CHURN REDUCTION PROJECT REPORT

-Shravya Suresh

INDEX

CHAPTER 1

Introduction

Customer churn the rate when a customer leaves or stop paying for a product of service. Reduction customer churn is important because cost of acquiring a new customer is higher than retaining an existing one. Reducing customer churn is a key business goal of every business. Predicting and preventing customer churn represents a huge additional potential revenue source for every business.

1.1 Problem Statement

The problem statement provides us with a train and test dataset of a telecom company. The data set consists of 20 variables describing various services and charges associated with them, duration and any service calls made by customer. The dataset also contains geographic location of customers in form of state. 'Churn' is the target variable, which tells us weather the customer has churned out or not.

1.2 Dataset

Two different datasets were provided as train data and test data. Data contains 20 predictor variables and 1 target variables. Train data had 3333 observations while test data consist of 1667. Total 21 variables were present in data. All the variables are described in below.

State	Customer's State
Account length	Service usage period
Area code	Pin code
Phone number	Contact number of the customer
International plan	'yes' if customer opted for international plan else 'no'
Voice mail plan	'yes' if customer opted for voice mail plan else 'no'
Number of vmail messages	Number of voice mail messages received or sent
Total day minutes	Total minutes of usage in day
Total day calls	Total number of calls made or received in the day
Total day charge	Total charge in daytime
Total eve minutes	Total minutes of usage in evening
Total eve calls	Total number of calls made or received in the evening
Total eve charge	Total charge in evening time
Total night minutes	Total minutes of usage in night
Total night calls	Total number of calls made or received in the night
Total night charge	Total charge in night time
Total intl minutes	Total minutes of international usage
Total intl calls	Total number of international calls made or received
Total intl charge	Total international charge
Number of customer service calls	Services call made by customer
Churn	Target - 'True.' If customer churned else 'False'

Out of 20 variables, 04 were categorical and 16 were continuous.

Column1	Column2	Column3	Column4	Column5
state	state	state	state	state
KS	128	415	382-4657	no
OH	107	415	371-7191	no
NJ	137	415	358-1921	no
OH	84	408	375-9999	yes
OK	75	415	330-6626	yes
AL	118	510	391-8027	yes

Column1	Column2	Column3	Column4	Column5
number vmail messages	number vmail messages	number vmail messages	total day charge	total eve minutes
25	265.1	110	45.07	197.4
26	161.6	123	27.47	195.5
0	243.4	114	41.38	121.2
0	299.4	71	50.9	61.9
0	166.7	113	28.34	148.3
0	223.4	98	37.98	220.6
24	218.2	88	37.09	348.5

total night calls	total night charge	total intl minutes	total intl calls	total intl charge	Churn
91	11.01	10	3	2.7	False.
103	11.45	13.7	3	3.7	False.
104	7.32	12.2	5	3.29	False.
89	8.86	6.6	7	1.78	False.
121	8.41	10.1	3	2.73	False.
118	9.18	6.3	6	1.7	False.
118	9.57	7.5	7	2.03	False.
96	9.53	7.1	6	1.92	False.

CHAPTER 2

Methodology

Customer churn reduction is a business scenario in which a company is trying to retain a customer which is more likely to leave the services. For reducing churn rate, we need to identify which customers are most likely to churn and which are not. So Churn reduction is a classification problem.

The solution is divided into 3 parts.

1. Exploratory data analysis(EDA) was performed to explore the structure of data. Some of the basic assumptions were made about the data ie. Which variables are most likely causing churn. During exploration dataset was checked for missing values, multi collinearity and other model/algorithm specific assumptions.

2. After EDA, for learning two models were used, logistic regression and random forest. Some data pre-processing was done to prepare training data for learning model.

3. In the last part, performance tuning was done to increase the accuracy of models. Both the algorithms and EDA were implemented in R and python. Both implementations were similar with little difference due to difference in learning algorithm implementation.

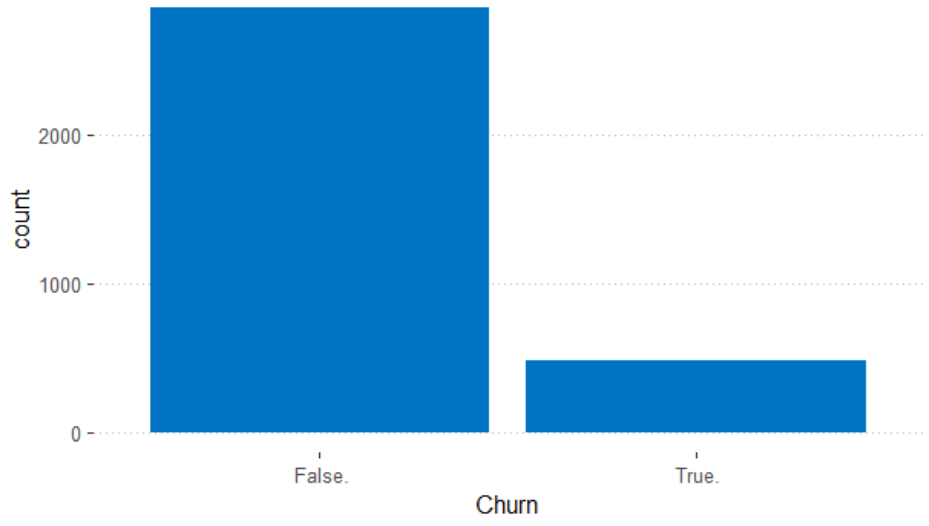
2.1 Exploratory Data Analysis (EDA)

Exploratory data analysis helps us in understanding the data as well as clearing it and hence cleaning it eventually.

