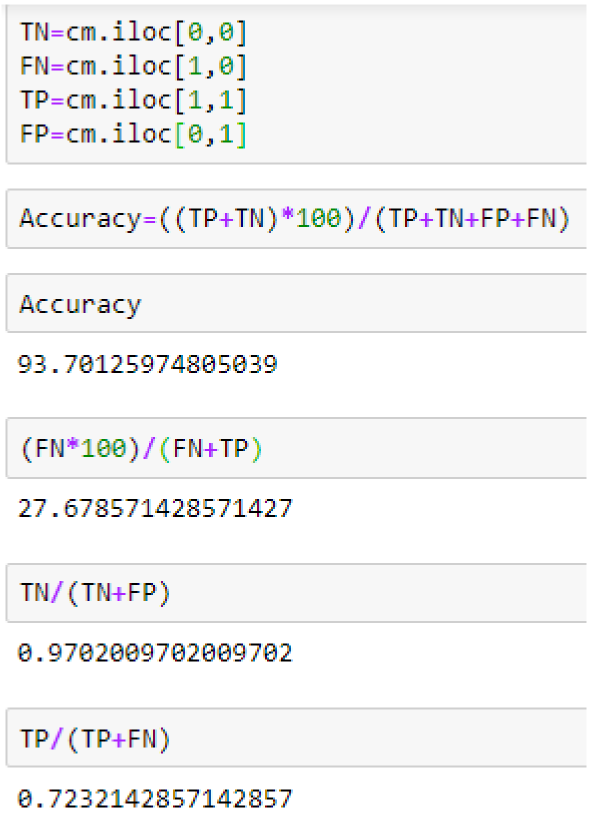
```
rf=RandomForestClassifier(n_estimators=100).fit(X_train,Y_train)

prediction=rf.predict(X_test)

cm=pd.crosstab(Y_test,prediction)

cm
```

## Confusion matrix



| col_0 | 0 | 1 |
|---|---|---|
| Churn | | |
| 0 | 1400 | 43 |
| 1 | 62 | 162 |

## Output



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

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

```
Accuracy
```

```
93.70125974805039
```

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

```
27.678571428571427
```

```
TN/(TN+FP)
```

```
0.9702009702009702
```

```
TP/(TP+FN)
```

```
0.7232142857142857
```

## Conclusion

From the model evaluation the results are as follows;

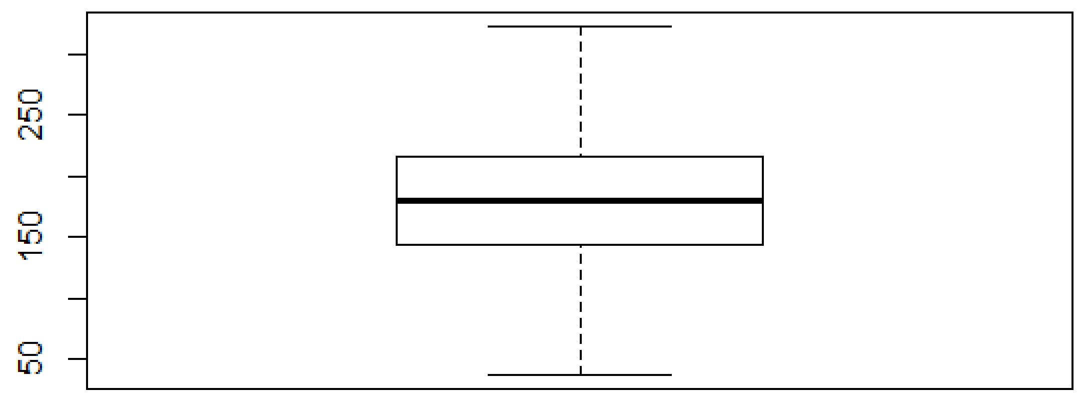
| Model Name | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| RandomForest | 84.1 | 87.04 | 65.18 |
| Decision Tree | 87.76 | 91.55 | 63.39 |
| Naïve Bayes | 78.76 | 81.64 | 60.27 |
| Logistic Regression | 80.38 | 82.60 | 66.07 |

From the above result I found RandomForest, Decision Tree and Logistic Regression gives the best result for the Churn Reduction problem.

I choose RandomForest model as the best model for this project as the Specificity and accuracy is better than other models and as per the business requirement my positive class is False.

