**CHURN REDUCTION**

*done by* ***SAMUEL JOHN***

**CONTENTS**

1. **Introduction**
   1. Problem Statement
   2. Data
2. **Methodology**
   1. Pre-Processing
   2. Modeling
3. **Conclusion**
   1. Model Evaluation
   2. Model Selection
4. **Appendix A – Python Code**
5. **Appendix B – R Code**
6. **INTRODUCTION**
   1. **Problem Statement**

Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving.

The objective of this Case is to predict customer behaviour. We are provided with a public dataset that has customer usage pattern and if the customer has moved or not. We are expected to develop an algorithm to predict the churn score based on usage pattern.

* 1. **Data**

Our task is to create model that can predict customer churn based on the available data. We are provided with data for training and testing purposes. Below is a sample of data along with the variables.

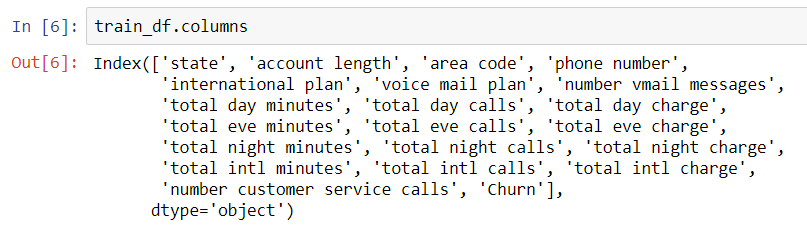
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| state | account length | area code | phone number | international plan | voice mail plan |
| KS | 128 | 415 | 382-4657 | no | yes |
| OH | 107 | 415 | 371-7191 | no | yes |
| NJ | 137 | 415 | 358-1921 | no | no |
| OH | 84 | 408 | 375-9999 | yes | no |
| OK | 75 | 415 | 330-6626 | yes | no |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| number vmail messages | total day minutes | total day calls | total day charge | total eve minutes | total eve calls |
| 25 | 265.1 | 110 | 45.07 | 197.4 | 99 |
| 26 | 161.6 | 123 | 27.47 | 195.5 | 103 |
| 0 | 243.4 | 114 | 41.38 | 121.2 | 110 |
| 0 | 299.4 | 71 | 50.9 | 61.9 | 88 |
| 0 | 166.7 | 113 | 28.34 | 148.3 | 122 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | Churn |
| 16.78 | 244.7 | 91 | 11.01 | 10 | 3 | 2.7 | 1 | False. |
| 16.62 | 254.4 | 103 | 11.45 | 13.7 | 3 | 3.7 | 1 | False. |
| 10.3 | 162.6 | 104 | 7.32 | 12.2 | 5 | 3.29 | 0 | False. |
| 5.26 | 196.9 | 89 | 8.86 | 6.6 | 7 | 1.78 | 2 | False. |
| 12.61 | 186.9 | 121 | 8.41 | 10.1 | 3 | 2.73 | 3 | False. |

The data is split into training dataset and test dataset, each having a number of variables of 21.

The column names are as below.



The target variable is Churn. We have to predict using the remaining variables, whether the customer had churned or not.

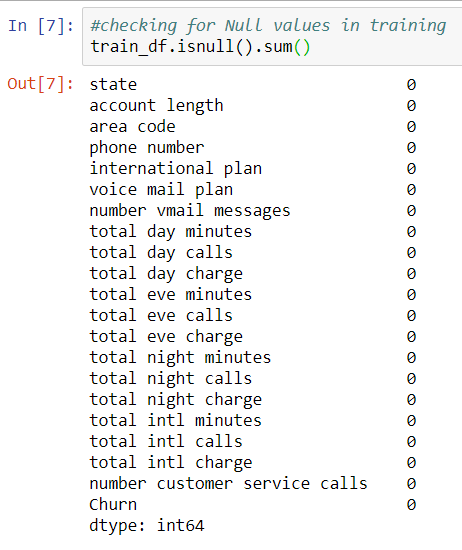
1. **METHODOLOGY**
   1. **Pre-Processing**

The initial step in any project is to clean the data that is handed over to us. There are various parts to it. This phase is also known as Exploratory Data Analysis. During this phase, we clean the data in such a way that it is suitable for giving as an input to the model.

For this project, the first pre-processing step that we tried to do was Missing Value Analysis.

Usually we check whether there are any missing values and treat them accordingly.

Using the below code, we checked whether there any missing values.



From the above snippet, we can see that number of missing values is 0. The same was done against the test dataset too and the results were the same.

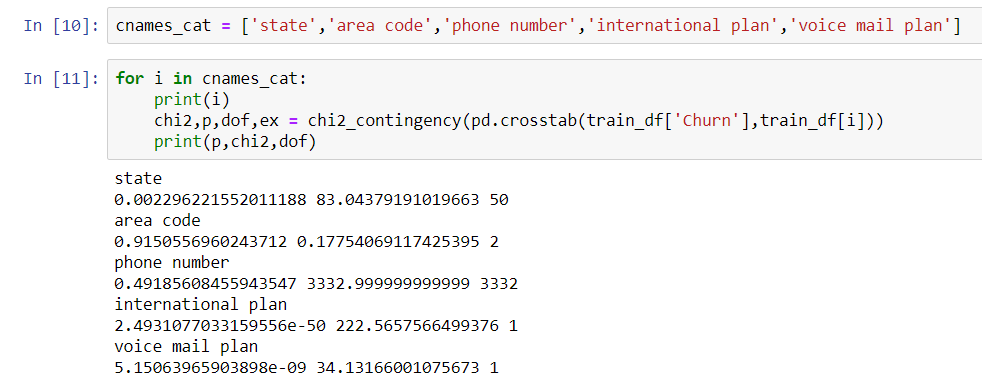
On checking the data, it was found that the variable ‘area code’ behaved like a categorical variable, even-though the data type was mentioned as numeric.

So, we converted it to object data-type using the below code snippet.



The next part of the pre-processing journey is the Feature Scaling. For performing feature scaling of the categorical variables, we are going to choose Chi-Square test.

For performing the chi-square test, we used the below code snippet.



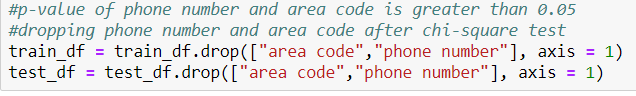
On the above snippet, we can see the probability after the chi-square test of each categorical value against the target variable.

If the probability is less than 0.05, we can reject the null hypothesis.

Null Hypothesis: The predictor and the target variable are independent.

Alternate Hypothesis: The predictor and the target variable are dependent.

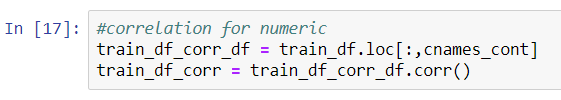
The probability of state, international plan, voice mail plan are less than 0.05. So, we can reject null hypothesis for these variables. The probability of ‘area code’ and ‘phone number’ are greater than 0.05, so we have to accept the null hypothesis for these variables. This means that ‘phone number’ and ‘area code’ are independent of the target variable ‘Churn’. Hence, we can remove the variables ‘phone number’ and ‘area code’ from our project.



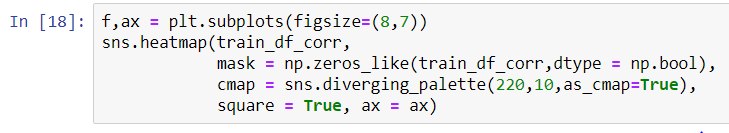
Next, we have to perform feature scaling for numeric variables. This can be done through Correlation Analysis.