Using the following exploratory data analysis, we have understood the data.

2.1.1 The Target Variable - Churn

The target variable was 'churn'. Initially if the customers churned out it flagged as 'True.' , otherwise 'False.'. Later during it was changes as True = 1 and False = 0. Out of 3333 customers, 483 customers churned out and 2850 didn't churned out.



2.1.2 Missing Value Analysis

No missing values were present in the training and test dataset.

Columns	Missing_percentage
state	0
account.length	0
area.code	0
phone.number	0
international.plan	0
voice.mail.plan	0
number.vmail.messages	0
total.day.minutes	0
total.day.calls	0
total.day.charge	0
total.eve.minutes	0
total.eve.calls	0
total.eve.charge	0
total.night.minutes	0
total.night.calls	0
total.night.charge	0
total.intl.minutes	0
total.intl.calls	0
total.intl.charge	0
number.customer.service.calls	0
Churn	0

2.1.3 Multicollinearity

Multicollinearity is basically a feature which defines the close relation between two variables or predictors. We say that two variables are highly collinear when the are very much dependent on each other. This is a problem because, it gives unnecessary variables in our dataset. Hence if two variables are highly collinear, then we can remove one of them, and retain the other one, preferably the one which is more dependent on our target variable. On applying the correlation plot in R and python, we get the following data.

From the correlation plot we can see that –

1. 'Total day minutes' and 'total day charges' are highly collinear

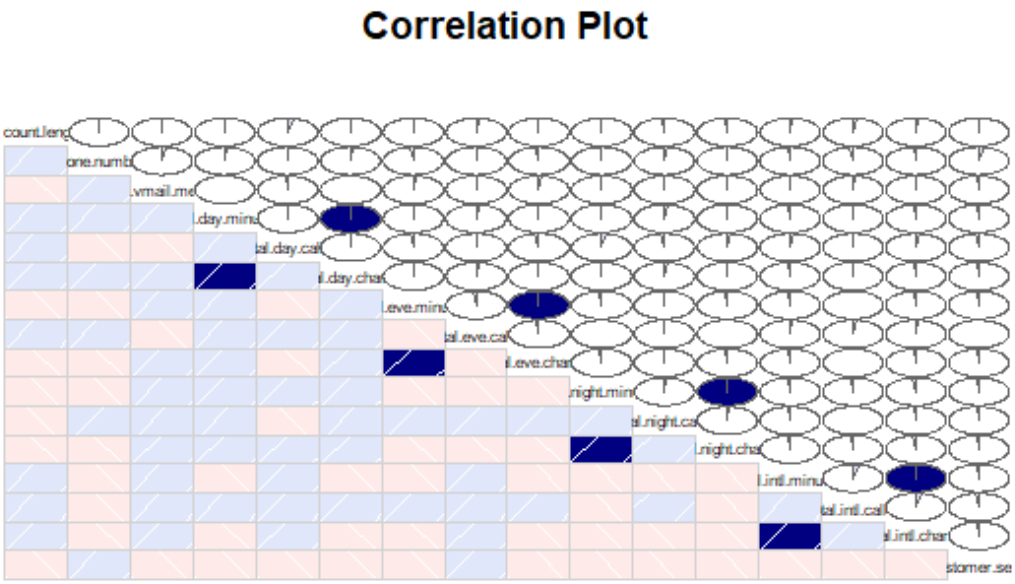
2. 'Total eve minutes' and 'total eve charges' are highly collinear

3. 'Total night minutes' and 'total night charges' are highly collinear

4. 'Total intl minutes' and 'total intl charges' are highly collinear

Multicollinearity matrix is visualized .

Variables are highly collinear are highlighted with red colour with their corresponding score. Apart from correlation matrix, multicollinearity was again checked during logistic regression diagnostics using VIF (Variation inflation factor).