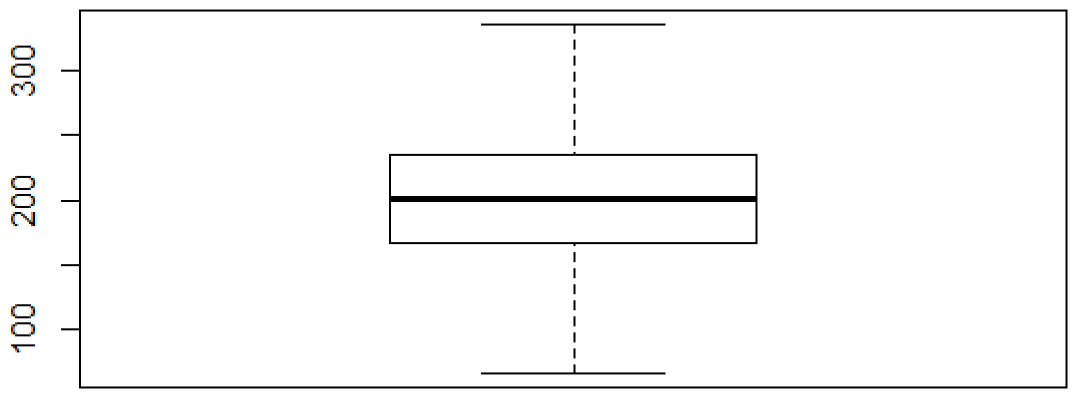
**Appendix-1-Extra Figures**

**Variables without outliers(some figures)**

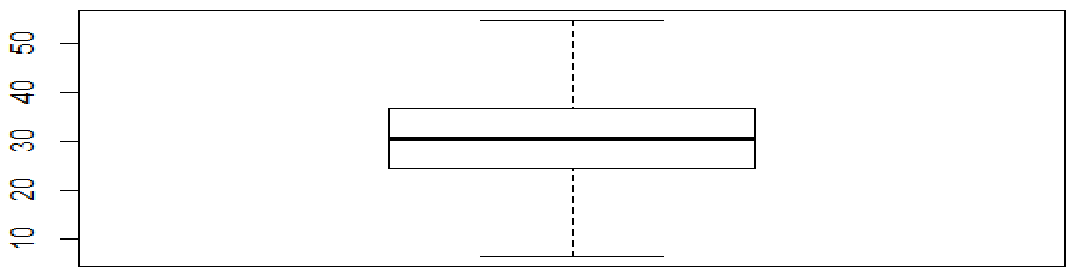


---

(**Total.day.minutes)**

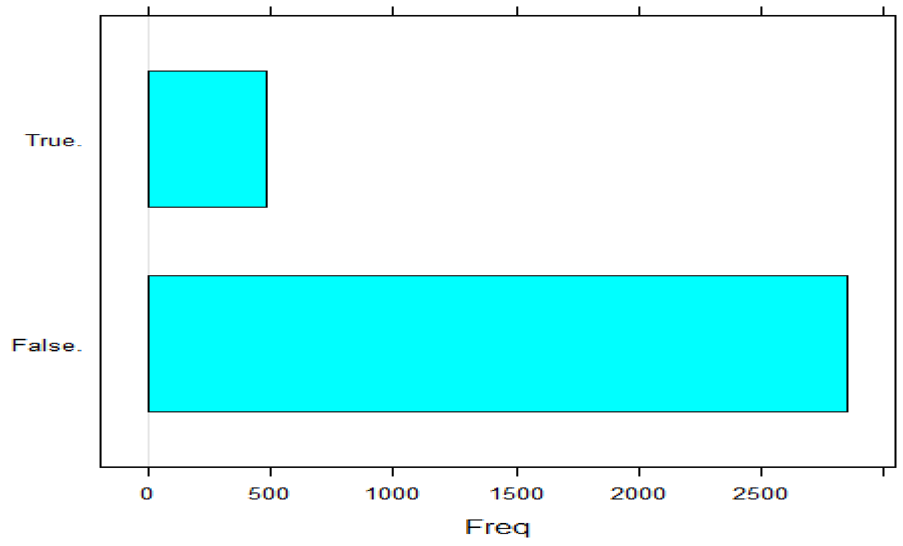


**(Total.eve.minutes)**

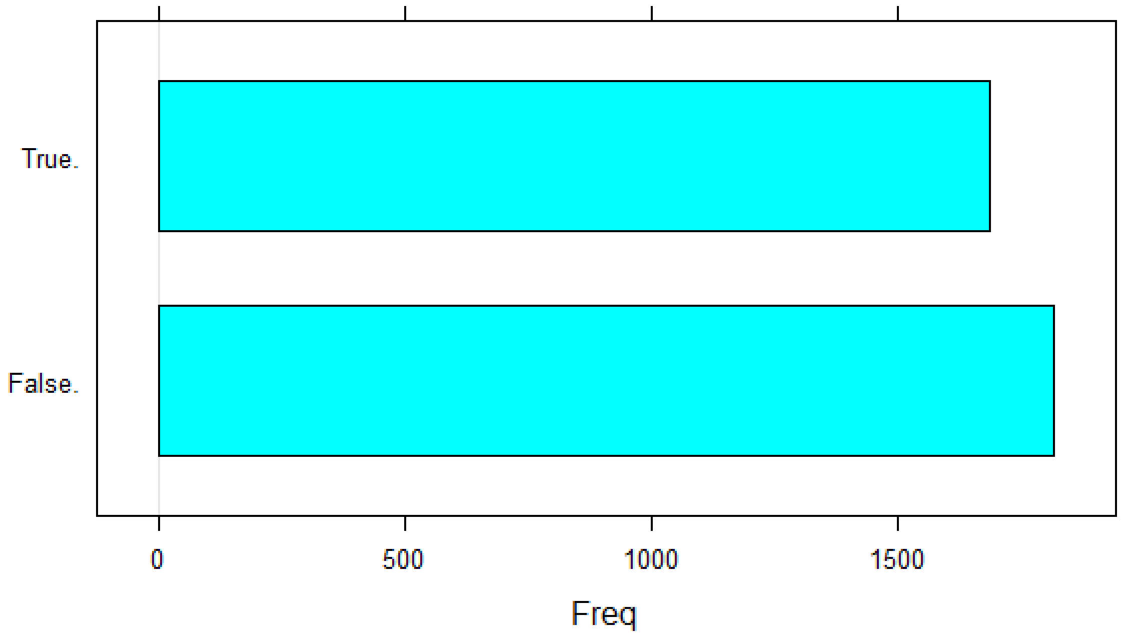


**(Total.day.charges)**

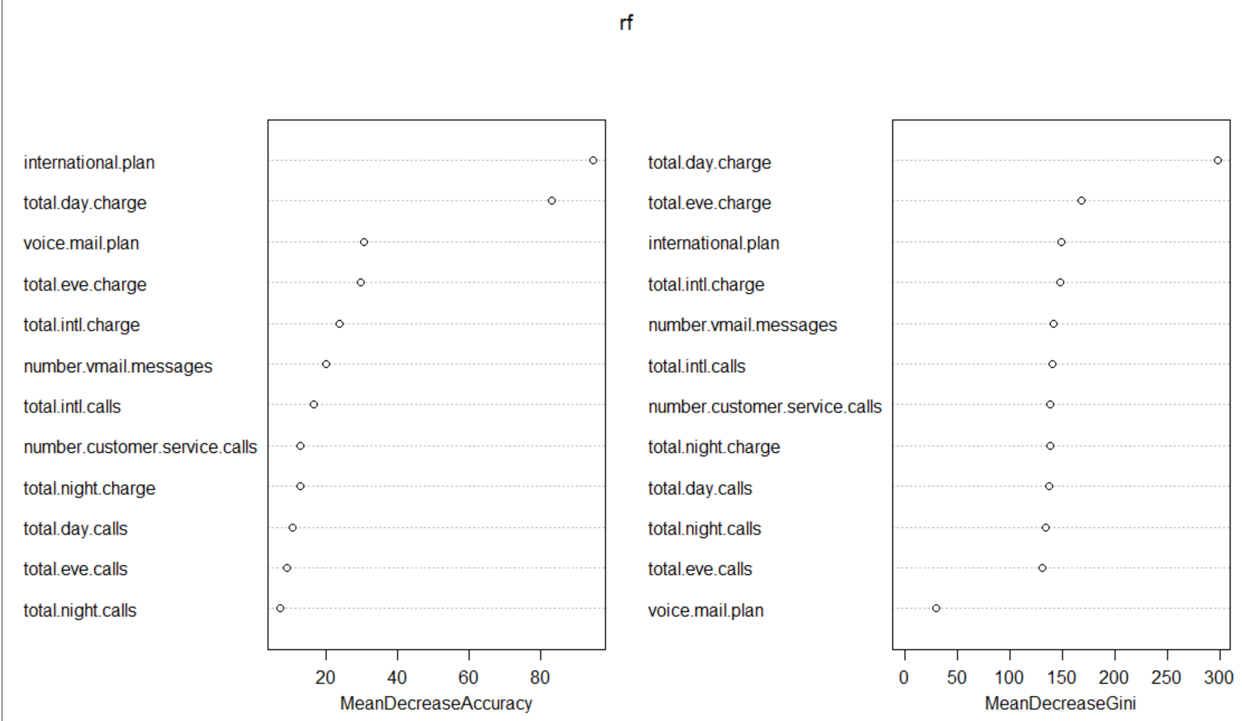
**Data before balanced**



**Data after balanced**



## Variable Importance Plot RandomForest



## Appendix 2 – R Code

```
> setwd("D:/PROJECT")
> train=read.csv("Train_data.csv",header=T)
> test=read.csv("Test_data.csv",header=T)
> train=train[,-c(1,2,3,4)]
> numeric_index=sapply(train,is.numeric)
> numeric_data=train[,numeric_index]
> cnames=colnames(numeric_data)
> cnames
> library(ggplot2)
> 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])))}
>
> library(gridExtra)
> gridExtra::grid.arrange(gn1,gn2,gn3,ncol = 3)
> gridExtra::grid.arrange(gn4,gn5,gn6,ncol = 3)
> gridExtra::grid.arrange(gn7,gn8,gn9,ncol = 3)
> gridExtra::grid.arrange(gn10,gn11,gn12,ncol = 3)
> gridExtra::grid.arrange(gn13,gn14,ncol = 2)
> library(DMwR)
> for(i in cnames)
+ {val=train[,i][train[,i]%in% boxplot.stats(train[,i])$out]
+ train[,i][train[,i]%in%val]=NA}
> train=knnImputation(train,k=3)
> library(corrgram)
> corrgram(train[,numeric_index],order = F,upper.panel = panel.pie,
+         main="correlation plot")
