**PROJECT II**

**ASHISH KUMAR SAW**

**1st AUGUST 2018**

**CONTENTS**

CHAPTER 1:- PROBLEM STATEMENT

CHAPTER 2:- LOADING AND UNDERSTANDING DATA

CHAPTER 3:- DATA PREPARATION

CHAPTER 4:- MODEL DEVELOPMENT AND EVALUATION

COMPLETE R CODE

COMPLETE PYTHON CODE

**CHAPTER 1**

**PROBLEM STATEMENT:-**

Churn (loss of customers to competition) is a problem for companies because it is moreexpensive to acquire a new customer than to keep your existing one from leaving. Thisproblem statement is targeted at enabling churn reduction using analytics concepts.The objective of this Case is to predict customer behaviour. There is apublic dataset given that has customer usage pattern and if the customer has moved or not.We have to develop an algorithm to predict the churn score based on usagepattern.

**HYPOTHESIS GENERATION:-**

Hypothesis Generation is a very important stage in the Data Science/Machine Learning pipeline. Here, we have to get and understand all the possible factors which are responsible for getting the outcome by Brainstorming. This is done before looking at the data and thoroughly understand the problem statement.

There are some factors which I can think can affect the Churn.

1. Number of voice calls made by the user
2. Charges for the voice calls
3. How the network working in different regions
4. Different plans used by the user like voice plan, internet plan
5. Internet charges monthly
6. Speed of the internet
7. Usage of internet
8. User spend time on voice calls
9. User complaint
10. How old the user is being with the industry

**CHAPTER 2**

**LOADING AND UNDERSTANDING THE DATA:-**

Here, we first load the libraries which are used for loading, manipulating, modelling and evaluating the data.

After, this we load the data and understand the data one bay one.

> #Load the Data

> train=fread("Train\_data.csv",header = T)

> test=fread("Test\_data.csv",header = T)

> #Understanding the data

> dim(train)

[1] 3333 21

> dim(test)

[1] 1667 21

We have two data set given one is train and other is test data. Train data contains 3333 records and 21 features. Whereas, the test data contains 1667 records and 21 features.

> str(train)

Classes ‘data.table’ and 'data.frame': 3333 obs. of 21 variables:

$ state : chr "KS" "OH" "NJ" "OH" ...

$ 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 : chr "382-4657" "371-7191" "358-1921" "375-9999" ...

$ international plan : chr "no" "no" "no" "yes" ...

$ voice mail plan : chr "yes" "yes" "no" "no" ...

$ 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 : chr "False." "False." "False." "False." ...

In the dataset there are 21 features in which 20 are independent variable and 1 is dependent variable. In the dataset 5 are categorical variable and 16 are numerical variable.

Using these independent variables we have to make the model for making predictions for the target variable.

**EXPLORATORY DATA ANALYSIS:-**

After understanding the dimensions and properties of data, we have to deep dive and explore the data visually. It helps us in understanding the nature of data in terms of distribution of the individual variables/features, finding missing values, relationship with other variables and many other things.

**UNIVARIATE ANALYSIS:-**  It is the simplest form of analyzing data where we examine each variable individually. For categorical features we can use frequency table or bar plots which will calculate the number of each category in a particular variable. For numerical features, probability density plots can be used to look at the distribution of the variable.

**TARGET VARIABLE:-** In the given data set the target variable is categorical variable. So, we use frequency table and bar plots for calcualtion of each category.

> prop.table(table(train$Churn))

False. True.

0.8550855 0.1449145

> ggplot(train %>% group\_by(Churn) %>% summarise(Count = n())) +

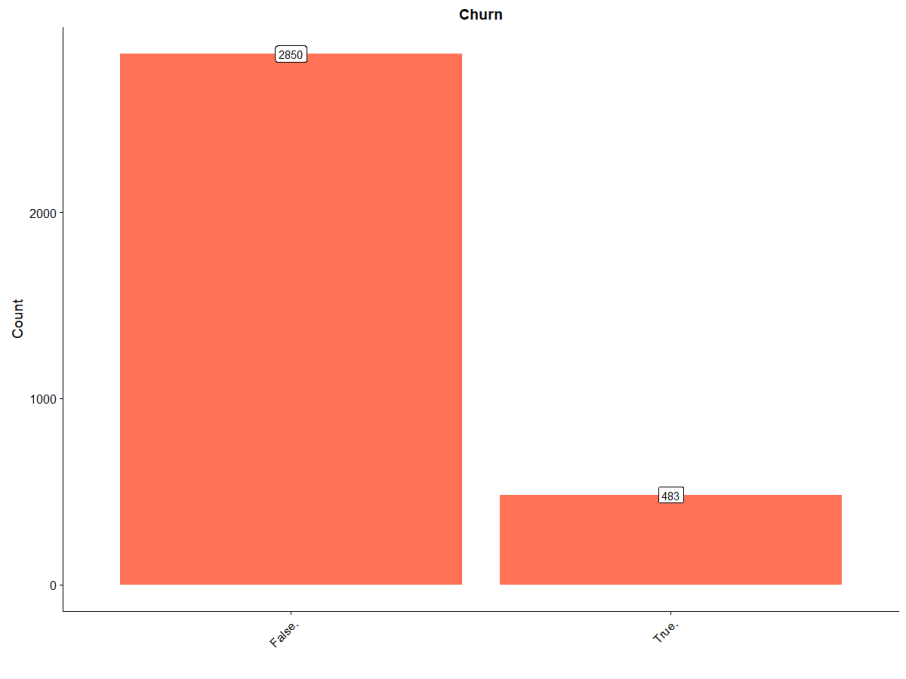
+ geom\_bar(aes(Churn, Count), stat = "identity", fill = "coral1") +

+ xlab("") +

+ geom\_label(aes(Churn, Count, label = Count), vjust = 0.5) +

+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

+ ggtitle("Churn")



Since, the target variable has 85.5% of False data which means that the user is not moving whereas 14.5% of data is True which means that the user is moving.

Since, majority is False value. So, if we make a model in which we make prediction false for all the new data(new coming customer) then we get the accuracy of 85.5% but there is a problem of increase of False Negative Rate. False Negative Rate is the measure cause of the industry downfall. Its, definition is the the actual the value is True but model predicted False. In the industry point of view it is defined as the actual the customer is not Churning but the model predicted the customer is Churning . This results in the customer which are already industry client will be lost which leads to the downfall of the company. So, this is the bad model.

**INDEPENDENT VARIABLE:-** Now, let explore the independent variable one by one.

First we will explore the categorical variable one by one.

> # plot for state

> p1 = ggplot(train %>% group\_by(state) %>% summarise(Count = n())) +

+ geom\_bar(aes(state, Count), stat = "identity", fill = "coral1") +

+ xlab("") +

+ geom\_label(aes(state, Count, label = Count), vjust = 0.5) +

+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

+ ggtitle("State")

> # plot for international plan

> p2 = ggplot(train %>% group\_by(`international plan`) %>% summarise(Count = n())) +

+ geom\_bar(aes(`international plan`, Count), stat = "identity", fill = "coral1") +

+ geom\_label(aes(`international plan`, Count, label = Count), vjust = 0.5) +

+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

+ ggtitle("International Plan")

> # plot for voice mail plan

> p3 = ggplot(train %>% group\_by(`voice mail plan`) %>% summarise(Count = n())) +

+ geom\_bar(aes(`voice mail plan`, Count), stat = "identity", fill = "coral1") +

+ xlab("") +

+ geom\_label(aes(`voice mail plan`, Count, label = Count), vjust = 0.5) +

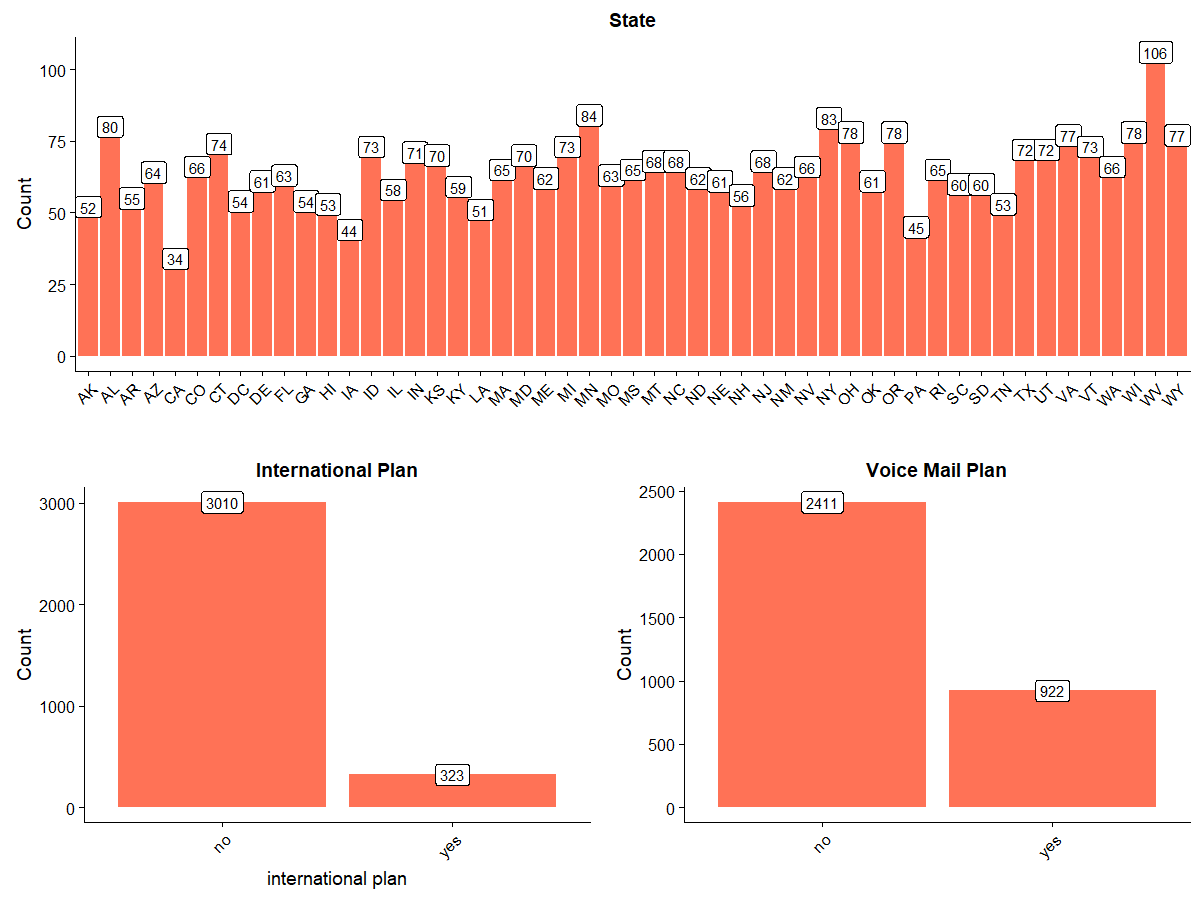
+ theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