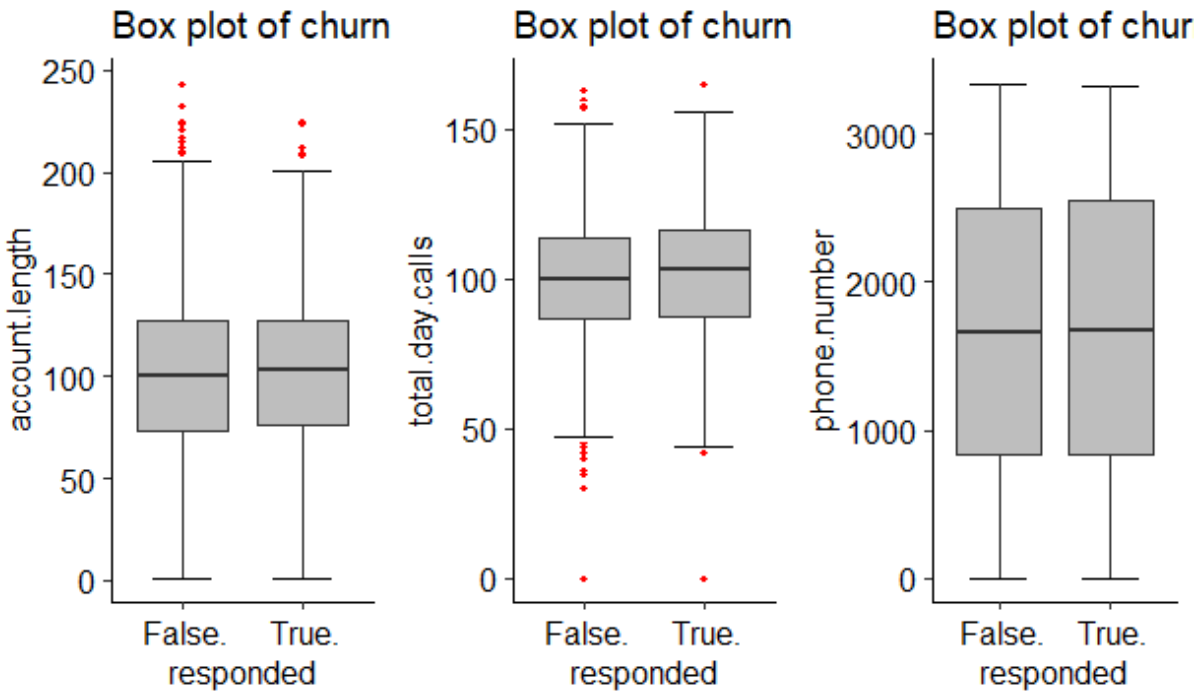


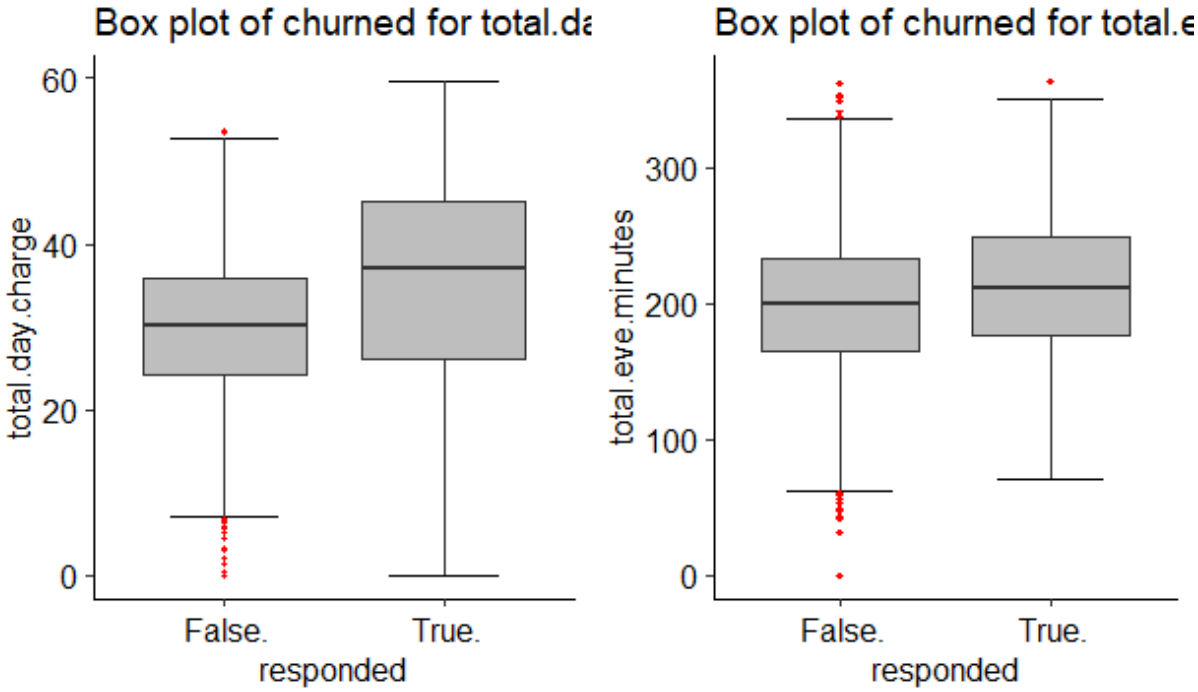
2.1.4 variable reduction

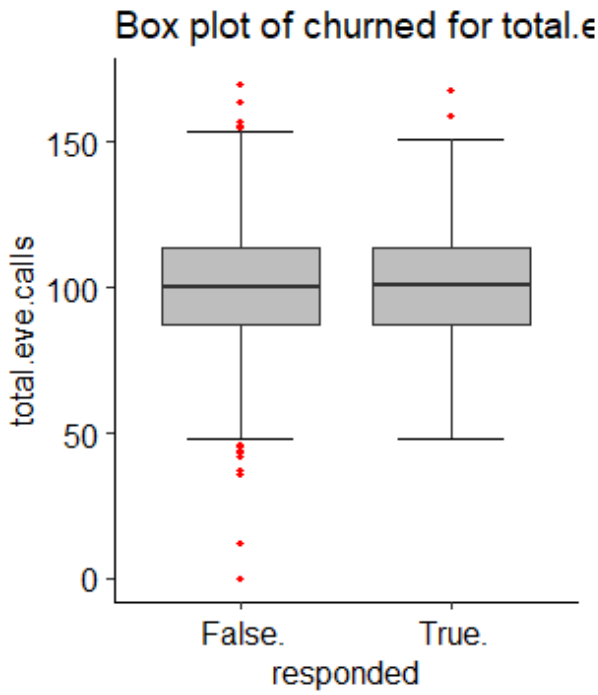
After applying the co relation analysis for continuous variables and chi square test for the factor variables, we hence reduce the dimention of the dataset by removing the following variables,