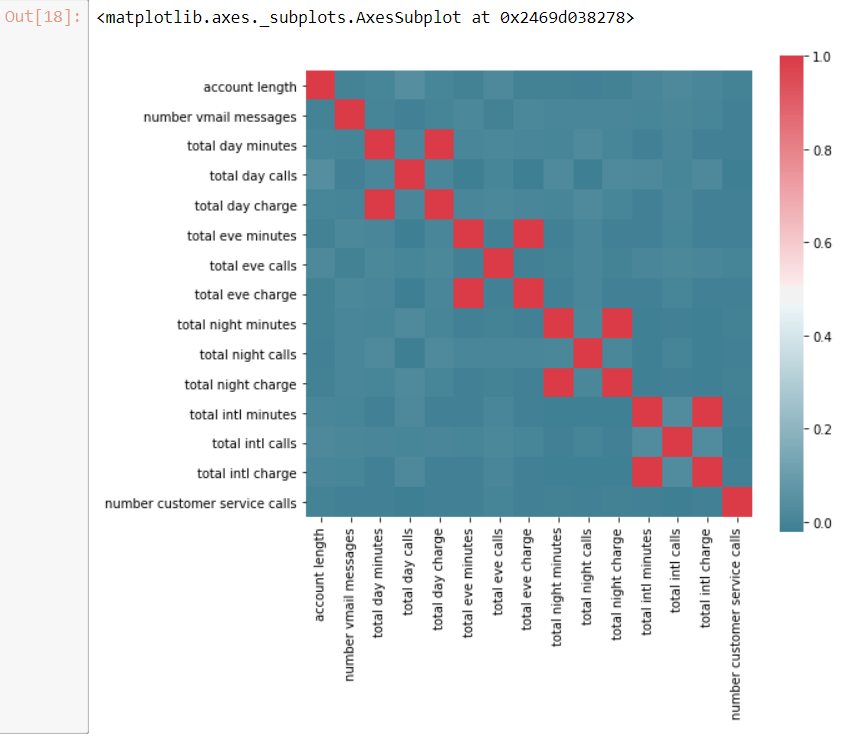
This was done as per the below code snippets.





The Correlation plot is below.

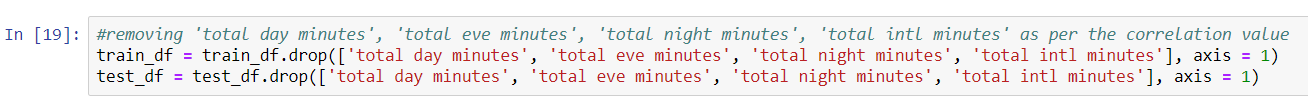




From the diagram, we can see that ‘total day minutes’ and ‘total day charge’ are correlated. The same can be said for ‘total eve minutes’ and ‘total eve charge’, ‘total night minutes’ and ‘total night charge’, ‘total intl minutes’ and ‘total intl charge’.

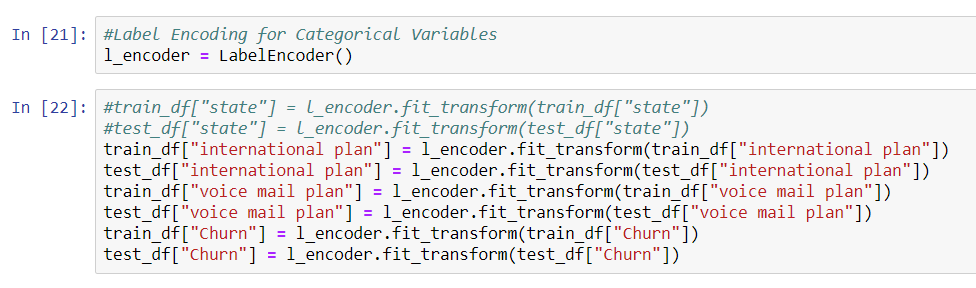
So we have to drop any one of the variables from these pairs of correlated variables. I decided to drop the ones ending with ‘minutes’.

I used the below code snippet to remove it.

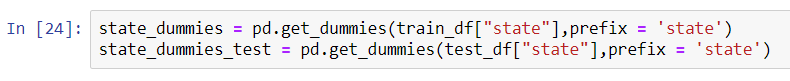


The next pre-processing technique which we did was encoding. This involves converting the character values to numerics. For variables with just two values, such as True or False, Yes or No, we can use Label Encoding. For variables which has more than 2 values, we can use dummy encoding.

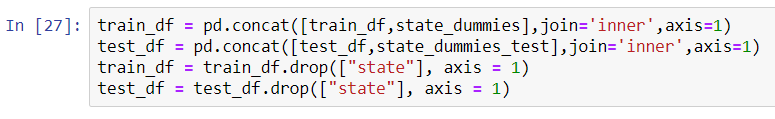
Below are the snippets of code I used for performing label encoding.



The variable ‘state’ has multiple character values. Hence, we performed dummy encoding for it.

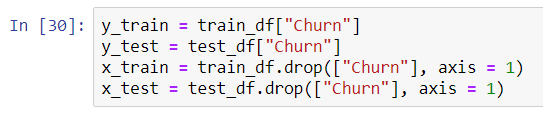


Then we removed the variable ‘state’ and then concatenated the dummy-variables to the training data.



* 1. **Modeling**

Before feeding the data to machine learning algorithms, we always have to split the data into training and test data. But in our case, already we have the training and test data. We have to remove the target variables from the training and test data and store it in a separate table.



1. **Decision Tree Model**

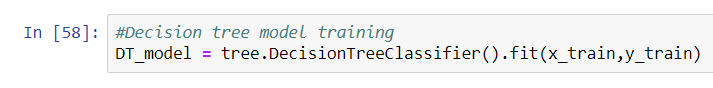
The first machine learning algorithm that we tried is Decision tree. It is an ML algorithm that follows tree-based learning. It’s a predictive model based on a branching series of Boolean tests.

The two most popular Decision-Tree Algorithms that are used are C5.0 and CART algorithms.

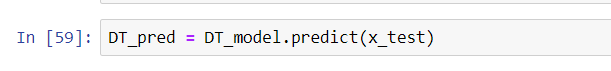
The metric used in C5.0 is Information Gain, while the metric used in CART algorithm is Gini Index.

The algorithm that we are going to use in this model is CART algorithm.

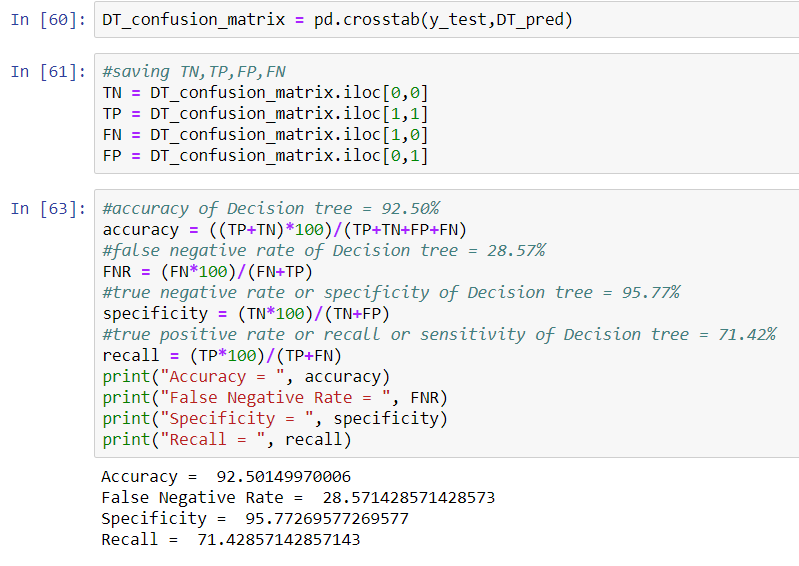
The model was trained by using the below code snippet.



The test dataset was predicted by using the below code snippet.



The confusion matrix is used to compare our predictions with the actual results. Confusion matrix along with few error metrics were measured using the below code snippets.

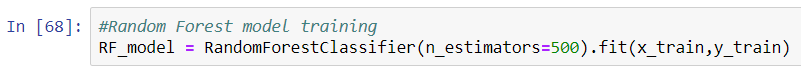


As per the above snippet, we can see that we have received an accuracy of 92.5%. Let’s try few more models and see the accuracy of them as well.

1. **Random Forest Model**

Random Forest is an ensemble technique. It consists of many decision trees. This method combines Breiman’s bagging idea and random selection of features.

Using the below code snippet, we train the Random Forest model.