+ ggtitle("Voice Mail Plan")

> second\_row = plot\_grid(p2, p3, nrow = 1)

> plot\_grid(p1, second\_row, ncol = 1)

The below figure shows the the distribution of the categorical variable. Most of the users have no international plan as well as voice mail plan. The state diagram shows the number of user in different state



Now, we explore the numerical independent variable one by one.

> p4 = ggplot(train) + geom\_histogram(aes(`account length`), binwidth = 10, fill = "blue")

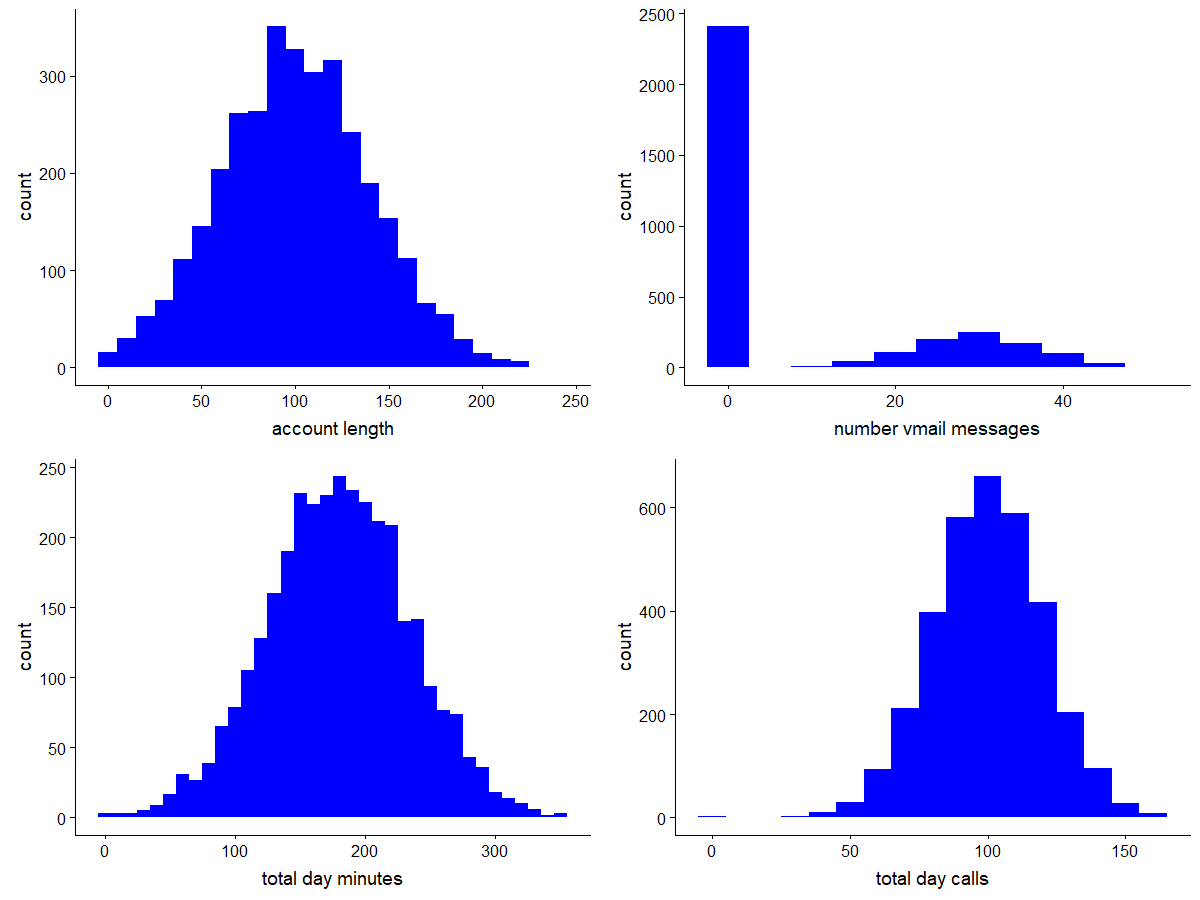
> p5 = ggplot(train) + geom\_histogram(aes(`number vmail messages`), binwidth = 5, fill = "blue")

> p6 = ggplot(train) + geom\_histogram(aes(`total day minutes`), binwidth = 10, fill = "blue")

> p7 = ggplot(train) + geom\_histogram(aes(`total day calls`), binwidth = 10, fill = "blue")

> plot\_grid(p4, p5, p6,p7, nrow = 2) # plot\_grid() from cowplot package

The below figure shows the distribution of the independents variables. Here, the features account length, total day minutes and total day calls are uniformlly distributed but the features number vmail message is Right skewed. This shows that most of the user have zero vmail messages and less have some vmail messages.



> p8 = ggplot(train) + geom\_histogram(aes(`total day charge`), binwidth = 5, fill = "blue")

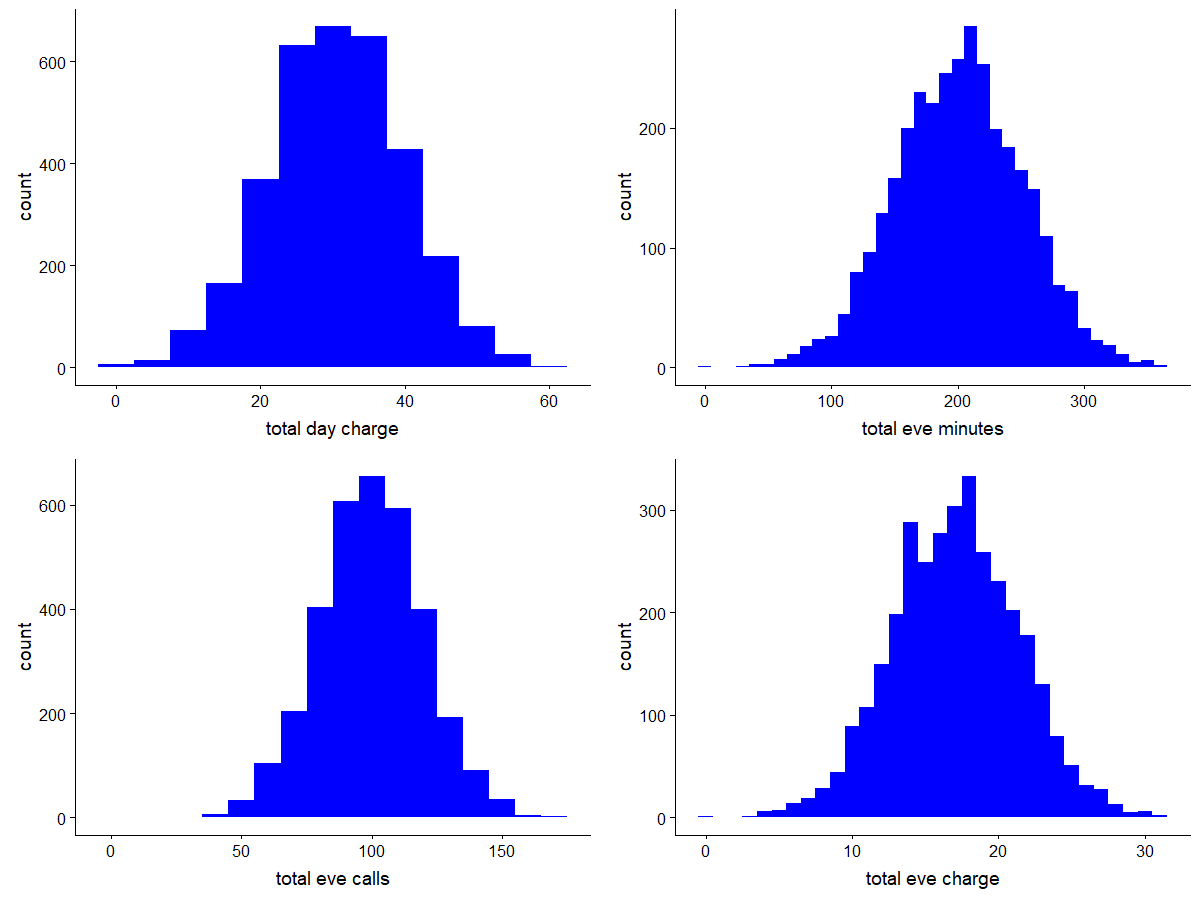
> p9 = ggplot(train) + geom\_histogram(aes(`total eve minutes`), binwidth = 10, fill = "blue")

> p10 = ggplot(train) + geom\_histogram(aes(`total eve calls`), binwidth = 10, fill = "blue")

> p11 = ggplot(train) + geom\_histogram(aes(`total eve charge`), binwidth = 1, fill = "blue")

> plot\_grid(p8, p9, p10,p11, nrow = 2) # plot\_grid() from cowplot package

Here, the Histogram shows that the features total day charge, total eve minutes, total eve calls and total eve charge are uniformlly distributed.