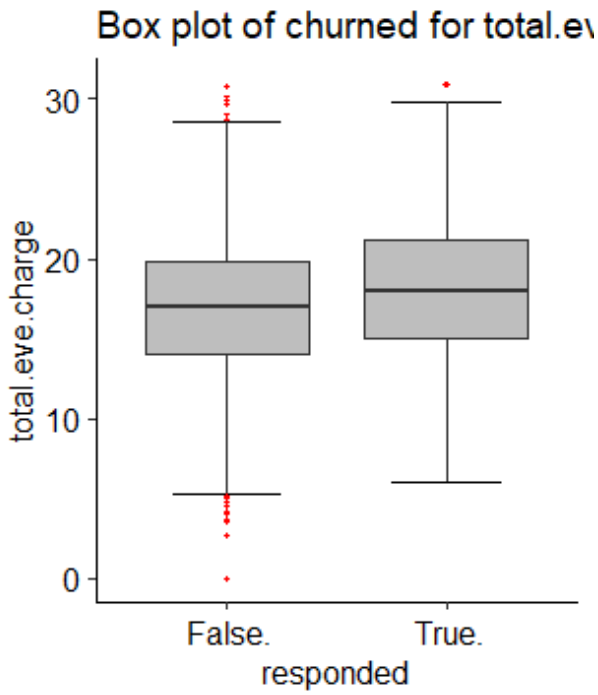
'state','area code', 'total day minutes', 'total eve minutes', 'total night minutes', 'total intl minutes','phone number'

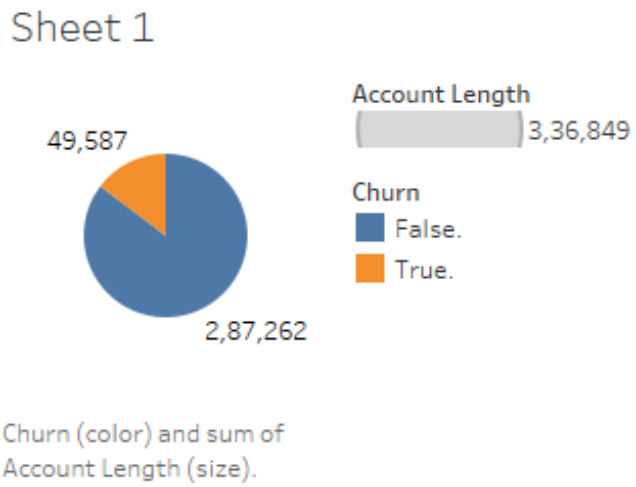
2.1.5 Analysis of churn ratio with different predictors


Box plot of churn
account.length
250
200
150
100
50
0
False.
True.
responded
Box plot of churn
total.day.calls
150
100
50
0
False.
True.
responded
Box plot of chur
phone.number
3000
2000
1000
0
False.
True.
responded

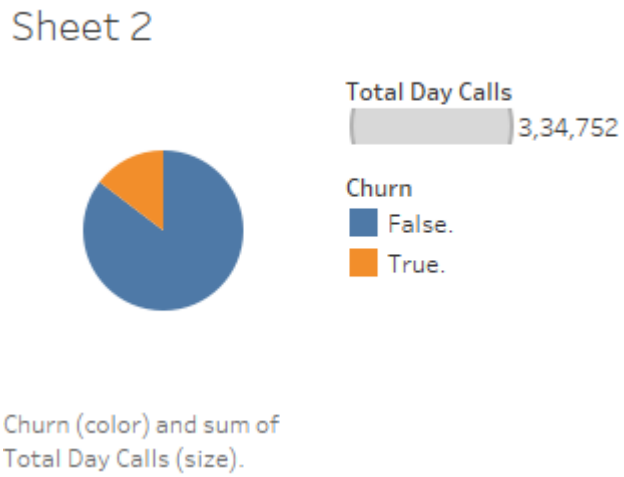

Box plot of churned for total.da
total.day.charge
60
40
20
0
False.
True.
responded
Box plot of churned for total.e
total.eve.minutes
300
200
100
0
False.
True.
responded


Box plot of churned for total.e
total.eve.calls
150
100
50
0
False.
True.
responded

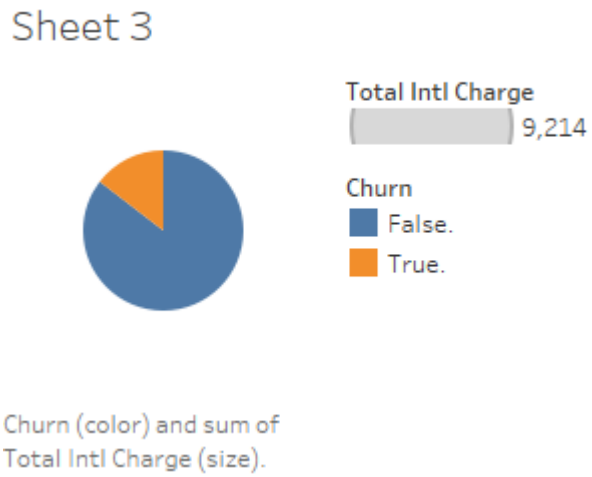

Box plot of churned for total.ev
total.eve.charge
30
20
10
0
False.
True.
responded



Churn (color) and sum of Account Length (size).



Churn (color) and sum of Total Day Calls (size).



Churn (color) and sum of Total Intl Charge (size).

2.1.6 Outlier analysis

Box plots are plotted using ggplot and hence, outliers are detected. They are replaced by null values and then applied with missing value analysis. Since we dint find many extreme outliers in the dataset, this process was ignored.

2.2 Modelling

Customer churn reduction is a binary classification problem. Here we have to build a model which can classify if a customer will churn out of not. In our dataset, dependent variable 'churn' was binary. So two machine learning algorithms were selected for learning.

