Predicting Churn in Telecom

-Sarthak Gupta

CONTENTS

Chapter 1

INTRODUCTION

1.1 Problem Statement

Churn (loss of customers to competition) is a problem for most companies. In wake of high competition in telecom sector every company is facing it. The objective of the case is to predict customer behavior. Based on certain factors of usage pattern of a customer it is expected that it could be predicted whether the customer will churn or not.

1.2 Data

Our task is to build classification models and subsequently to predict on the basis of usage pattern whether a customer will churn or not. A sample of train and test dataset provided are given below.

Train Dataset :

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge
KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16.78
OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16.62
NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10.3
OH	84	408	375-9999	yes	no	0	299.4	71	50.9	61.9	88	5.26
OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12.61
AL	118	510	391-8027	yes	no	0	223.4	98	37.98	220.6	101	18.75
MA	121	510	355-9993	no	yes	24	218.2	88	37.09	348.5	108	29.62

total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
244.7	91	11.01	10	3	2.7	1	False.
254.4	103	11.45	13.7	3	3.7	1	False.
162.6	104	7.32	12.2	5	3.29	0	False.
196.9	89	8.86	6.6	7	1.78	2	False.
186.9	121	8.41	10.1	3	2.73	3	False.
203.9	118	9.18	6.3	6	1.7	0	False.
212.6	118	9.57	7.5	7	2.03	3	False.

Test Dataset :

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge
HI	101	510	354-8815	no	no	0	70.9	123	12.05	211.9	73	18.01
MT	137	510	381-7211	no	no	0	223.6	86	38.01	244.8	139	20.81
OH	103	408	411-9481	no	yes	29	294.7	95	50.1	237.3	105	20.17
NM	99	415	418-9100	no	no	0	216.8	123	36.86	126.4	88	10.74
SC	108	415	413-3643	no	no	0	197.4	78	33.56	124	101	10.54
IA	117	415	375-6180	no	no	0	226.5	85	38.51	141.6	68	12.04
ND	63	415	348-8073	no	yes	32	218.9	124	37.21	214.3	125	18.22
LA	94	408	359-9881	no	no	0	157.5	97	26.78	224.5	112	19.08

total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
236	73	10.62	10.6	3	2.86	3	False.
94.2	81	4.24	9.5	7	2.57	0	False.
300.3	127	13.51	13.7	6	3.7	1	False.
220.6	82	9.93	15.7	2	4.24	1	False.
204.5	107	9.2	7.7	4	2.08	2	False.
223	90	10.04	6.9	5	1.86	1	False.
260.3	120	11.71	12.9	3	3.48	1	False.
310.8	106	13.99	11.1	6	3	0	False.

Chapter 2

Methodology

2.1 Preprocessing

The very first thing in predictive model building is to have a look at the data and explore it. This is known as Exploratory Data Analysis.

2.1.1 Missing Value Analysis

We check for missing value in both train and test datset.

The following code is used for checking missing values in test and train dataset.