> train=train[,-c(4,7,10,13)]
> test=test[,-c(1,2,3,4)]
> test=test[,-c(4,7,10,13)]
> factor_index=sapply(train,is.factor)
> factor_data=train[,factor_index]
> for (i in 1:2)
+ {
+   print(names(factor_data)[i])
+   print(chisq.test(table(factor_data$Churn,factor_data[,i])))
+ }
```

```
> library(randomForest)
> library(ROSE)
> library(caret)
> rose=ROSE(Churn~.,data=train,N=3500,seed=1321)$data
> rf=randomForest(Churn~.,rose,importance=TRUE,ntree=500)
> confusionMatrix(predict(rf,test),test$Churn)
> varImpPlot(rf)
```

## Appendix 3 – Python Code

```
import os

import pandas as pd

import numpy as np

import matplotlib as mlt

import matplotlib.pyplot as plt

import seaborn as sn

from imblearn.over_sampling import SMOTE

from sklearn.ensemble import RandomForestClassifier

os.chdir("D:/PROJECT")

train=pd.read_csv("Train_data.csv")

test=pd.read_csv("Test_data.csv")

train=train.drop(['state','account length','area code','phone number'],axis=1)

test=test.drop(['state','account length','area code','phone number'],axis=1)

def cat_to_num(df):

    for i in range(0, df.shape[1]):

        #print(i)

        if(df.iloc[:,i].dtypes == 'object'):

            df.iloc[:,i] = pd.Categorical(df.iloc[:,i])

            df.iloc[:,i] = df.iloc[:,i].cat.codes

            df.iloc[:,i] = df.iloc[:,i].astype('object')

    return df

train = cat_to_num(train)