1. Random Forest
2. Logistic regression

Both training models logistic regression and random forest were implemented in R and python. After building an initial model, performance tuning was done using hyperparameter tuning for optimised parameters.

2.2.1 random forest

Random forest model was first implemented in both R and python. We got satisfactory results using random forest. The code used for Random forest for R and python is given below.

R implementation

In R , the code used was as follows.

The accuracy achieved was, 94.66%

The false negative rate achieved was about, 31.25

The code is as follows,

```
library(randomForest)

RF_model = randomForest(Churn ~ ., Train_data, importance = TRUE, ntree = 1000)

#Extract rules fromn random forest

#transform rf object to an inTrees' format

library(RRF)

library(inTrees)

treeList = RF2List(RF_model)

#

# #Extract rules

exec = extractRules(treeList, Train_data[,-16])  # R-executable conditions

# #Make rules more readable:

readableRules = presentRules(exec, colnames(Train_data))

# #Get rule metrics

ruleMetric = getRuleMetric(exec, Train_data[,-16], Train_data$Churn)  # get rule metrics
```

```
#Presdict test data using random forest model

RF_Predictions = predict(RF_model, Test_data[,-16])


##Evaluate the performance of classification model

ConfMatrix_RF = table(Test_data$Churn, RF_Predictions)

confusionMatrix(ConfMatrix_RF)


#False Negative rate

FNR = FN/FN+TP


#Accuracy = 94.66

#FNR = 31.25
```

Python implementation

In Python , the code used was as follows.

The accuracy achieved was, 94.75%

The false negative rate achieved was about, 28.26

The code is as follows,

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF_model = RandomForestClassifier(n_estimators = 20).fit(X_train, y_train)
                                                                    In [38]:
RF_Predictions = RF_model.predict(X_test)
                                                                    In [39]:
#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)

#Accuracy: 94.75%
FNR: 28.26
```

2.2.2 Logistic regression

Logistic algorithm works on numeric data. So categorical data ie. International plan and voice mail plan and target predictor 'Churn' were encoded in to binary encoding. In case if international plan and voice mail plan, yes was coded as 1 and no as 0. In case of Churn True was encoded as 1 and False as 0. True/1 means customer churned and False/0 means it stayed.

R implementation

In R , the code used was as follows.

The accuracy achieved was, 90.89%

The false negative rate achieved was about, 67.85

The code is as follows,

```
#Logistic Regression
logit_model = glm(Churn ~ ., data = Train_data, family = "binomial")


#summary of the model
summary(logit_model)


##################predict using logistic regression#################
logit_Predictions = predict(logit_model, newdata = Test_data, type = "response")


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

```
##Evaluate the performance of classification model

ConfMatrix_LR = table(Test_data$Churn, logit_Predictions)

confusionMatrix(ConfMatrix_LR)


#False Negative rate

FNR = FN/FN+TP


#Accuracy: 90.89

#FNR: 67.85

set.seed(4543)

Train.rf <- randomForest(Churn ~ ., data=Train_data, ntree=1000, keep.forest=FALSE,

                         importance=TRUE)

varImpPlot(Train.rf)
```

Python implementation

In Python , the code used was as follows.

The accuracy achieved was, 87.40%

The false negative rate achieved was about, 79.01

The code is as follows,

```
#logistic regression
import numpy as np
import matplotlib.pyplot as plt

                                                                    In [16]:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()

                                                                    In [18]:
logisticRegr.fit(X_train, y_train)

                                                                    In [20]:
logisticRegr.predict(X_test[0:13])

                                                                    In [21]:
predictions = logisticRegr.predict(X_test)

                                                                    In [22]:
```

```
score = logisticRegr.score(X_test, y_test)
print(score)
#0.8740629685157422
```

```
0.8740629685157422
```

In [23]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
```

In [24]:

```
cm = metrics.confusion_matrix(y_test, predictions)
print(cm)
#FNR = 79.01
```