> p12 = ggplot(train) + geom\_histogram(aes(`number customer service calls`), binwidth = 0.5, fill = "blue")

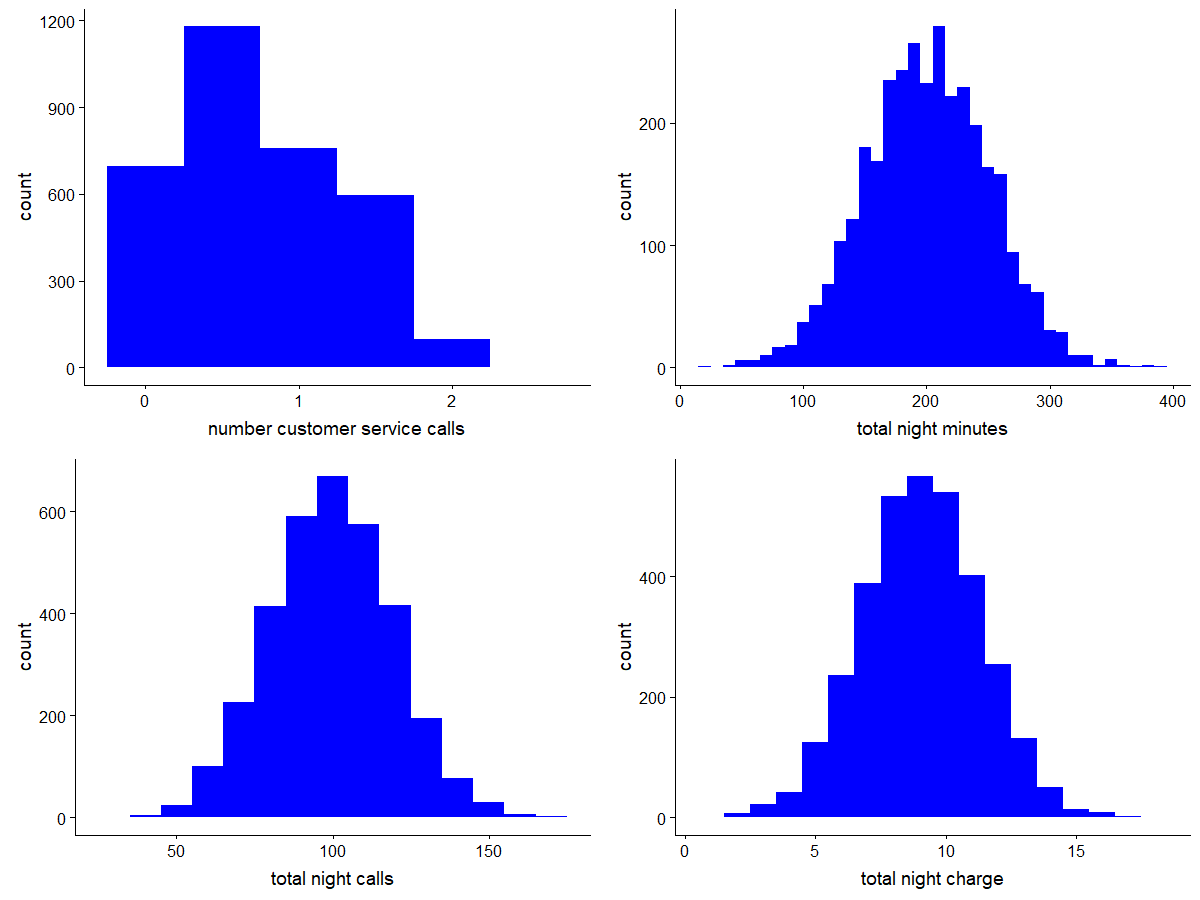
> p13 = ggplot(train) + geom\_histogram(aes(`total night minutes`), binwidth = 10, fill = "blue")

> p14 = ggplot(train) + geom\_histogram(aes(`total night calls`), binwidth = 10, fill = "blue")

> p15 = ggplot(train) + geom\_histogram(aes(`total night charge`), binwidth = 1, fill = "blue")

> plot\_grid(p12, p13, p14,p15, nrow = 2) # plot\_grid() from cowplot package

Here, the grapg shows that the feature number customer service calls is almost uniformly distributed and the features total night minutes, total night calls and total night charges are uniformly distributed. There are also outliers present in the features.



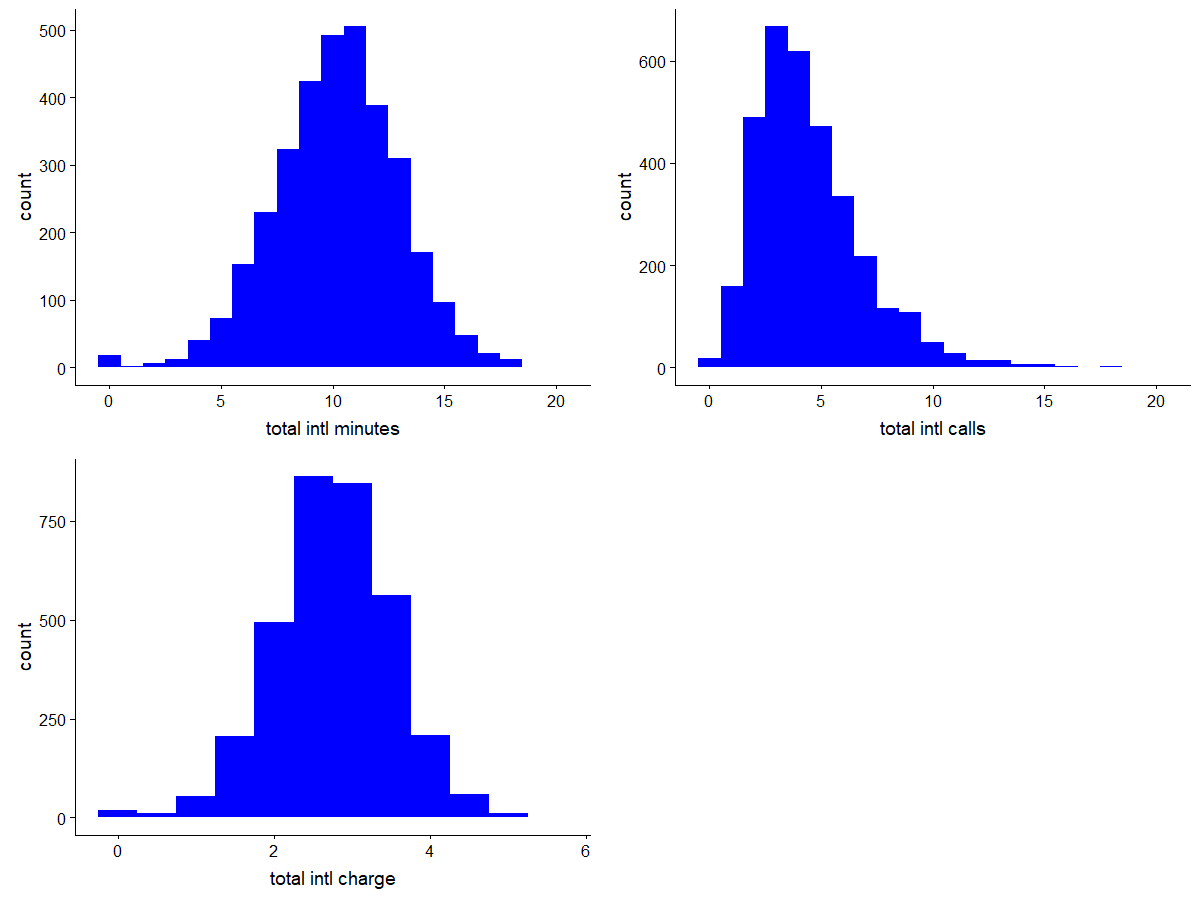
> p16 = ggplot(train) + geom\_histogram(aes(`total intl minutes`), binwidth = 1, fill = "blue")

> p17 = ggplot(train) + geom\_histogram(aes(`total intl calls`), binwidth = 1, fill = "blue")

> p18 = ggplot(train) + geom\_histogram(aes(`total intl charge`), binwidth = 0.5, fill = "blue")