test= cat_to_num(test)

train_class_0 = train[train['Churn'] == 0]
```

```
train_class_1 = train[train['Churn'] == 1]
count_class_0, count_class_1 = train.Churn.value_counts()
train_class_1_over = train_class_1.sample(count_class_0,replace=True)
new = pd.concat([train_class_0,train_class_1_over], axis=0)
new.Churn.value_counts()
train=new
cnames=['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',
        'number customer service calls']
train_corr=train.loc[:,cnames]
f,ax=plt.subplots(figsize=(7,5))
corr=train_corr.corr()
sn.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sn.diverging_palette(220,10,as_cmap=True),square=True,ax=ax)
train=train.drop(['total day minutes','total eve minutes','total night minutes','total intl minutes', ],axis=1)
test=test.drop(['total day minutes','total eve minutes','total night minutes','total intl minutes', ],axis=1)
X_train=train.iloc[:,0:12]
Y_train=train.iloc[:,12]
X_test=test.iloc[:,0:12]
Y_test=test.iloc[:,12]
X_train=X_train.astype('int')
```

```
Y_train=Y_train.astype('int')

rf=RandomForestClassifier(n_estimators=100).fit(X_train,Y_train)

prediction=rf.predict(X_test)

from sklearn.metrics import confusion_matrix

cm=pd.crosstab(Y_test,prediction)

cm

TN=cm.iloc[0,0]

FN=cm.iloc[1,0]

TP=cm.iloc[1,1]

FP=cm.iloc[0,1]

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

(FN*100)/(FN+TP)

TN/(TN+FP)

TP/(TP+FN)
```