```
[[566  20]
 [ 64  17]]
```

NOTE:

Along with these methods, I tried other methods like KNN method and naïve bayes method with the following results

KNN Method:

Accuracy: 87.10%

False negative rate:97

Naïve bayes method:

Accuracy: 86.95%

False negative rate: 64.36

Since these methods don't yield satisfactory results, we reject these methods.

CHAPTER 3

Result and performance measure

Test data set was pre-processed in the same way as the train data was processed.

3.1 R results

The R results of various models are as follows,

Random Forest:

Accuracy: 94.66%

False negative Rate: 31.25

Logistic regression:

Accuracy: 90.89%

False negative rate: 67.85

3.2 Python results

The Python results of various models are as follows,

Random Forest: 94.75%

Accuracy: 28.26

False negative Rate:

Logistic regression:

Accuracy: 87.40%

False negative rate: 79.01

As we know, accuracy is a measure of how accurate our model is. The accuracy percentage tells us the percentage of times, our model has proved to be correct. Hence higher the accuracy, better is our model. Another important feature considered here is, false negative rate. The false negative tells us the number of times, we have declared the results to be negative but It is proved to be wrong. Hence lesser the false negative rate, the better is our model.

Now, After comparing all the required performance measure, random forest was performing overall better than logistic regression.

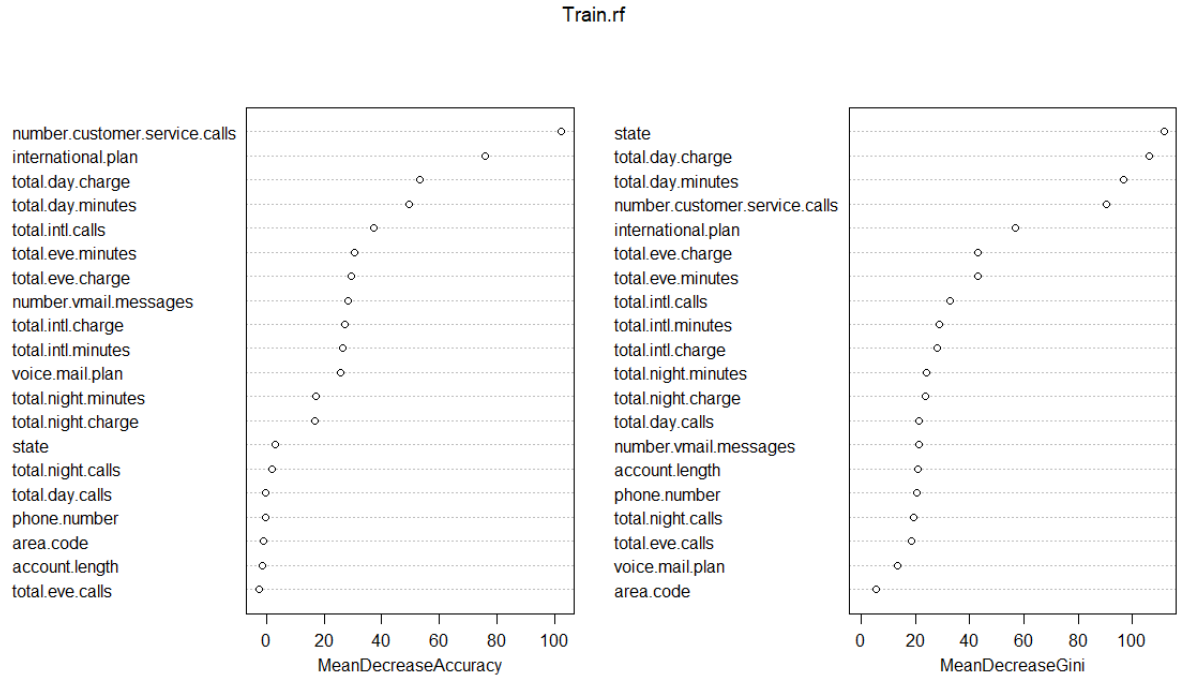
And also, on comparing the results of R and python, we can tell that our R model is performing better. Hence our random forest model of R is accepted.

Implementation

After seeing the various visuals, we can predict the following,

1. **Only 11.49 % customer with national plan churn out. 42.42 % customers with international plan had churn out. It means that the telecom company. Hence, is mainly loosing customers with international plans.**
2. **Customers without voice plan have higher churn rate**
3. **Customers with greater account length have less probability to churn out.**
4. **Higher the amount of day calls, lesser is the probability to churn out.**
5. **Higher is the international calls, lesser is the probability to churn out.**

Variable importance plot



COMPLETE R CODE

```
#remove all the objects stored

rm(list=ls())

#set current working directory

setwd("C:/Users/SHRAVYA/Desktop/edwisor/project 2")

#install packages

install.packages(c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced",
"C50", "dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE", "sampling",
"DataCombine", "inTrees"))

library(readxl)     # Super simple excel reader

library(mice)       # missing values imputation

library(naniar)     # visualize missing values

library(dplyr)

library(corrplot)

library(ggplot2)

library(tidyverse)

library(randomForest)

library(caret)

library(data.table)

library(Boruta)

library(rpart)

## Read the data

Train_data = read.csv("Train_data.csv", header = T, na.strings = c(" ", "", "NA"))

Test_data = read.csv("Test_data.csv", header = T, na.strings = c(" ", "", "NA"))

##############################exploratory data
analysis##############################

#studying the structure of the given dataset

str(Train_data)

#As we can see, few variables have a wrongly placed datatype.
```

```
#The variable , phone.number is actually meant to be a continuoos variable but it is analysed as a factor variable with 3333 levels. hence it has to be altered

#The variable area.code has to be a factor variable.

Train_data$phone.number  = as.numeric(Train_data$phone.number )

Test_data$phone.number  = as.numeric(Test_data$phone.number )

Train_data$area.code  = as.factor(Train_data$area.code )

Test_data$area.code  = as.factor(Test_data$area.code )

str(Train_data)

###################################missing value analysis############################

missing_val = data.frame(apply(Train_data,2,function(x){sum(is.na(x))}))

missing_val$Columns = row.names(missing_val)

names(missing_val)[1] =  "Missing_percentage"

missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(Train_data)) * 100

missing_val = missing_val[order(-missing_val$Missing_percentage),]

row.names(missing_val) = NULL

missing_val = missing_val[,c(2,1)]

write.csv(missing_val, "Missing_perc.csv", row.names = F)

sum(is.na(Train_data))

sum(is.na(Test_data))

#as we can see, there are no missing values. Hence no missing value imputation is requirede.calls, Churn)

###############################################variable analysis#########################

library(ggplot2)

library(ggpubr)

theme_set(theme_pubr())

ggplot(Train_data, aes(Churn)) +

 geom_bar(fill = "#0073C2FF") +

 theme_pubclean()
```

```
#outlier analysis
numeric_index = sapply(Train_data,is.numeric) #selecting only numeric


numeric_data = Train_data[,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_data))+
         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="responded")+
          ggtitle(paste("Box plot of churned for",cnames[i])))
}