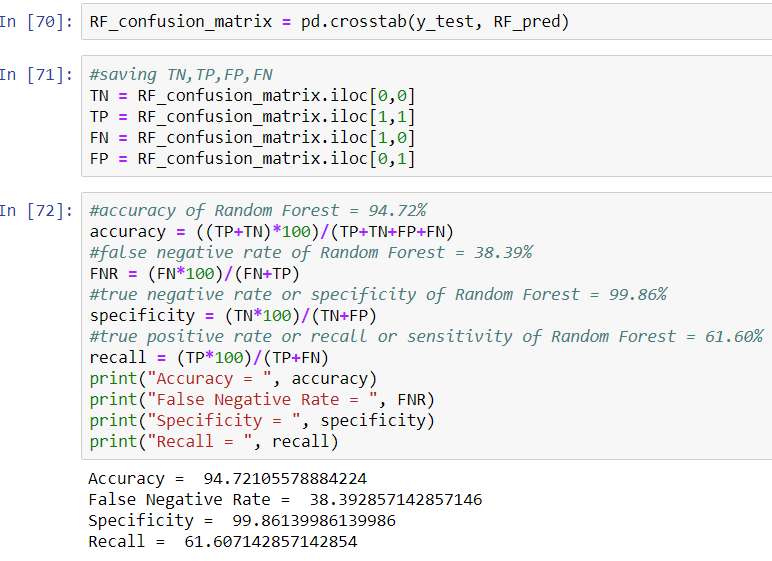


We should give the number of trees as quite low in the beginning. We should keep increasing the number of trees to see if the accuracy is increasing and select a suitable number of trees according to the better accuracy and other metrics.

Once the model is trained, we should predict the test data. Here, we did this using the below code snippet.



Next, we should compare our predictions with the actual output using the confusion matrix.



This model has given us a very high accuracy. Let’s check a few more algorithms.

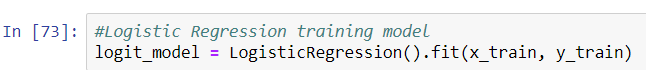
1. Logistic Regression model

Logistic Regression is a statistical model which can be used for classification problems.

It can be used to predict the probability of particular outcomes.

We have to assume that there is no multi-collinearity.

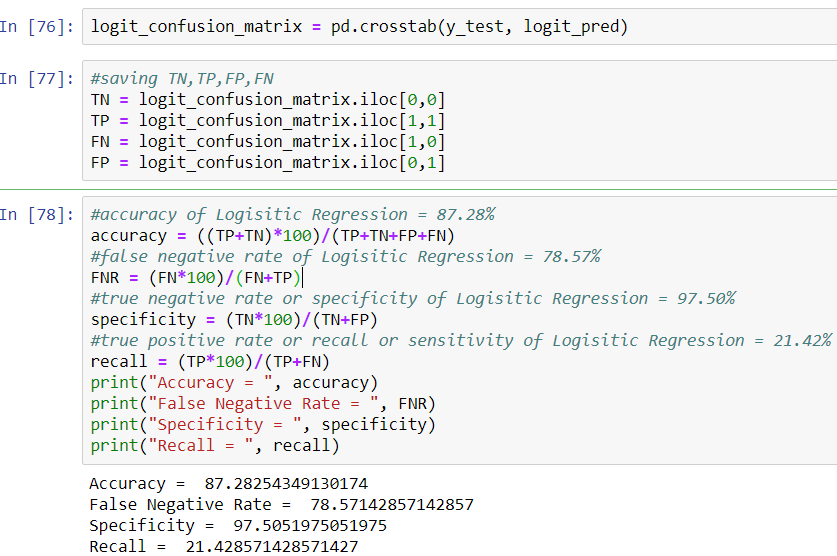
Using the below code snippet, we can train the logistic regression model.



Once the model is trained, we can use the below code snippet to test the test dataset.



After checking that the outputs are in the form of the actual results,i.e 0s and 1s,we have to compare the test results with the actual results using the confusion matrix.

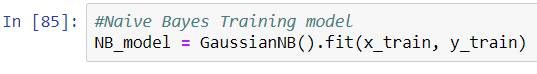


On checking the accuracy, it is lower than that of the tree models.

1. Naïve Bayes model

Naïve Bayes is a classification algorithm. It is based on probabilistic classification. It works on Bayes theorem of probability to predict the class of unknown dataset.

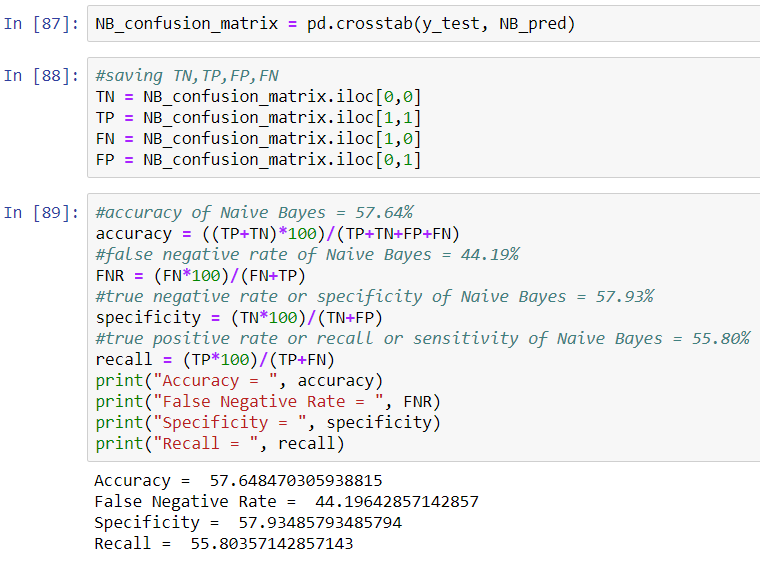
We can train the model using the below code snippet.



After the model is trained, we can test the test dataset using the below code snippet.



Next, we have to compare the actual test results with our predictions.

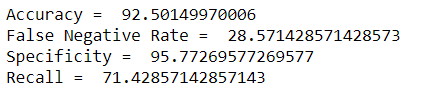


As per the above results, the accuracy is very low compared to the other models.

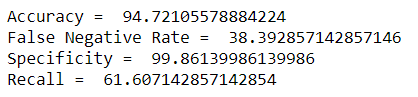
1. **CONCLUSION**

Metrics of the various models we used are as follows.