Python:

```
train.isnull().any()

test.isnull().any()
```

R:

```
is.na(data1)
table(is.na(data1))
is.na(test)
table(is.na(test))
```

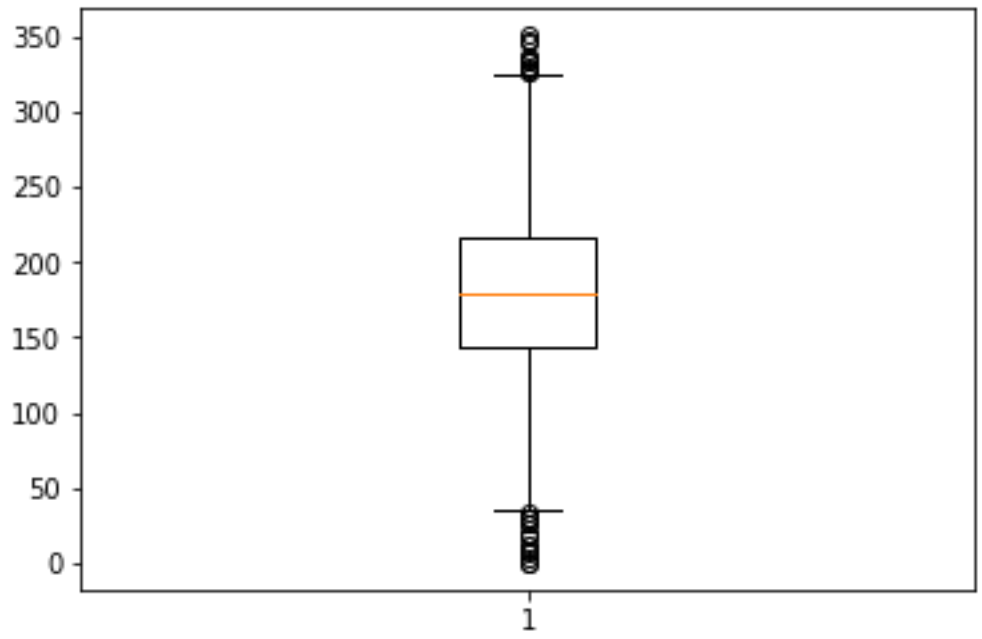
No missing values were found at both train and test dataset.

2.1.2 Outlier Analysis

Every piece of data has values which are inconsistent with the rest of the data. This implies that they are either too high or too low in comparison with majority of data. Such values have the potential of distorting the data, the model and therefore the prediction. Therefore, it is necessary that these are identified and deleted or substituted according to a suitable method.

Numeric Variables :

We go for boxplot method. First we plot boxplot for some variables to see if they have outliers. An example is given below. It is clear that many variables do have outliers. Then we use the boxplot method to remove the observations with outliers.



Python Code :

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.boxplot(train1['total day minutes'])
```

```
import numpy as np
df1 = train1.select_dtypes(include=[np.number])
```

```
#saving numeric names
cnames = df1.columns
```

```
#Detect and delete outliers from data
for i in cnames:
     print(i)
     q75, q25 = np.percentile(train1.loc[:,i], [75 ,25])
     iqr = q75 - q25

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

```
    print(max)

    train1 = train1.drop(train1[train1.loc[:,i] < min].index)
    train1 = train1.drop(train1[train1.loc[:,i] > max].index)
```

The R code:

```
#outlier analysis

numeric_index = sapply(data2,is.numeric) #selecting only numeric

numeric_data = data2[,numeric_index]

cnames = colnames(numeric_data)

#Delete the outliers using boxplot method

for(i in cnames){

print(i)

val = data2[,i][data2[,i] %in% boxplot.stats(data2[,i])$out]

#print(length(val))

data2 = data2[which(!data2[,i] %in% val),]

}
```

Now, we have a dataset free of outliers.