# ##Plotting plots together
gridExtra::grid.arrange(gn1,gn5,gn2,ncol=3)
gridExtra::grid.arrange(gn6,gn7,ncol=2)
gridExtra::grid.arrange(gn8,gn9,ncol=2)
#feature selection
################################### Correlation Plot - to check multicolinearity
between continous variables
library(corrgram)
corrgram(Train_data[,numeric_index], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
```

```
#few independent variables are found to be highly correlated to each other. hence one of each copy is eliminated

## Chi-squared Test of Independence-to check the multicolinearity between categorical variables

factor_index = sapply(Train_data,is.factor)

factor_data = Train_data[,factor_index]

for (i in 1:5)

{

 print(names(factor_data)[i])

 print(chisq.test(table(factor_data$Churn,factor_data[,i])))

}

#the variable area code has a p value of greater than 0.05. Hence we eliminate the variable

## Dimension Reduction

Train_data = subset(Train_data,

                    select = -c(area.code,total.day.minutes,total.eve.minutes, total.night.minutes, total.intl.minutes))

Test_data = subset(Test_data,

                   select = -c(area.code,total.day.minutes,total.eve.minutes, total.night.minutes, total.intl.minutes))

####################################model developement#################################

#Clean the environment

library(DataCombine)

rm(list= ls()[!(ls() %in% c('Train_data','Test_data'))])

##Decision tree for classification

#Develop Model on training data

install.packages("C50")

library(C50)

C50_model = C5.0(Churn ~., Train_data, trials = 100, rules = TRUE)

#Summary of DT model

summary(C50_model)
```

```
#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")
#Lets predict for test cases
C50_Predictions = predict(C50_model, Test_data[,-16], type = "class")
##Evaluate the performance of classification model
library(caret)
ConfMatrix_C50 = table(Test_data$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)
#accuracy=95.86%
#false negetive rate=27.67%
##########################Random Forest#####################################
library(randomForest)
RF_model = randomForest(Churn ~ ., Train_data, importance = TRUE, ntree = 1000)
#Extract rules fromn random forest
#transform rf object to an inTrees' format
library(RRF)
library(inTrees)
treeList = RF2List(RF_model)
#
# #Extract rules
exec = extractRules(treeList, Train_data[,-16])  # R-executable conditions
# #Make rules more readable:
readableRules = presentRules(exec, colnames(Train_data))
# #Get rule metrics
ruleMetric = getRuleMetric(exec, Train_data[,-16], Train_data$Churn)  # get rule metrics
#Presdict test data using random forest model
RF_Predictions = predict(RF_model, Test_data[,-16])


##Evaluate the performance of classification model
```

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


#False Negative rate
FNR = FN/FN+TP


#Accuracy = 94.66
#FNR = 31.25
#Logistic Regression
logit_model = glm(Churn ~ ., data = Train_data, family = "binomial")


#summary of the model
summary(logit_model)


#####################################predict using logistic
regression##################################
logit_Predictions = predict(logit_model, newdata = Test_data, type = "response")


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



##Evaluate the performance of classification model
ConfMatrix_LR = table(Test_data$Churn, logit_Predictions)
confusionMatrix(ConfMatrix_LR)


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

```
#Accuracy: 90.89
#FNR: 67.85
set.seed(4543)
Train.rf <- randomForest(Churn ~ ., data=Train_data, ntree=1000, keep.forest=FALSE,
                         importance=TRUE)
varImpPlot(Train.rf)
```

COMPLETE PYTHON CODE

```
#Load libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
                                                                    In [6]:
#Set working directory
os.chdir("C:/Users/SHRAVYA/Desktop/edwisor/project 2")
                                                                    In [7]:
#Load data
Train_data= pd.read_csv("Train_data.csv")
Test_data = pd.read_csv("Test_data.csv")
                                                                    In [8]:
#----------------------------------PRE PROCESSING-EXPLORATORY DATA ANALYSIS
-----------------------------------------------------#
#As we can see, few variables have a wrongly placed datatype.
#The variable , phone.number is actually meant to be a continuoos variable
but it is analysed as a factor variable with 3333 levels. hence it has to b
e altered
#The variable area.code has to be a factor variable.
Train_data['area code']=Train_data['area code'].astype(object)
Test_data['area code']=Test_data['area code'].astype(object)
                                                                    In [5]:
#-------------------------------------MISSING VALUE ANALYSIS-------------
-----------------------------------------------------#
#Create dataframe with missing percentage
missing_val = pd.DataFrame(Train_data.isnull().sum())

#Reset index
missing_val = missing_val.reset_index()

#Rename variable
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missi
ng_percentage'})

#Calculate percentage
```

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

#descending order
missing_val = missing_val.sort_values('Missing_percentage', ascending = Fal
se).reset_index(drop = True)

#save output results
missing_val.to_csv("Missing_perc.csv", index = False)
Train_data.dtypes
missing_val
Train_data.isnull().sum()
Test_data.isnull().sum()
```

In [9]:

```
#As we can see, no missing values are present. Hence no missing analysis is
required.
Train_data.dtypes
```

In [7]:

```
#Plot boxplot to visualize Outliers
%matplotlib inline
plt.boxplot(Train_data['account length'])
```

In [8]:

```
%matplotlib inline
plt.boxplot(Train_data['number vmail messages'])
```

In [9]:

```
%matplotlib inline
plt.boxplot(Train_data['total day minutes'])
```

In [10]:

```
%matplotlib inline
plt.boxplot(Train_data['total day calls'])
```

In [11]:

```
%matplotlib inline
plt.boxplot(Train_data['total day charge'])
```

In [12]:

```
%matplotlib inline
plt.boxplot(Train_data['total eve minutes'])
```