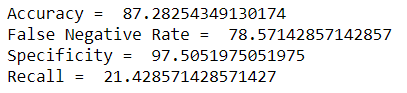
**Decision Tree:**



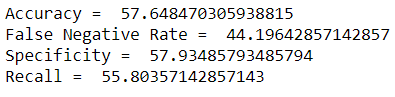
**Random Forest:**



**Logistic Regression:**



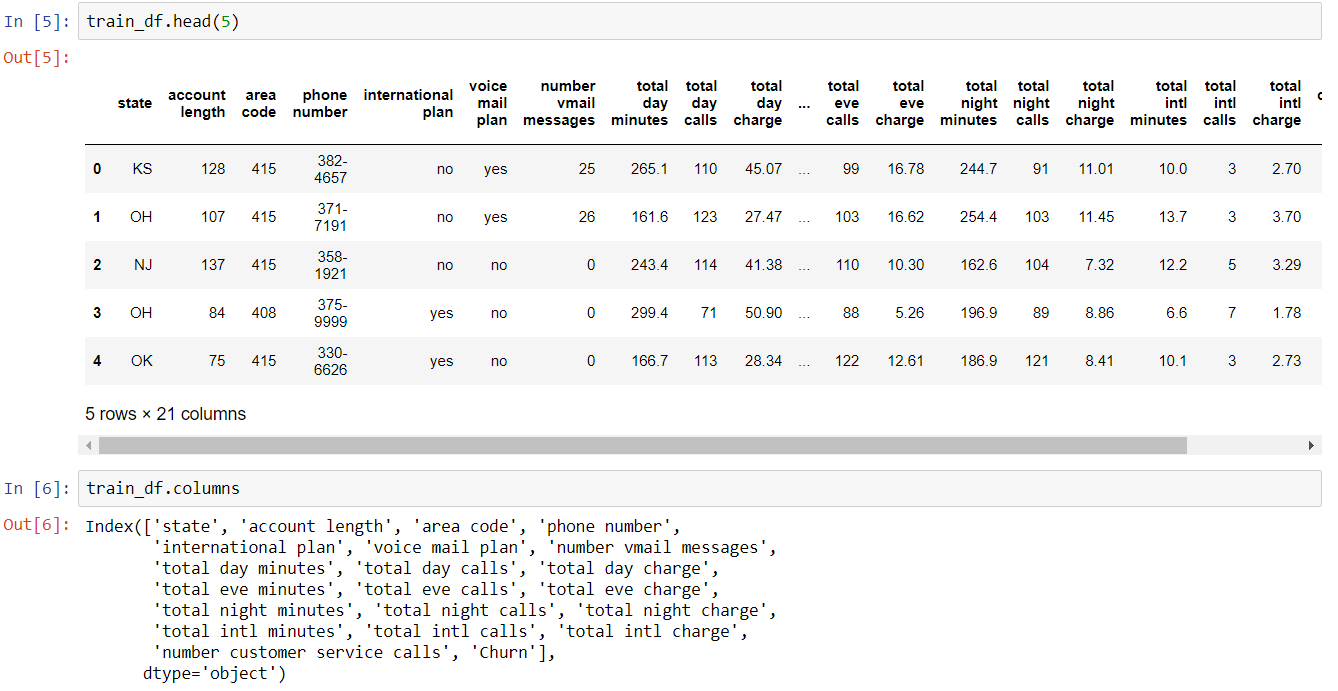
**Naïve Bayes:**

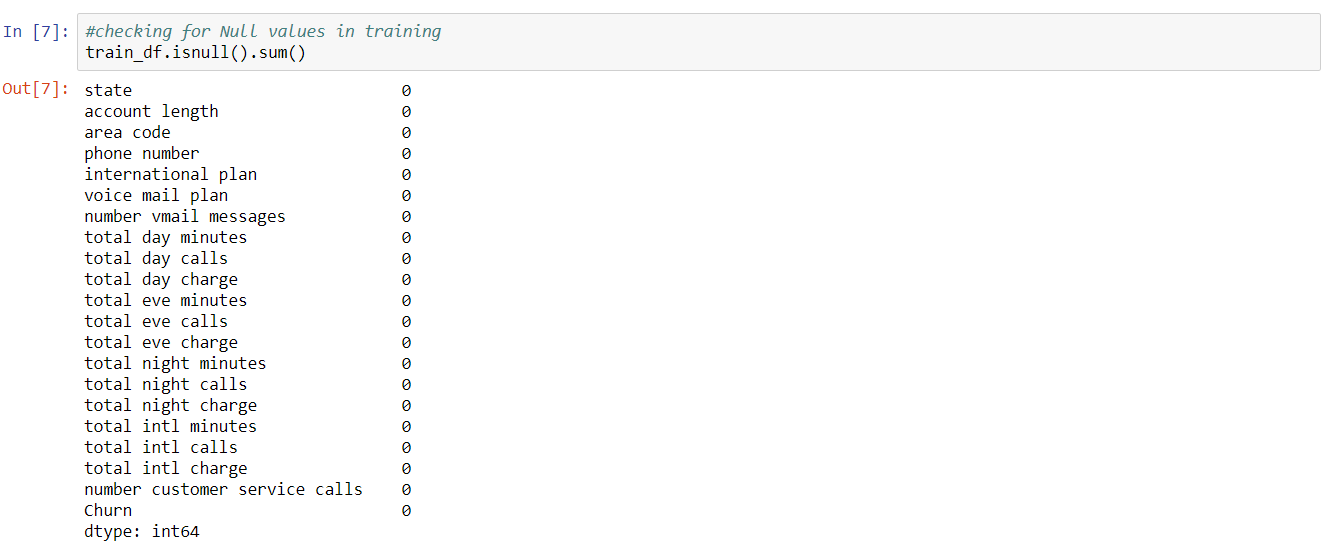


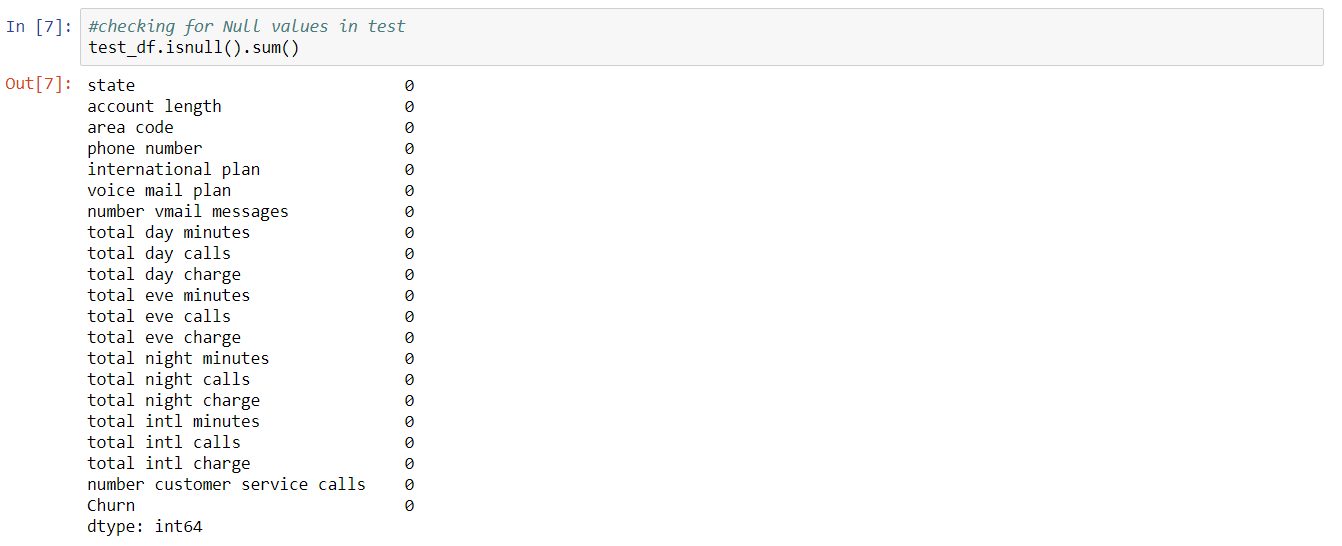
As per the above metrics, we can conclude that **Random Forest** is the better model as it has the better Accuracy and Specificity and along with a reasonable less False Negative Rate.