2.1.3 Feature Selection

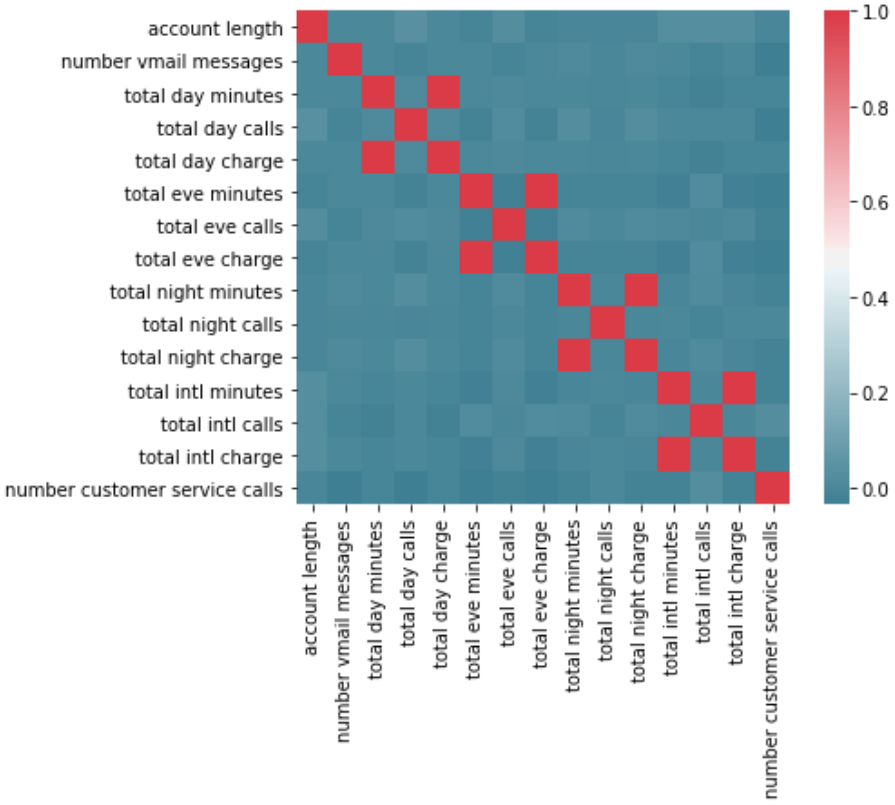
This is one of the most important steps. It has to be decided which variables are to be kept and which to be dropped for model building. The lesser number of variables we use the better as it reduces complexity.

At the preliminary stage we drop three variables of state, phone number and area code. These are dropped as they do not form part of the list of predictors in given problem statement, have a large number of levels.

For the rest of variables we have to go with different approach for numeric and categorical variables.

Numeric : We plot a correlation plot between independent variables. The variables having high correlation with another independent variables are deleted as they do not contribute anything extra to the model.


account.length
nber.vmail.messag
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
r.customer.service


account length
number vmail messages
total day minutes
total day calls
total day charge
total eve minutes
total eve calls
total eve charge
total night minutes
total night calls
total night charge
total intl minutes
total intl calls
total intl charge
number customer service calls
1.0
0.8
0.6
0.4
0.2
0.0

On the basis of this four variables are deleted :

i. Total day charge
ii. Total eve charge
iii. Total night charge
iv. Total intl charge

They have high co-relation with the corresponding variable for minutes. This is intuitive as well.

R code:

```
#Feature Selection
## Correlation Plot (for numeric varaiables)
library(corrgram)
corrgram(data2[,numeric_index])
#Deleting variables having high corelation with another independent
variable
data3 = subset(data2,select = -
c(total.day.charge,total.eve.charge,total.night.charge,total.intl.char
ge))
test1 = subset(test1,select = -
c(total.day.charge,total.eve.charge,total.night.charge,total.intl.char
ge))
```

Python Code :

```
#Feature Selection
##Correlation analysis
#Correlation plot
df_corr = train1.loc[:,cnames]
```

```
import seaborn as sns
#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.diverg
ing_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

```
#Deleting variables having high corelation with another independent variab
le
train_deleted = train1.drop(['total day charge','total eve charge','total
night charge','total intl charge'],axis=1)
```

```
test_deleted = test1.drop(['total day charge','total eve charge','total ni
ght charge','total intl charge'],axis=1)
```

Categorical Variable : For them we use chi-square test of independence. If the p value is less than 0.05, the variable is kept otherwise dropped. For both the categorical variables i.e. international plan and voice mail plan p<0.05. Therefore, no categorical variable is dropped.

R code :

```
## Chi-squared Test of Independence
factor_index = sapply(data3,is.factor)
factor_data = data3[,factor_index]
for (i in 1:2)
{
print(names(factor_data)[i])
print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}
```

Python code :

```
#Chisquare test of independence
#Save categorical variables
cat_names = ["international plan","voice mail plan"]
```