> plot\_grid(p16, p17, p18, nrow = 2) # plot\_grid() from cowplot package

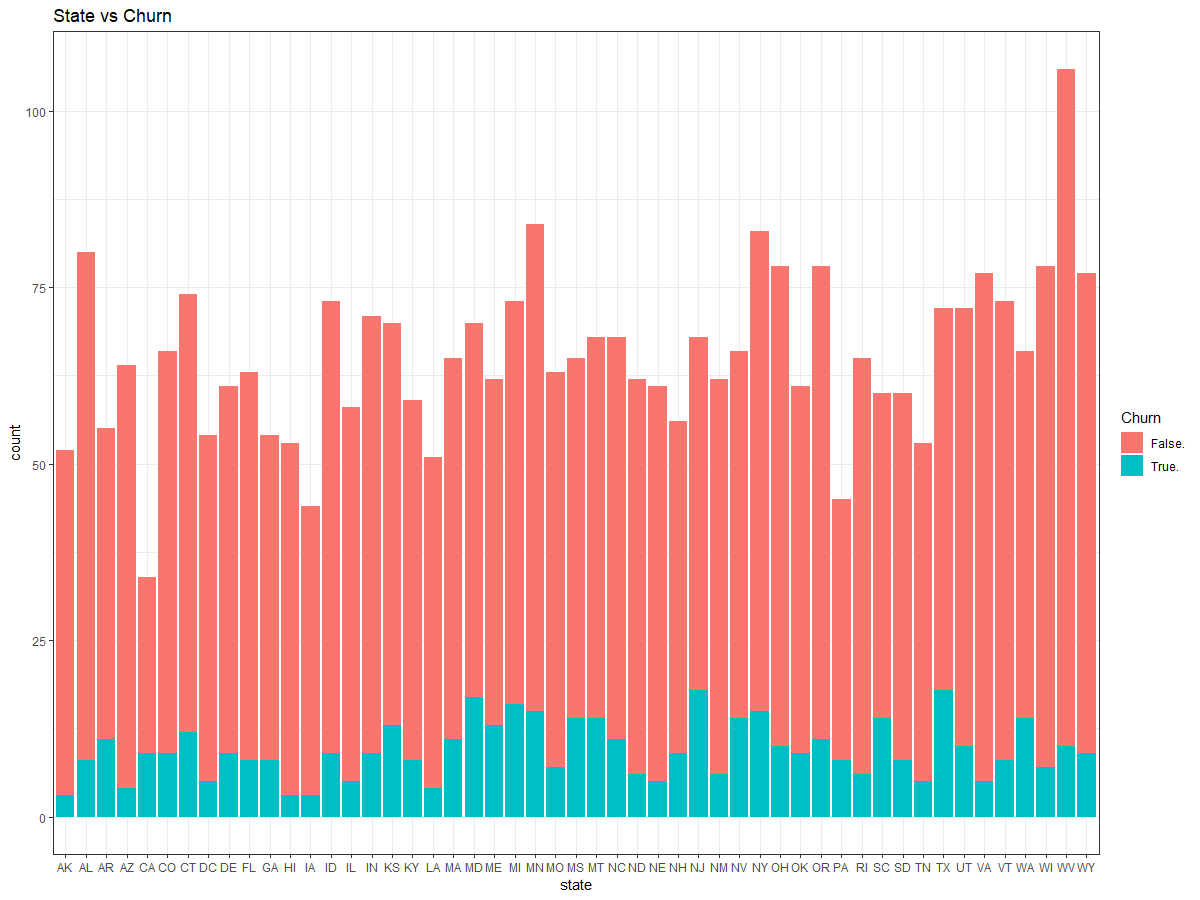
Here, the features total intl minutes, total intl calls and total intl charge are uniformly distributed. There are also outliers resent in the features.



**MULTIVARIATE ANALYSIS:-** Multivariate Analysis finds out the relationship between two or more features.

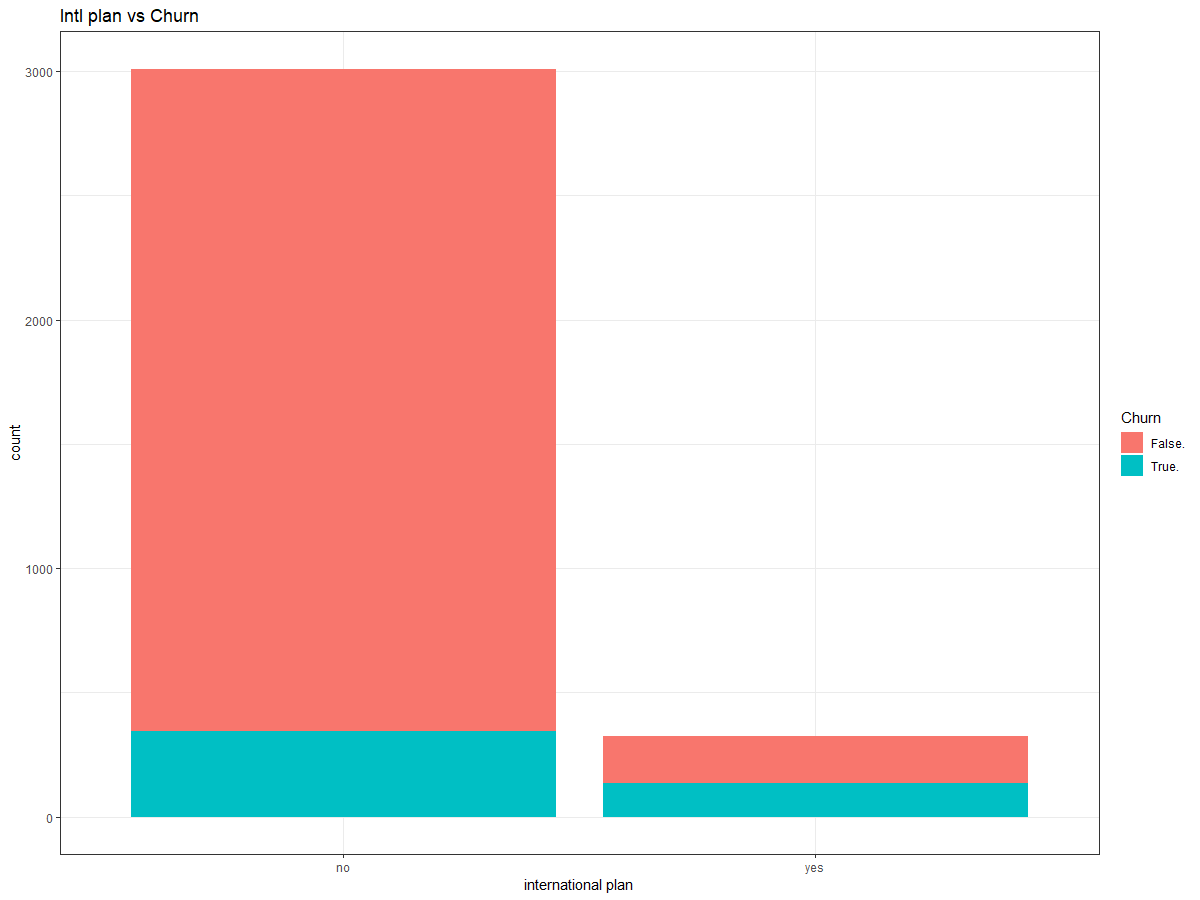
> #plot a stacked barcharts

> ggplot(train,aes(state,fill=Churn))+geom\_bar()+labs(title="State vs Churn",x="state",y="count")+theme\_bw()



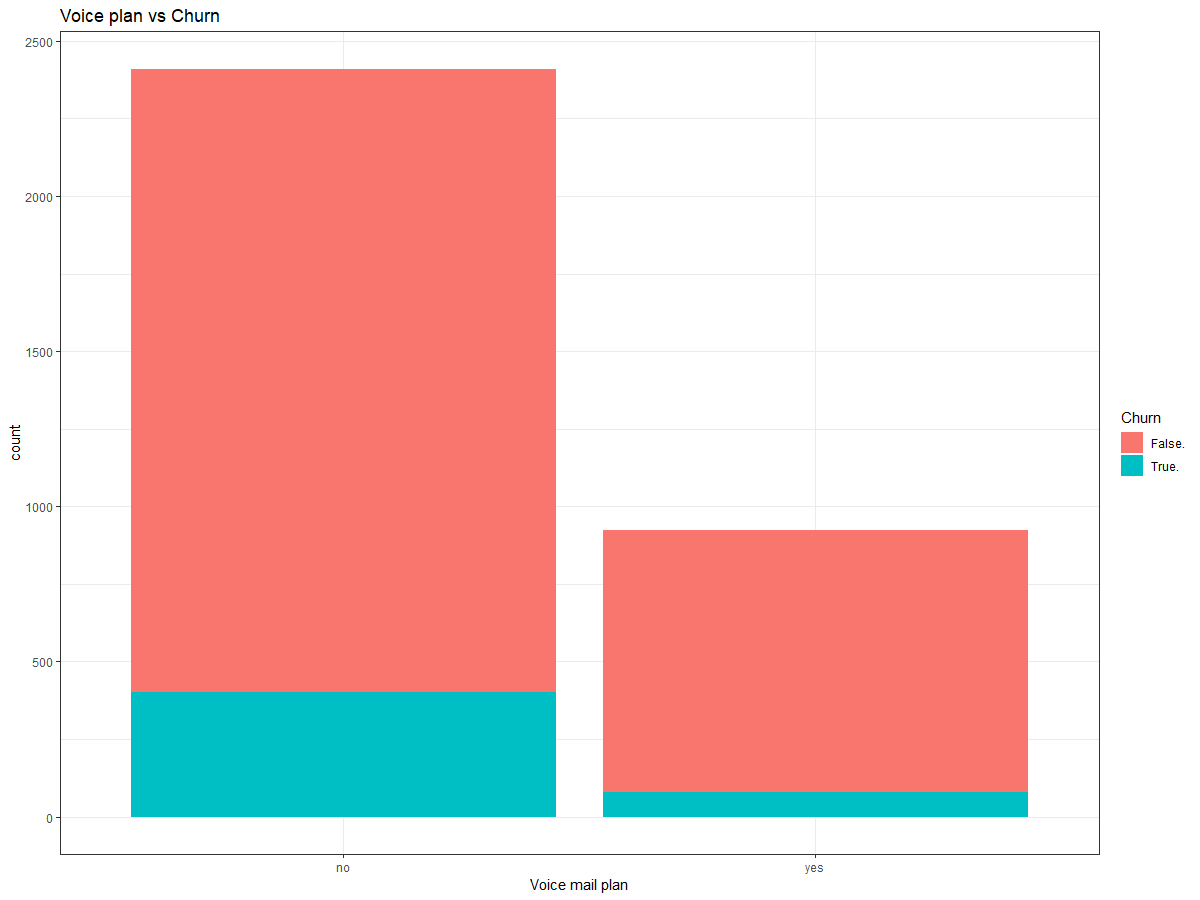
> ggplot(train,aes(`international plan`,fill=Churn))+geom\_bar()+labs(title="Intl plan vs Churn",

+ x="international plan",y="count")+theme\_bw()



> ggplot(train,aes(`voice mail plan`,fill=Churn))+geom\_bar()+labs(title="Voice plan vs Churn",

+ x="Voice mail plan",y="count")+theme\_bw()



The, above diagram shows the distribution of the target variable with respect to the categorical variable.