1. **APPENDIX A – PYTHON CODE**

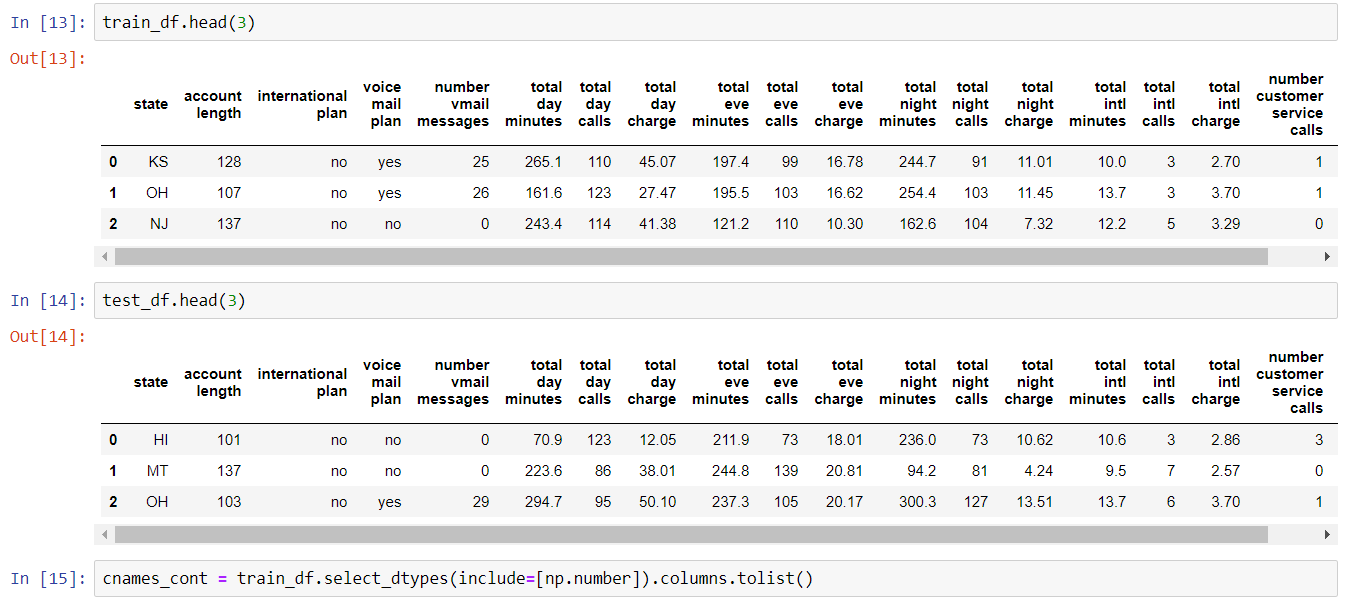
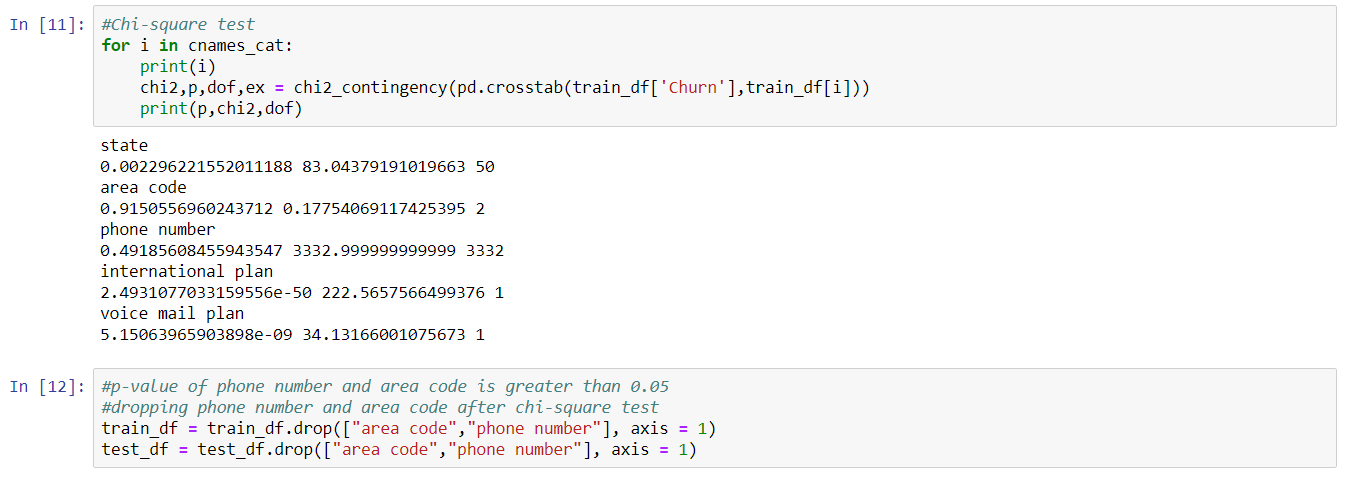


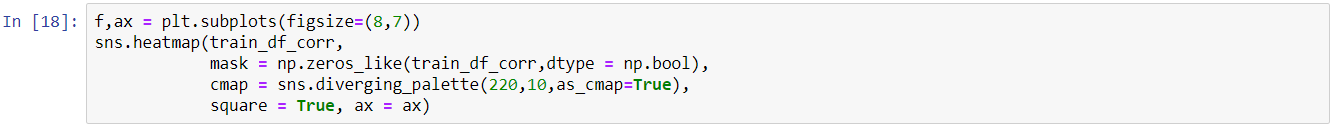


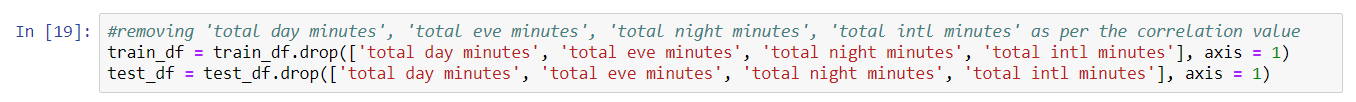
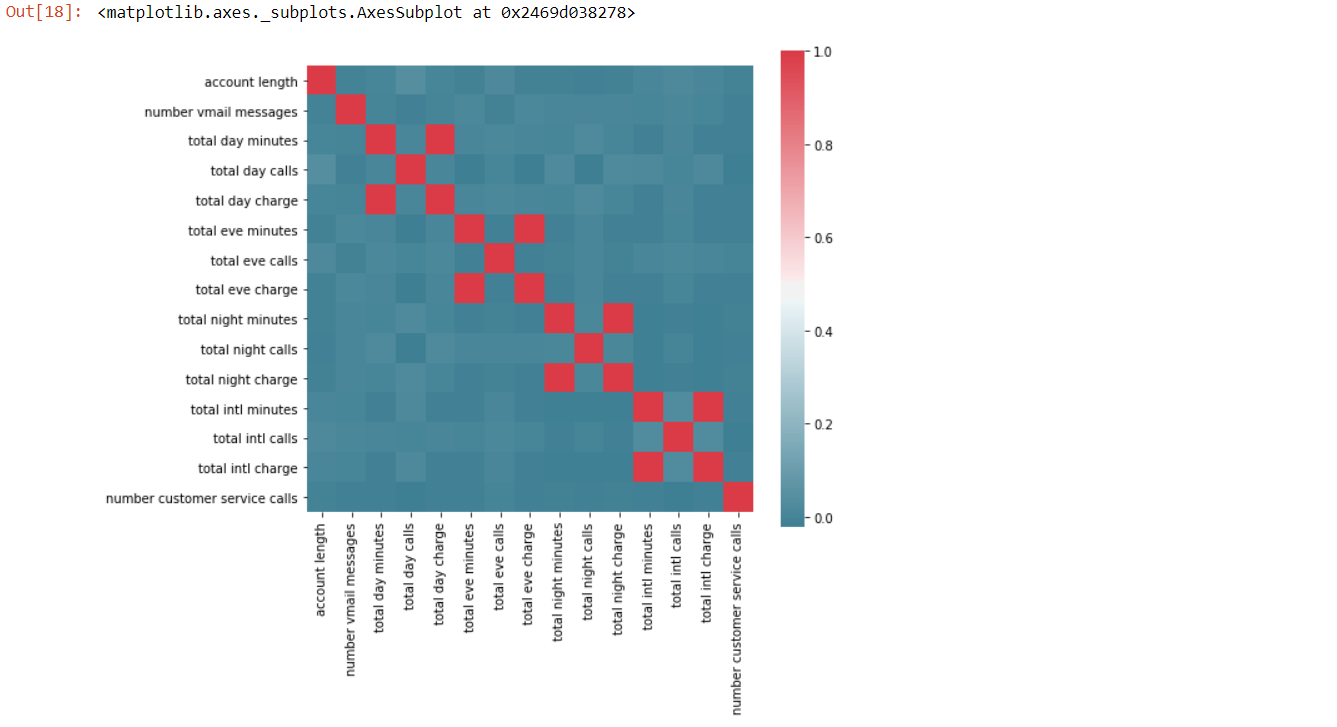