In [13]:
```
%matplotlib inline
plt.boxplot(Train_data['total eve calls'])
```

In [14]:
```
%matplotlib inline
plt.boxplot(Train_data['total eve charge'])
```

In [15]:
```
%matplotlib inline
plt.boxplot(Train_data['total night minutes'])
```

In [16]:
```
%matplotlib inline
plt.boxplot(Train_data['total night calls'])
```

In [17]:
```
%matplotlib inline
plt.boxplot(Train_data['total night charge'])
```

In [18]:
```
%matplotlib inline
plt.boxplot(Train_data['total intl minutes'])
```

In [19]:
```
%matplotlib inline
plt.boxplot(Train_data['total intl calls'])
```

In [20]:
```
%matplotlib inline
plt.boxplot(Train_data['total intl charge'])
```

In [21]:
```
%matplotlib inline
plt.boxplot(Train_data['number customer service calls'])
```

In [22]:
```
#save numeric names
cnames =  ["account length", "number vmail messages", "total day minutes",
"total day calls", "total day charge", "total eve minutes", "total eve call
s", "total eve charge",
```

```
          "total night minutes", "total night calls", "total night charge"
,"total intl minutes","total intl calls","total intl charge","number custom
er service calls"]
```

In [23]:

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

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

     Train_data = Train_data.dropTrain_data[Train_data.loc[:,i] < min].inde
x)
     Train_data = Train_data.drop(Train_data[Train_data.loc[:,i] > max].ind
ex)
```

```
  File "<ipython-input-23-7d3fff23d9c6>", line 12
    Train_data = Train_data.dropTrain_data[Train_data.loc[:,i] < min].index
)

^
SyntaxError: invalid syntax
```

In [24]:

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

In [25]:

```
#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.divergi
ng_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

In [26]:

```
#Chisquare test of independence
#Save categorical variables
```

```
cat_names = ["state", "area code", "phone number", "international plan", "v
oice mail plan"]
```

In [27]:

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

```
state
0.002296221552011188
area code
0.9150556960243712
phone number
0.49185608455943547
international plan
2.4931077033159556e-50
voice mail plan
5.15063965903898e-09
```

In [28]:

```
Train_data.dtypes
Test_data.dtypes
```

```
dtype: object
```

In [10]:

```
Test_data = Test_data.drop(['state','area code', 'total day minutes', 'tota
l eve minutes', 'total night minutes', 'total intl minutes','phone number']
, axis=1)
```

In [11]:

```
Train_data = Train_data.drop(['state','area code', 'total day minutes', 'to
tal eve minutes', 'total night minutes', 'total intl minutes','phone number
'], axis=1)
```

In [12]:

```
Train_data['international plan'] = Train_data['international plan'].astype(
'category')
Train_data['voice mail plan'] = Train_data['voice mail plan'].astype('categ
ory')
Train_data['Churn'] = Train_data['Churn'].astype('category')
Test_data['international plan'] = Test_data['international plan'].astype('c
ategory')
Test_data['voice mail plan'] = Test_data['voice mail plan'].astype('categor
y')
Test_data['Churn'] = Test_data['Churn'].astype('category')
```

In [13]:

```
Train_data['international plan'] = (Train_data['international plan'] == ' y
es').astype(int)
```

```
Train_data['voice mail plan'] = (Train_data['voice mail plan'] == ' yes').a
stype(int)
Train_data['Churn'] = (Train_data['Churn'] == ' True.').astype(int)
Test_data['international plan'] = (Test_data['international plan'] == ' yes
').astype(int)
Test_data['voice mail plan'] = (Test_data['voice mail plan'] == ' yes').ast
ype(int)
Test_data['Churn'] = (Test_data['Churn'] == ' True.').astype(int)
```

In [15]:

```
#Divide data into train and test
X = Train_data.values[:, 0:13]
Y = Train_data.values[:,13]

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

In [35]:

```
#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  #http://webgraphviz.com/
# dotfile = open("pt.dot", 'w')
# df = tree.export_graphviz(C50_model, out_file=dotfile, feature_names = ma
rketing_train.columns)
```

In [36]:

```
#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)
#accuracy= 90.55%
#False Negative rate=19.0
(FN*100)/(FN+TP)
```

In [37]:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier

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

In [38]:

```
RF_Predictions = RF_model.predict(X_test)
```

In [39]:

```
#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)

#Accuracy: 94.75%
FNR: 28.26
```

In [46]:

```
#KNN implementation
from sklearn.neighbors import KNeighborsClassifier

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

In [47]:

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

In [50]:

```
#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)

#Accuracy: 87.10
#FNR: 97
```

In [51]:

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB

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

In [52]:

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

In [56]:

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

#Accuracy: 86.95
#FNR: 64.36
```

In [1]:

```
#logistic regression
import numpy as np
import matplotlib.pyplot as plt
```

In [16]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
```

In [18]:

```
logisticRegr.fit(X_train, y_train)
```

In [20]:

```
logisticRegr.predict(X_test[0:13])
```

In [21]:

```
predictions = logisticRegr.predict(X_test)
```

In [22]:

```
score = logisticRegr.score(X_test, y_test)
print(score)
#0.8740629685157422
```

```
0.8740629685157422
```

In [23]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
```

In [24]:

```
cm = metrics.confusion_matrix(y_test, predictions)
print(cm)
#FNR = 79.01
```

```
[[566  20]
 [ 64  17]]
```