**CHAPTER 3**

**DATA PREPARATION**

**MISSING VALUE:-** Missing data can have a severe impact on building predictive models because the missing values might be contain some vital information which could help in making better predictions. So, it becomes imperative to carry out missing data imputation. There are different methods to treat missing values based on the problem and the data. Some of the common techniques are as follows:

1. Deletion of Rows
2. Mean/Median/Mode Imputation
3. Building Predictive model

> sum(is.na(train))

[1] 0

> sum(is.na(test))

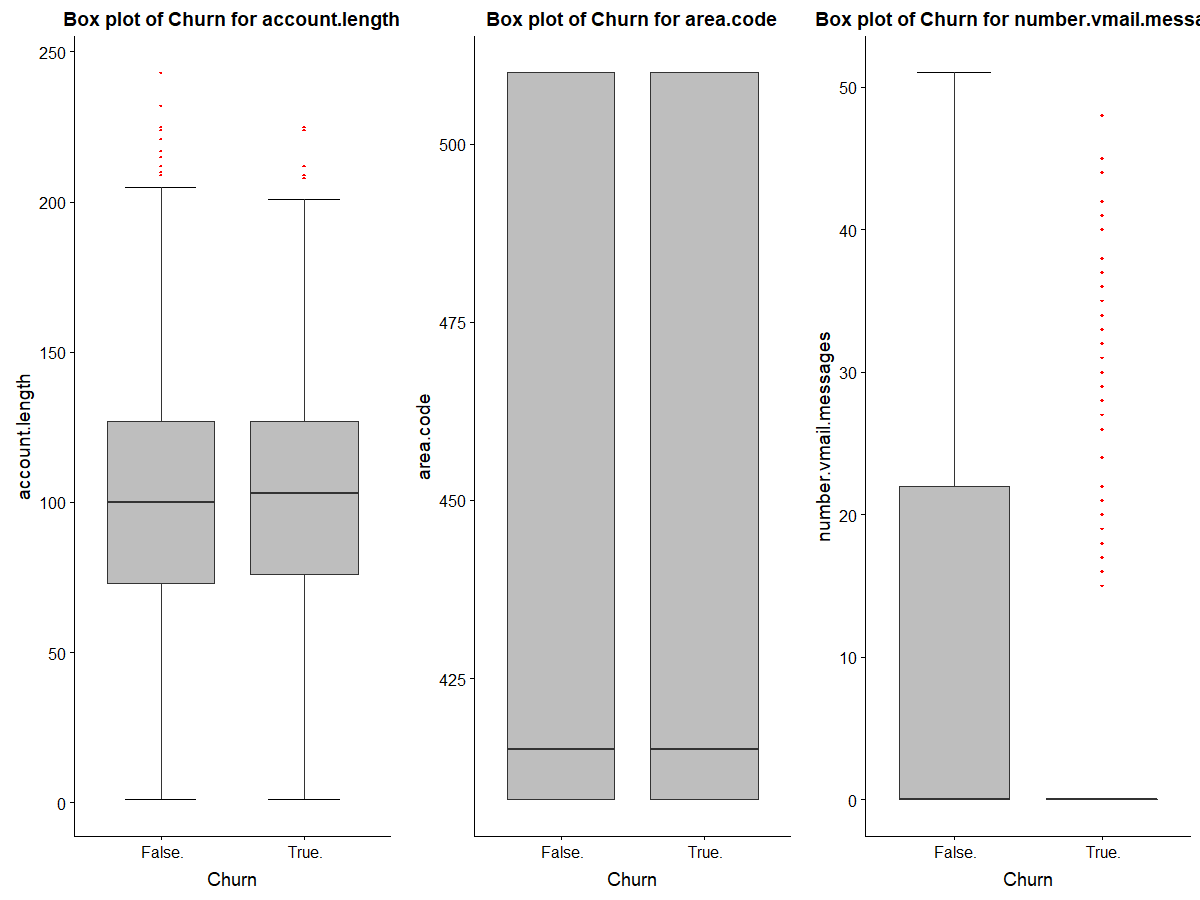
[1] 0

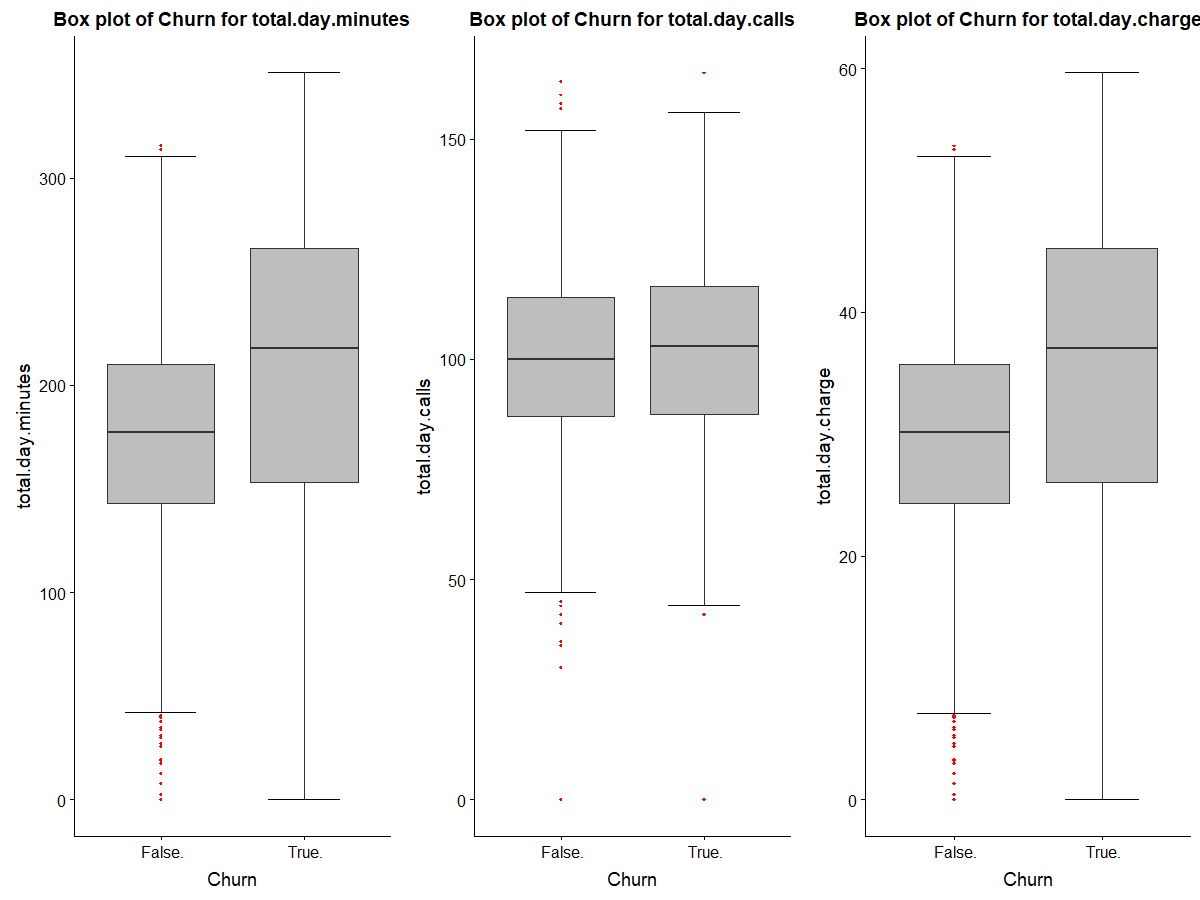
There are no missing value present in the train and test data set.