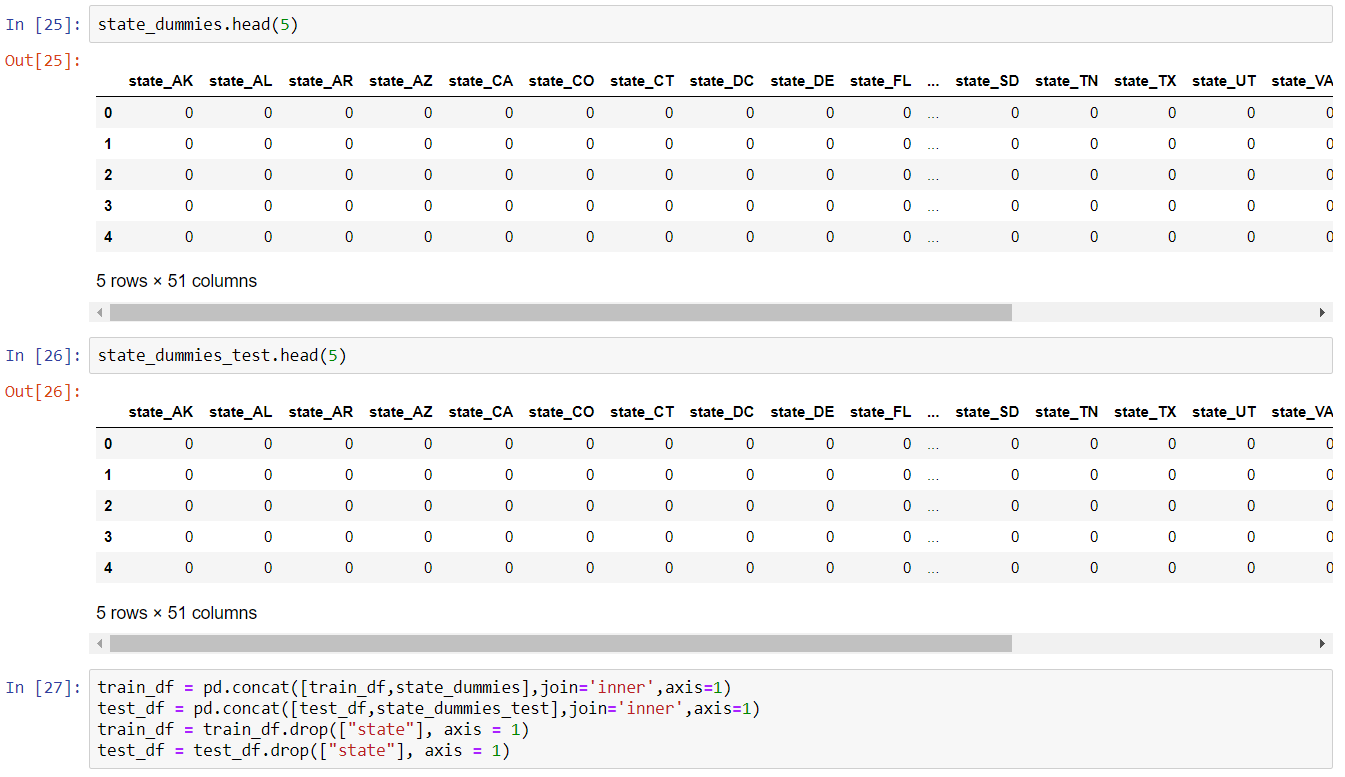


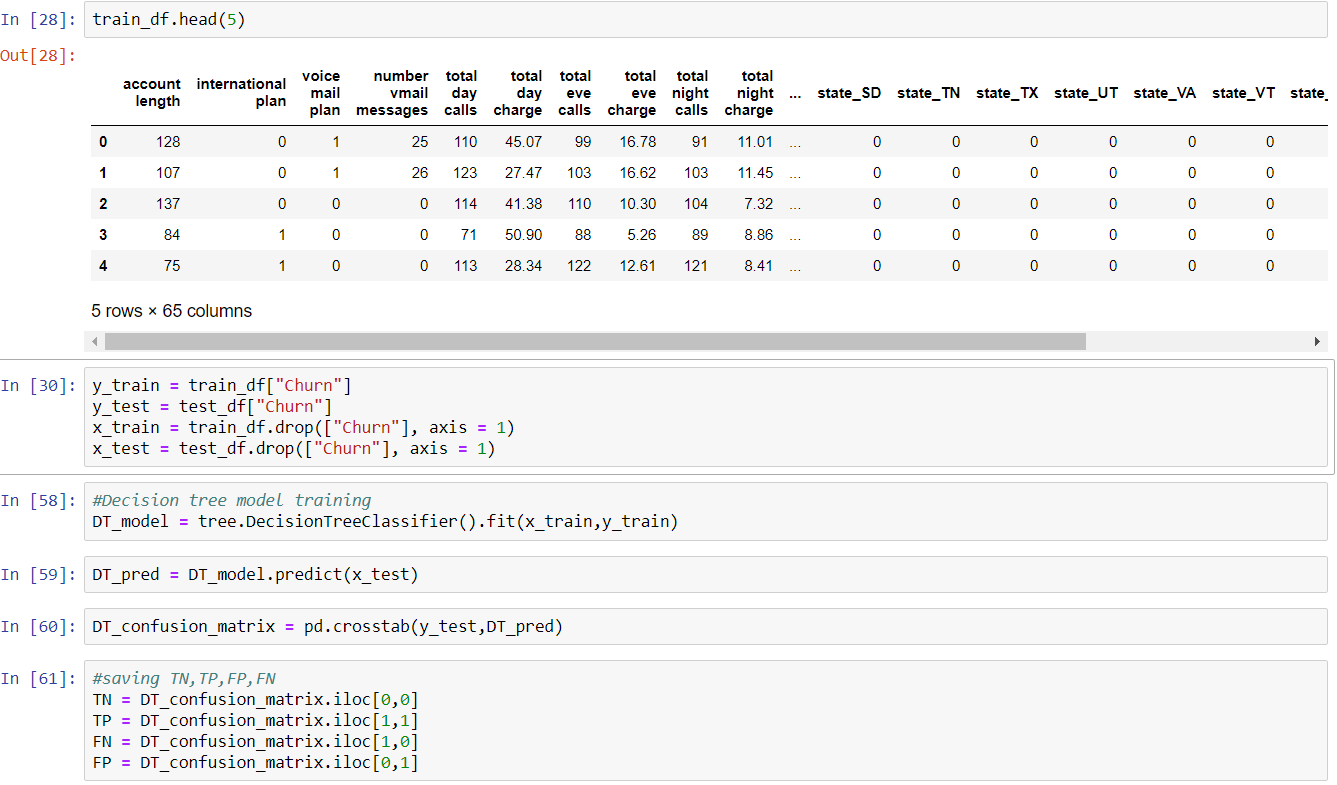


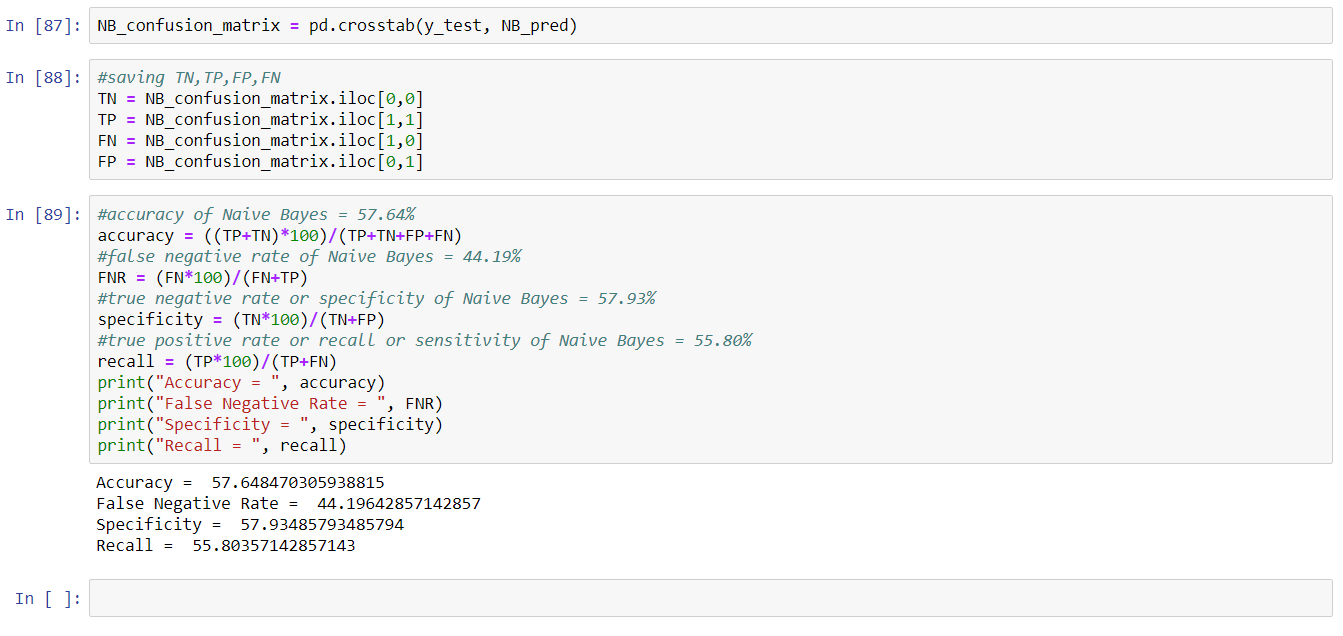