```
from scipy.stats import chi2_contingency
#loop for chi square values
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(train_deleted['Churn']
, train_deleted[i]))
    print(p)
```

2.2 Modelling

2.2.1 Model Selection

The dependent variable i.e. Churn has two classes true and false. It is not a continuous variable. Therefore, linear regression is out of the picture. We can go for Decision Tree, Random Forest, Logistic Regression and Naïve Bayes.

2.2.2 Decision Tree

We use Decision Tree model to build a model and make predictions. We use c5.0 model for this. It is based on information gain.

The R code :

```
library(C50)
c50_model = C5.0(Churn~.,data4,trials=100,rules=TRUE)
#summary of DT model
summary(c50_model)
#write rules into disk
write(capture.output(summary(c50_model)),"c50Rules.txt")
#Prediction on test data
c50_Predictions = predict(c50_model,test1[,-14],type = "class")
#Evaluate the performance of the model
ConfMatrix_c50 = table(test1$Churn,c50_Predictions)
library(caret)
confusionMatrix(ConfMatrix_c50)
#Accuracy is 93.88
```

Python Code :

```
#Import Libraries for decision tree
from sklearn import tree
from sklearn.metrics import accuracy_score
```

```
train_deleted['international plan'] = train_deleted['international plan'].
replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].
replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].re
place('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].re
place('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'yes', 1)
```

```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

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

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

```
#confusion matrix
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
((TP+TN)*100)/(TP+TN+FP+FN)
89.62207558488302
```

```
#False Negative rate
(FN*100)/(FN+TP)
42.857142857142854
```

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

2.2.3 Random Forest

In random forest we make subsets out of the dataset and make multiple decision trees. On the basis of combination by majority we decide the result. It converts a weak learner into a strong learner. Efficiency is increased.

The R code:

```
library(randomForest)

RF_model = randomForest(Churn~.,data4,importance = TRUE,ntree = 100)

#Extract rules from random forest

#Transform rf object to an inTrees' Format

library(inTrees)

treelist = RF2List(RF_model)

#Extract Rules

exec = extractRules(treelist,data4[,-14])

#Visualize some rules

exec[1:2,]

#Make rules more readable

readableRules = presentRules(exec,colnames(data4))

readableRules[1:2,]
```

#Get rule metrics

ruleMetric = getRuleMetric(exec,data4[,-14],data4$Churn)

#Evaluate few rules

ruleMetric[1:2,]

#Predict test data using random forest model

RF_Predictions = predict(RF_model,test1[,-14])

#Evaluate the performance

confMatrix_RF = table(test1$Churn,RF_Predictions)

confusionMatrix(confMatrix_RF)

#Accuracy is 92.8%

Python Code :

```
train_deleted['international plan'] = train_deleted['international plan'].
replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].
replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].re
place('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].re
place('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'yes', 1)
```

```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

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

```
RF_model = RandomForestClassifier(n_estimators = 250).fit(X_train, y_train
)
```

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

```
#confusion matrix
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
((TP+TN)*100)/(TP+TN+FP+FN)
#accuracy has improved over decision tree
```

```
93.34133173365326
```

```
#False Negative rate
(FN*100)/(FN+TP)
#FNR has increased over Decision Tree
```

```
48.660714285714285
```

2.2.4 Logistic Regression

Logistic Regression is used for dataset where dependent variable is categorical. It is converted into log of odds to use a regression analysis.

The R code :

```
logit_model = glm(Churn~.,data=data4,family = "binomial")

#summary of model

summary(logit_model)

#predict usng logistic regression

logit_Predictions = predict(logit_model,newdata = test1,type = "response")

#convert into probabilities

logit_Predictions = ifelse(logit_Predictions>0.5,1,0)

#Evaluate the performance
```

confMatrix_RF = table(test1$Churn,logit_Predictions)

#Accuracy = (1424+60)*100/(1424+60+19+164) = 89.02%

Python Code:

```
df2 = train_deleted.select_dtypes(include=[np.number])
cnames1 = df2.columns
```

```
#Making the data suitable for logistic regression
train_deleted['Churn'] = train_deleted['Churn'].replace('false.', 0)
train_deleted['Churn'] = train_deleted['Churn'].replace('true.', 1)
test_deleted['Churn'] = test_deleted['Churn'].replace('false.', 0)
test_deleted['Churn'] = test_deleted['Churn'].replace('true.', 1)
```

```
train_deleted['international plan'] = train_deleted['international plan'].
replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].
replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].re
place('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].re
place('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'yes', 1)
```

```
#Create logistic data. Save target variable first
train_logit = pd.DataFrame(train_deleted['Churn'])
test_logit = pd.DataFrame(test_deleted['Churn'])
```

```
#Add continous variables
train_logit = train_logit.join(train_deleted[cnames1])
test_logit = test_logit.join(test_deleted[cnames1])
```

```
##Create dummies for categorical variables in train data
cat_names = ["international plan", "voice mail plan"]

for i in cat_names:
    temp = pd.get_dummies(train_deleted[i], prefix = i)
```

```
    train_logit = train_logit.join(temp)
```

```
##Create dummies for categorical variables in test data
cat_names = ["international plan", "voice mail plan"]

for i in cat_names:
    temp = pd.get_dummies(test_deleted[i], prefix = i)
    test_logit = test_logit.join(temp)
```

```
#select column indexes for independent variables
train_cols = train_logit.columns[1:16]
```

```
#Built Logistic Regression
np.warnings.filterwarnings("ignore")
import statsmodels.api as sm

logit = sm.Logit(train_logit['Churn'], train_logit[train_cols]).fit()
```

```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
logit.summary()
```

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

```
#If value of probability is more than 0.5 then assign 1 or if less than 0.
5 then assifn 0 in the new variable
test_logit['ActualVal'] = 1
test_logit.loc[test_logit.Actual_prob < 0.5, 'ActualVal'] = 0
```

```
#Build confusion matrix
CM = pd.crosstab(test_logit['Churn'], test_logit['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
((TP+TN)*100)/(TP+TN+FP+FN)
89.02219556088782
```

```
#FNR
(FN*100)/(FN+TP)
73.21428571428571
```

2.2.5 Naïve Bayes

Naïve Bayes algorithm assumes independence of variables. That is why it is called Naïve. It iteratively learns on the basis of probability. Bayes Theorem is used for this. It is a very fast learner. In case of large data set and less time available Naïve Bayes is the algorithm to go to.

The R code :

```
library(e1071)

NB_model = naiveBayes(Churn~.,data =data4)

#prediction

NB_Prediction = predict(NB_model,test1[,1:13],type = 'class')

#confusion matrix

Conf_matrix = table(observed = test1[,14],predicted = NB_Prediction)

confusionMatrix(Conf_matrix)

#Accuracy is 88.84%
```

Python Code:

```
train_deleted['international plan'] = train_deleted['international plan'].
replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].
replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replac
e('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].re
place('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].re
place('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace(
'yes', 1)
```

```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

```
#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)
```

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

```
86.50269946010798
```

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

```
65.625
```

Chapter 3

Conclusion

3.1 Model Evaluation & Selection

The accuracy we get in Random Forest is the highest. But Decision Tree and Random Forest are at similar level However, the difference between the accuracy level of the four algorithms is not very pronounced.

Interpretation wise it is always easy to explain Decision Tree as it can be explained even to a non technical person. Logistic Regression, Naïve Bayes require some technical knowledge for understanding.

The problem with Decision Tree is that it is a weak learner.

But looking at the fact that Decision Tree is yielding one of the highest accuracy, lowest False Negative Rate it would be suitable to pick this as the model. However, for larger datasets of similar nature it might be beneficial to go for Random Forest which combines multiple decision trees in order to overcome being a weak learner.