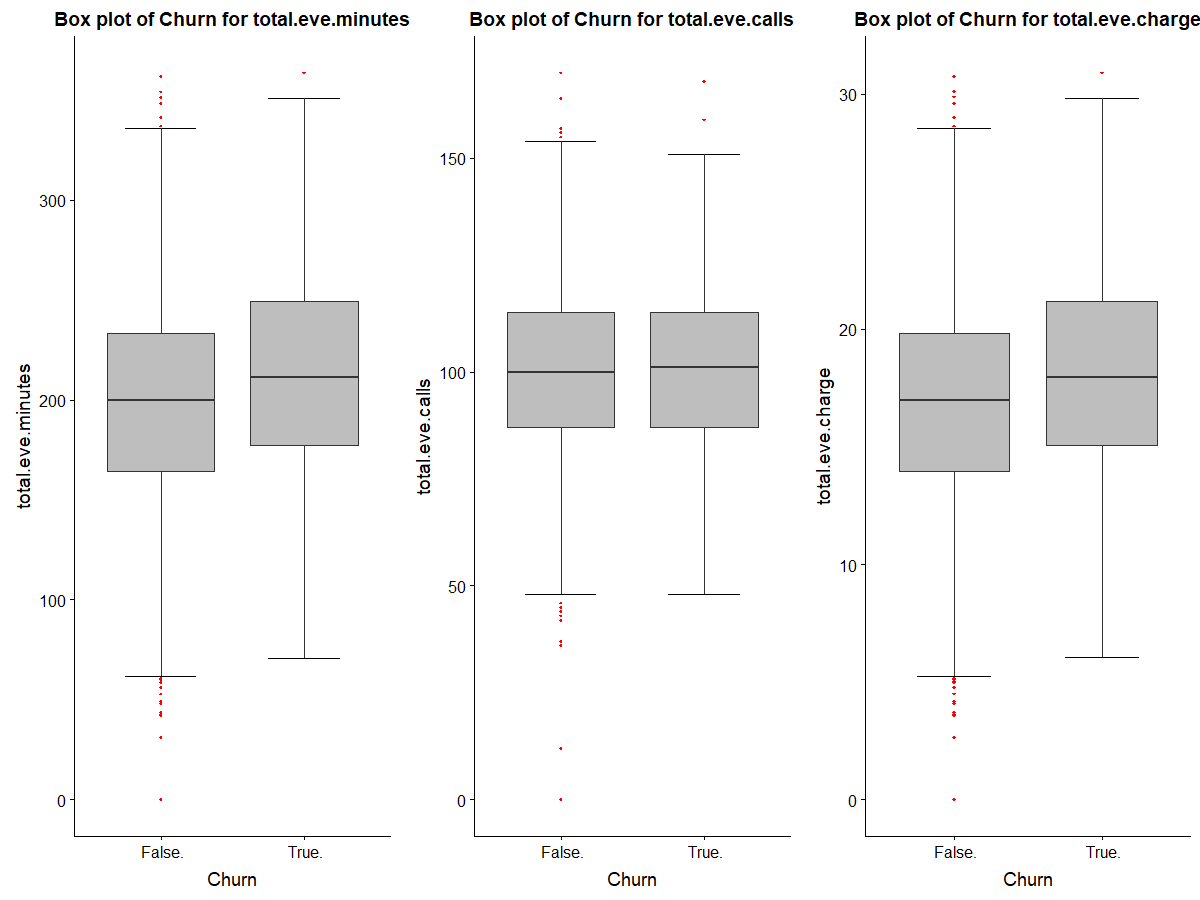
**OUTLIER ANALYSIS:-** Outliers are the observations inconsistent with rest of the data.

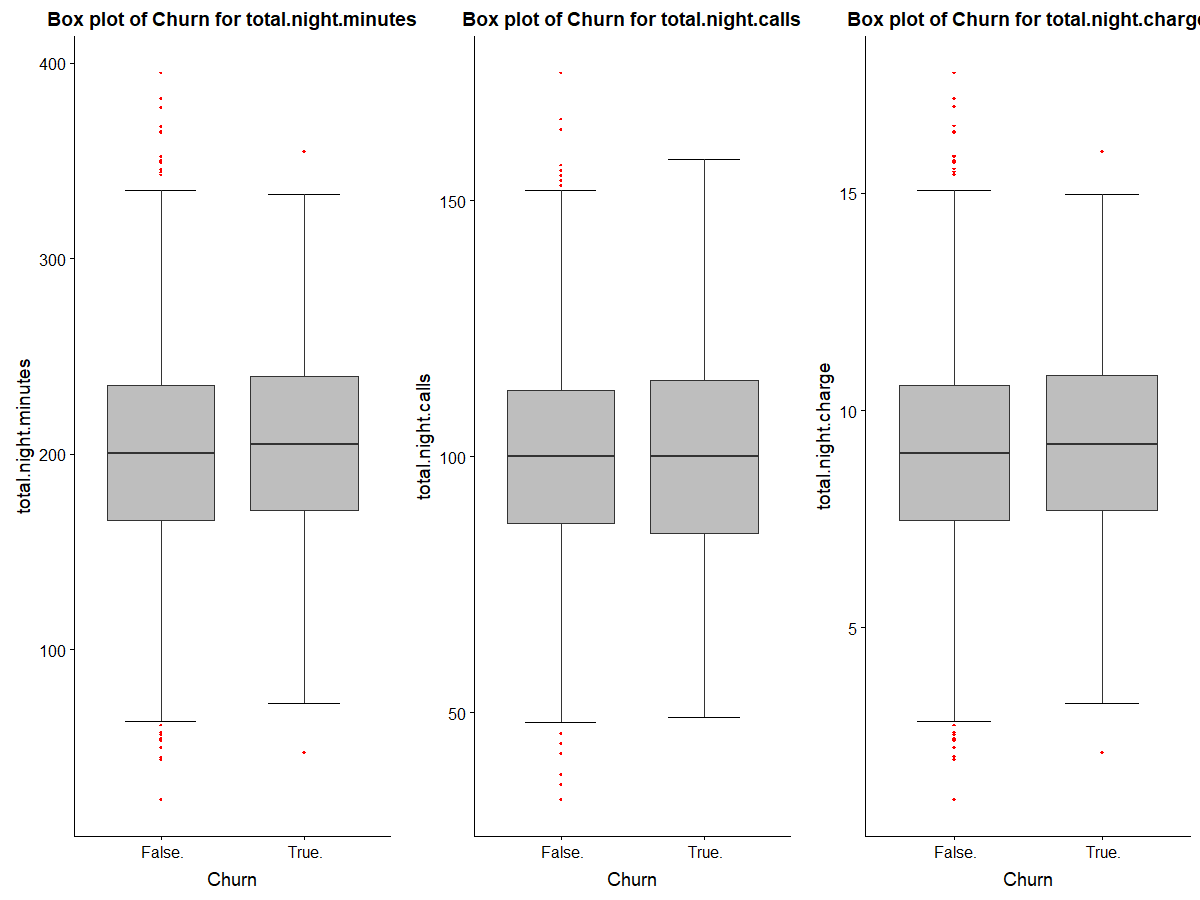
Here, in the outliers due to variation of the user for usage of the technology. Some of the has more vmail messages some have less. Some user talk for more minutes whereas some talk less.

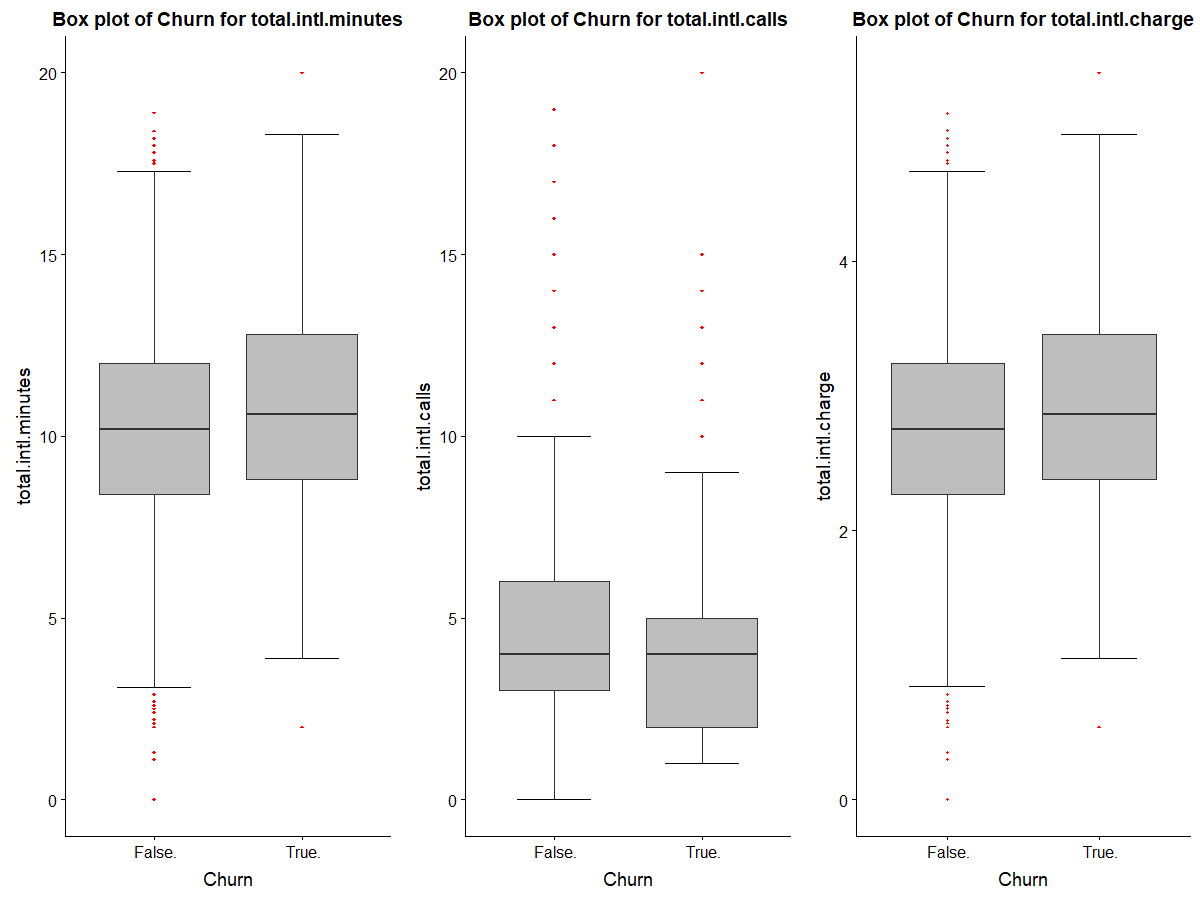
Since there are lots of outliers present in the data, so if we delete the outliers values then very less amount of data is present so this can’t be done. If we compute the outliers then this leads to the bias in the dataset which will not model the dataset correctly. So, better to leave this outliers.