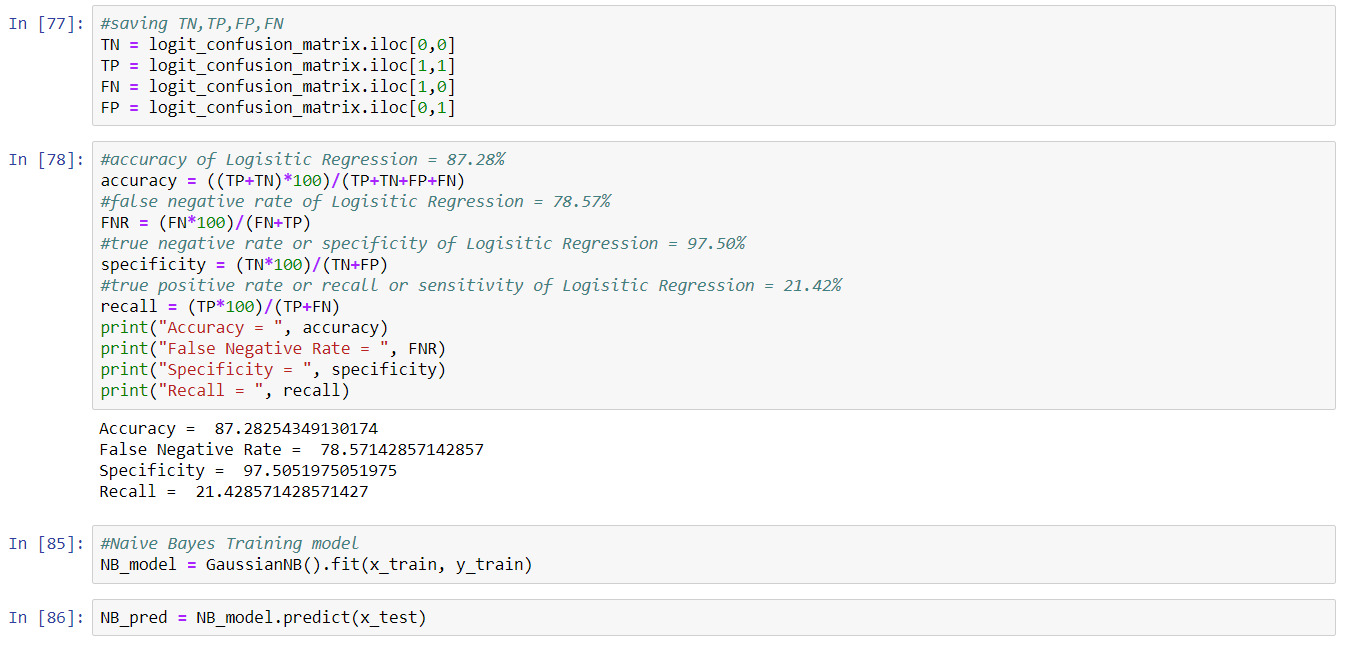
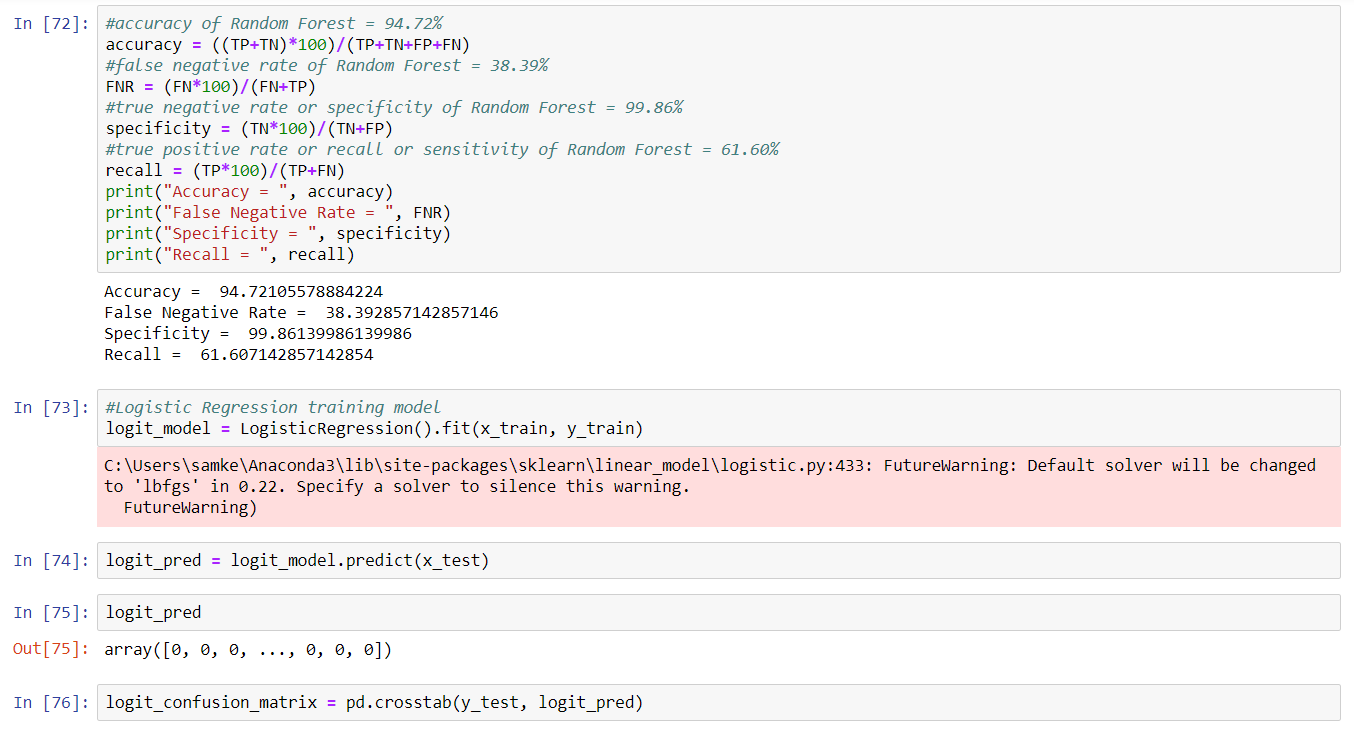
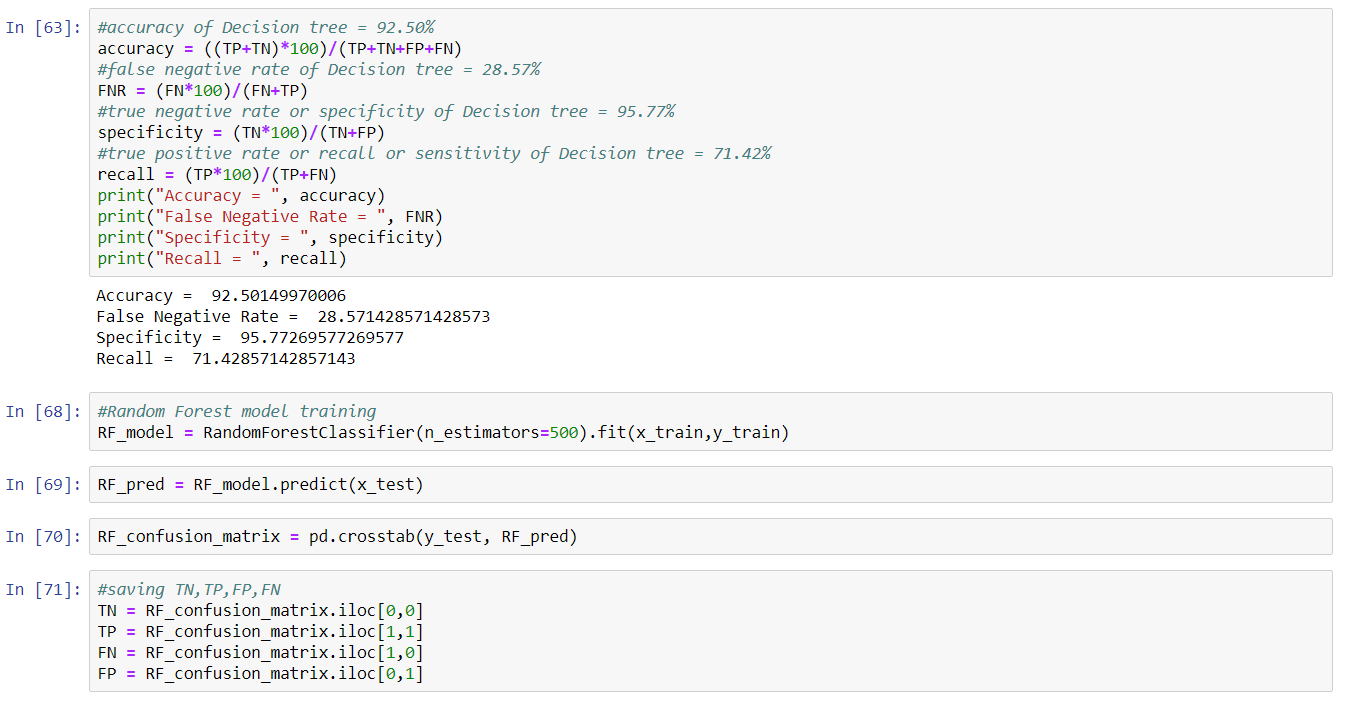