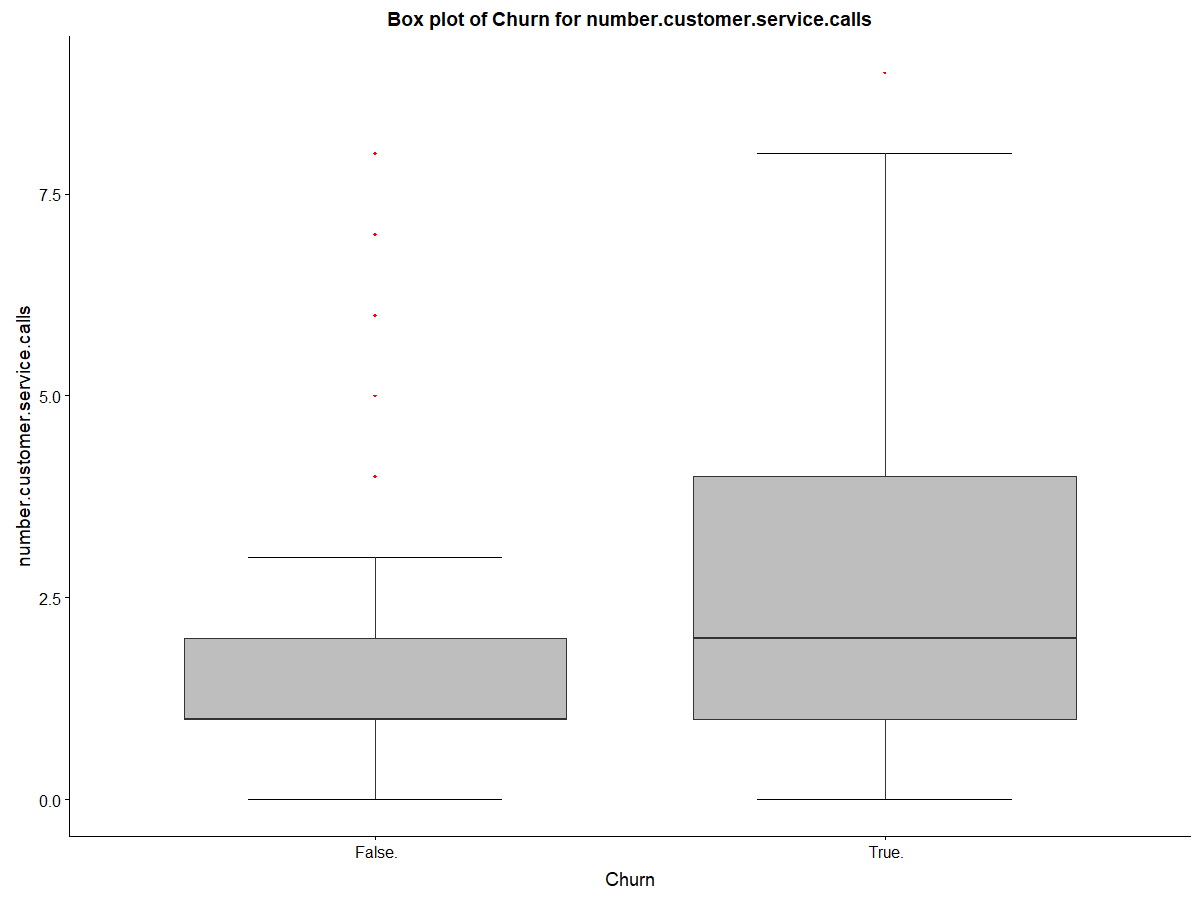












**FEATURE ENGINEERING:-**  Feature Engineering is the process in which we make new features on the basis of the hypothesis. There are already some features are given which we have make the hypothesis. Number, carges time for the voice call is given. Some of the hypothesis can’t be generated as the the given features can’t able to generate new features. Like, speed of internet, charges on internet can’t be generated. So, we have to make the model on the basis of given data only.

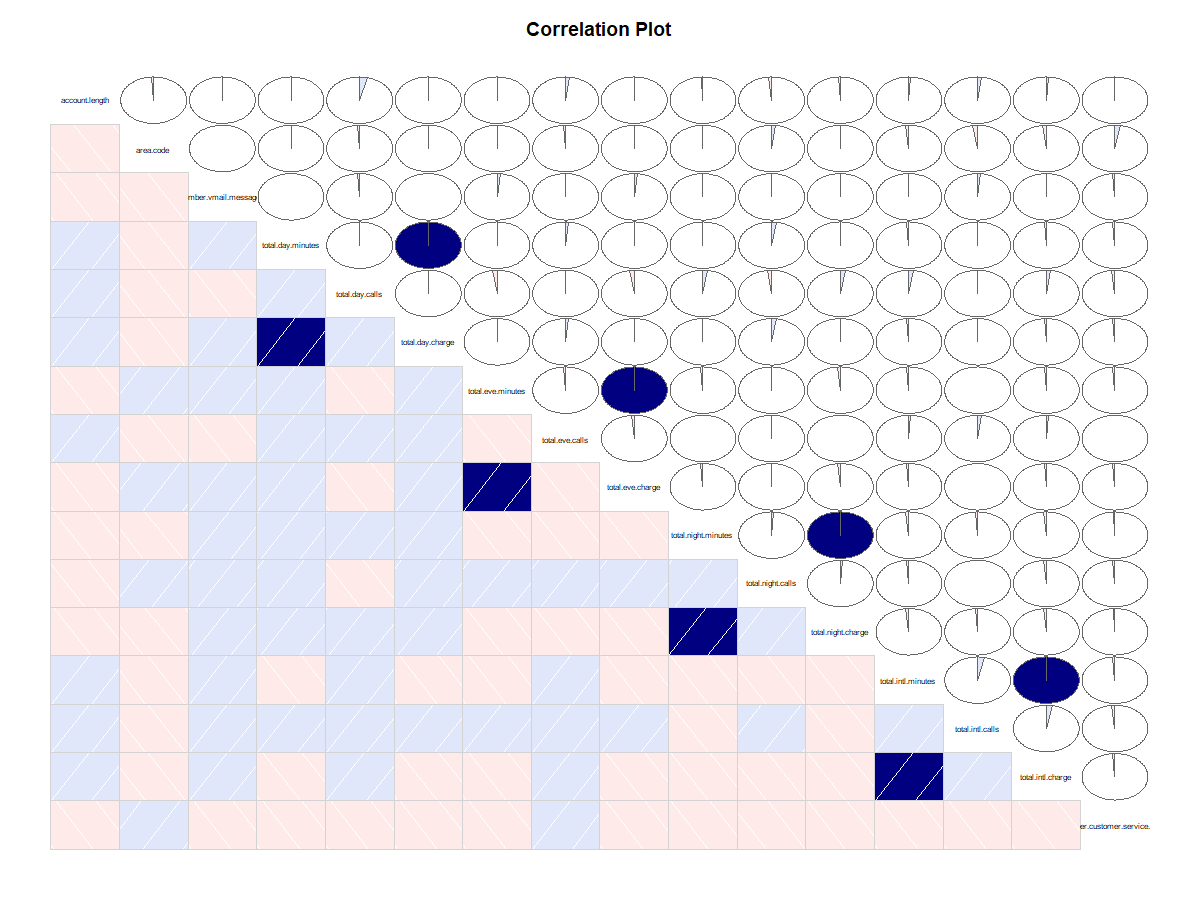
**CORRELATION:-** It is defined as how the different features are correlaed with each other.

For, defining correlation between the continous variable we use corgram plot and for the categorical variable we use the chi-square test.

> ## Correlation Plot

> corrgram(train[,numeric\_index], order = F,

+ upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")



The above diagram shows that the features total day minutes and total day charge are highly related with each other so here we delete total day charge. Similarly the features total eve minutes, total night minutes and total intl minutes are highly related with total eve charge, total night charge and total intl charge respectively. So, the features total eve charge, total night charge and total intl charge are deleted.

For, the continous variable:-

> ## Chi-squared Test of Independence

> factor\_index = sapply(train,is.factor)

> factor\_data = train[,factor\_index]

> for (i in 1:5)

+ {

+ print(names(factor\_data)[i])

+ print(chisq.test(table(factor\_data[,i])))

+ }

[1] "state"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 106.55, df = 50, p-value = 5.695e-06

[1] "phone.number"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0, df = 3332, p-value = 1

[1] "international.plan"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 2166.2, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 665.2, df = 1, p-value < 2.2e-16

[1] "Churn"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 1681, df = 1, p-value < 2.2e-16

In Chi-square test the phone number has p-value greater than 0.05 so, phone number is deleted whereas other features have p-value less than 0.05.