1. **APPENDIX B – R CODE**

rm(list=ls())

#setting working directory

setwd("D:/Edwisor/Project 1 - Churn Reduction/R")

getwd()

#importing libraries

library('corrgram')

library('caret')

library('C50')

library('rpart')

library('randomForest')

library('caTools')

library('e1071')

#loading the data

train\_df = read.csv("Train\_data.csv")

test\_df = read.csv("Test\_data.csv")

#checking summary

summary(train\_df)

#checking count of missing values in each column

sapply(train\_df, function(x) sum(is.na(x)))

sapply(test\_df, function(x) sum(is.na(x)))

#checking the datatype of each column

sapply(train\_df, function(x) class(x))

sapply(test\_df, function(x) class(x))

#converting area code to factor

train\_df$area.code = as.factor(train\_df$area.code)

#column names and data of categorical columns

factor\_ind = sapply(train\_df, is.factor)

factor\_train = train\_df[, factor\_ind]

#chi-square test of independence

for (i in 1:5){

print(names(factor\_train)[i])

print(chisq.test(table(train\_df$Churn, factor\_train[,i])))

}

#p-value of phone number and area code is greater than 0.05, hence dropping them

train\_df = train\_df[-c(3,4)]

test\_df = test\_df[-c(3,4)]

#column names of numerical columns

numeric\_ind = sapply(train\_df, is.numeric)

#correlation plot

corrgram(train\_df[,numeric\_ind], order = F, upper.panel = panel.pie, text.panel = panel.txt, main = "Correlation Plot")

#removing numeric variables with high correlation

train\_df = train\_df[-c(6,9,12,15)]

test\_df = test\_df[-c(6,9,12,15)]

#label encoding categorical variables

for(i in 2:ncol(train\_df)){

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

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

}

}

for(i in 2:ncol(test\_df)){

if(class(test\_df[,i]) == 'factor'){

test\_df[,i] = factor(test\_df[,i], labels = (1:length(levels(factor(test\_df[,i])))))

}

}

#dummy encoding

dmy = dummyVars("~ state", data = train\_df)

train\_df\_state = data.frame(predict(dmy,newdata = train\_df))

dmy = dummyVars("~ state", data = test\_df)

test\_df\_state = data.frame(predict(dmy,newdata = test\_df))

train\_df = cbind(train\_df,train\_df\_state)

test\_df = cbind(test\_df,test\_df\_state)

train\_df = train\_df[-c(1)]

test\_df = test\_df[-c(1)]

y\_train = train\_df$Churn

y\_test = test\_df$Churn

x\_train = train\_df[-c(14)]

x\_test = test\_df[-c(14)]

##DECISION TREE MODEL

DT\_model = rpart(y\_train ~., data = x\_train, method = "class")

DT\_pred = predict(DT\_model, type = "class", newdata = x\_test)

DT\_confusion\_matrix = table(y\_test, DT\_pred)

TN = DT\_confusion\_matrix[1,1]

TP = DT\_confusion\_matrix[2,2]

FN = DT\_confusion\_matrix[2,1]

FP = DT\_confusion\_matrix[1,2]

#accuracy of Decision tree = 94.36%

accuracy = ((TP+TN)\*100)/(TP+TN+FP+FN)

#false negative rate of Decision tree = 35.26%

FNR = (FN\*100)/(FN+TP)

#true negative rate or specificity of Decision tree = 98.96%

specificity = (TN\*100)/(TN+FP)

#true positive rate or recall or sensitivity of Decision tree = 64.73%

recall = (TP\*100)/(TP+FN)

##RANDOM FOREST MODEL

RF\_model = randomForest(y\_train ~., x\_train, importance = TRUE, ntree = 300)

RF\_pred = predict(RF\_model, x\_test)

RF\_confusion\_matrix = table(y\_test, RF\_pred)

TN = RF\_confusion\_matrix[1,1]

TP = RF\_confusion\_matrix[2,2]

FN = RF\_confusion\_matrix[2,1]

FP = RF\_confusion\_matrix[1,2]

#accuracy of Random Forest = 94.96%

accuracy = ((TP+TN)\*100)/(TP+TN+FP+FN)

#false negative rate of Random Forest = 37.05%

FNR = (FN\*100)/(FN+TP)

#true negative rate or specificity of Random Forest = 99.86%

specificity = (TN\*100)/(TN+FP)

#true positive rate or recall or sensitivity of Random Forest = 62.94%

recall = (TP\*100)/(TP+FN)

##LOGISTIC REGRESSION MODEL

logit\_model = glm(y\_train ~., data = x\_train, family = "binomial")

summary(logit\_model)

logit\_pred = predict(logit\_model, newdata = x\_test, type = "response")

#convert probabality

logit\_pred = ifelse(logit\_pred > 0.5, 2, 1)

logit\_confusion\_matrix = table(y\_test, logit\_pred)

TN = logit\_confusion\_matrix[1,1]

TP = logit\_confusion\_matrix[2,2]

FN = logit\_confusion\_matrix[2,1]

FP = logit\_confusion\_matrix[1,2]

#accuracy of Logistic Regression = 87.16%

accuracy = ((TP+TN)\*100)/(TP+TN+FP+FN)

#false negative rate of Logistic Regression = 75%

FNR = (FN\*100)/(FN+TP)

#true negative rate or specificity of Logistic Regression = 96.81%

specificity = (TN\*100)/(TN+FP)

#true positive rate or recall or sensitivity of Logistic Regression = 25%

recall = (TP\*100)/(TP+FN)

##NAIVE BAYES MODEL

NB\_model = naiveBayes(y\_train ~., data = x\_train)

NB\_pred = predict(NB\_model, x\_test, type = "class")

NB\_confusion\_matrix = table(y\_test, NB\_pred)

TN = NB\_confusion\_matrix[1,1]

TP = NB\_confusion\_matrix[2,2]

FN = NB\_confusion\_matrix[2,1]

FP = NB\_confusion\_matrix[1,2]

#accuracy of Naive Bayes = 56.68%

accuracy = ((TP+TN)\*100)/(TP+TN+FP+FN)

#false negative rate of Naive Bayes = 45.98%

FNR = (FN\*100)/(FN+TP)

#true negative rate or specificity of Naive Bayes = 57.10%

specificity = (TN\*100)/(TN+FP)

#true positive rate or recall or sensitivity of Naive Bayes = 54.01%

recall = (TP\*100)/(TP+FN)