Now, we reduce the dimension of the dataset. The unwanted features are deleted from the dataset

> ## Dimension Reduction

> train = subset(train,

+ select = -c(area.code,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge,

+ phone.number))

> test = subset(test,

+ select = -c(area.code,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge,

+ phone.number))

**ENCODING CATEGORICAL VARIABLES:-** Most of the machine learning algorithms produce better result with numerical variables only. So, it is essential to treat the categorical variables present in the data.

> ##Data Manupulation; convert string categories into factor numeric

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

+

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

+

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

+

+ }

+ }

> for(i in 1:ncol(test)){

+

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

+

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

+

+ }

+ }

**CHAPTER 4**

**MODEL DEVELOPMENT AND EVALUATION:-** There are various machine learning model we used which can predict the test case for best accuracy.

But, before this we will divide the train data into the two parts one is the train part and other is the validation part. Train part is to model the data and validation part is to make prediction and check the accuracy.

> set.seed(1234)

> train.index = createDataPartition(train$Churn, p = .80, list = FALSE)

> train = train[ train.index,]

> validation = train[-train.index,]

Train part contains 80% of data whereas validation part contains 20% of data.

Modeling the data:-

1. Decision Tree :- A decision tree is a predictive model based on a branching series of Boolean tests. It works on two algorithm, either on the forward propagation or the back propagation.
2. > #Lets predict for validation cases
3. > C50\_Predictions = predict(C50\_model, validation[,-15], type = "class")
4. > ##Evaluate the performance of classification model
5. > ConfMatrix\_C50 = table(validation$Churn, C50\_Predictions)
6. > confusionMatrix(ConfMatrix\_C50)
7. Confusion Matrix and Statistics
8. C50\_Predictions
9. 1 2
10. 1 469 0
11. 2 6 75
13. Accuracy : 0.9891
14. 95% CI : (0.9764, 0.996)
15. No Information Rate : 0.8636
16. P-Value [Acc > NIR] : < 2e-16
18. Kappa : 0.9552
19. Mcnemar's Test P-Value : 0.04123
21. Sensitivity : 0.9874
22. Specificity : 1.0000
23. Pos Pred Value : 1.0000
24. Neg Pred Value : 0.9259
25. Prevalence : 0.8636
26. Detection Rate : 0.8527
27. Detection Prevalence : 0.8527
28. Balanced Accuracy : 0.9937
30. 'Positive' Class : 1

Accuracy on the validation data is 98.9% and False Negative Rate is 7.7%.

Now, to check the accuracy on the test data.

> #Lets Predict for test case

> test\_D = predict(C50\_model, test[,-15], type = "class")

> ##Evaluate the performance of classification model

> ConfMatrix\_test\_D = table(test$Churn, test\_D)

> confusionMatrix(ConfMatrix\_test\_D)

Confusion Matrix and Statistics

test\_D

1 2

1 1433 10

2 66 158

Accuracy : 0.9544

95% CI : (0.9433, 0.9639)

No Information Rate : 0.8992

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7809

Mcnemar's Test P-Value : 2.81e-10

Sensitivity : 0.9560

Specificity : 0.9405

Pos Pred Value : 0.9931

Neg Pred Value : 0.7054

Prevalence : 0.8992

Detection Rate : 0.8596

Detection Prevalence : 0.8656

Balanced Accuracy : 0.9482

'Positive' Class : 1

Here, the accuracy is 95.4% and False Negative rate is 29.4%.

2. Random Forest:- Random forest is an ensemble that consists of many decision trees.

> RF\_model = randomForest(Churn ~ .,train, importance = TRUE, ntree = 500)

> #Extract rules fromn random forest

> #transform rf object to an inTrees' format

> treeList = RF2List(RF\_model)

> #

> # #Extract rules

> exec = extractRules(treeList, train[,-15]) # R-executable conditions

3959 rules (length<=6) were extracted from the first 100 trees.

There were 50 or more warnings (use warnings() to see the first 50)

> #

> # #Visualize some rules

> exec[1:2,]

[1] "X[,3] %in% c('1') & X[,4] %in% c('1') & X[,6]<=263.55 & X[,6]<=236.85 & X[,11]<=117.5 & X[,12]<=4.8"

[2] "X[,3] %in% c('1') & X[,4] %in% c('1') & X[,6]<=263.55 & X[,6]<=236.85 & X[,11]<=117.5 & X[,12]>4.8"

> #

> # #Make rules more readable:

> readableRules = presentRules(exec, colnames(train))

> readableRules[1:2,]

[1] "international.plan %in% c('1') & voice.mail.plan %in% c('1') & total.day.minutes<=263.55 & total.day.minutes<=236.85 & total.night.calls<=117.5 & total.intl.minutes<=4.8"

[2] "international.plan %in% c('1') & voice.mail.plan %in% c('1') & total.day.minutes<=263.55 & total.day.minutes<=236.85 & total.night.calls<=117.5 & total.intl.minutes>4.8"

> #

> # #Get rule metrics

> ruleMetric = getRuleMetric(exec, train[,-15], train$Churn) # get rule metrics

[1] "861 paths are ignored."

> #

> # #evaulate few rules

> ruleMetric[1:2,]

len freq err

[1,] "6" "0.014" "0.108"

[2,] "6" "0.442" "0.07"

condition pred

[1,] "X[,3] %in% c('1') & X[,4] %in% c('1') & X[,6]<=263.55 & X[,6]<=236.85 & X[,11]<=117.5 & X[,12]<=4.8" "1"

[2,] "X[,3] %in% c('1') & X[,4] %in% c('1') & X[,6]<=263.55 & X[,6]<=236.85 & X[,11]<=117.5 & X[,12]>4.8" "1"

> #Presdict test data using random forest model

> RF\_Predictions = predict(RF\_model, validation[,-15])

> ##Evaluate the performance of classification model

> ConfMatrix\_RF = table(validation$Churn, RF\_Predictions)

> confusionMatrix(ConfMatrix\_RF)

Confusion Matrix and Statistics

RF\_Predictions

1 2

1 469 0

2 0 81

Accuracy : 1

95% CI : (0.9933, 1)

No Information Rate : 0.8527

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0000

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 1.0000

Prevalence : 0.8527

Detection Rate : 0.8527

Detection Prevalence : 0.8527

Balanced Accuracy : 1.0000

'Positive' Class : 1

Here, the accuracy is 100% on validation data and False Negative Rate is 0%

Now, predict for test case and check its accuracy.

> #Random forest for test case

> RF\_Test=predict(RF\_model,test[,-15])

> ##Evaluate the performance of classification model

> ConfMatrix\_RF\_Test = table(test$Churn, RF\_Test)

> confusionMatrix(ConfMatrix\_RF\_Test)

Confusion Matrix and Statistics

RF\_Test

1 2

1 1419 24

2 68 156

Accuracy : 0.9448

95% CI : (0.9327, 0.9553)

No Information Rate : 0.892

P-Value [Acc > NIR] : 2.524e-14

Kappa : 0.7413

Mcnemar's Test P-Value : 7.358e-06

Sensitivity : 0.9543

Specificity : 0.8667

Pos Pred Value : 0.9834

Neg Pred Value : 0.6964

Prevalence : 0.8920

Detection Rate : 0.8512

Detection Prevalence : 0.8656

Balanced Accuracy : 0.9105

'Positive' Class : 1

Here, the accuracy for test data is 94.5 and False Negative Rate is 30.4%

3 . Logistic Regression:- It is used for classification model and it uses probability theorem for prediction

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2209.0 on 2666 degrees of freedom

Residual deviance: 1599.9 on 2603 degrees of freedom

AIC: 1727.9

Number of Fisher Scoring iterations: 6

> #Logistic Regression

> logit\_model = glm(Churn ~ ., data = train, family = "binomial")

> #summary of the model

> summary(logit\_model)

> #predict using logistic regression

> logit\_Predictions = predict(logit\_model, newdata = validation, type = "response")

> #convert prob

> logit\_Predictions = ifelse(logit\_Predictions > 0.5, 1, 0)

> ##Evaluate the performance of classification model

> ConfMatrix\_RF = table(validation$Churn, logit\_Predictions)

> #Accuracy

> sum(diag(ConfMatrix\_RF))/nrow(validation)

Call:

glm(formula = Churn ~ ., family = "binomial", data = train)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.0006 -0.4840 -0.2933 -0.1492 2.9769

[1] 0.8654545

Here, the accuracy for the validation part is 86.5 and False Negative Rate is 72.8%.

Now, for the test case:-

> #prediction for test cases

> logit\_Predictions\_Test = predict(logit\_model, newdata = test, type = "response")

> #convert prob

> logit\_Predictions\_Test = ifelse(logit\_Predictions\_Test > 0.5, 1, 0)

> ##Evaluate the performance of classification model

> ConfMatrix\_RF\_T = table(test$Churn, logit\_Predictions\_Test)

> #Accuracy

> sum(diag(ConfMatrix\_RF\_T))/nrow(test)

[1] 0.8674265

Here, the accuracy for the test case is 86.7% and False Negative Rate is 75.4%.

1. KNN-Imputation:- KNN is simple algorithm that stores all available cases and classifies new cases based on a similarity measure.
2. > ##KNN Implementation
3. > library(class)
4. > #Predict test data
5. > KNN\_Predictions = knn(train[, 1:14], validation[, 1:14], train$Churn, k = 7)
6. > #Confusion matrix
7. > Conf\_matrix = table(KNN\_Predictions, validation$Churn)
8. > #Accuracy
9. > sum(diag(Conf\_matrix))/nrow(validation)
10. [1] 0.9036364

Here, the accuracy for the validation data is 90.4% and False Negative Rate is 6.25%

Now, for the test case:-

> #Prediction for test case

> KNN\_Predictions\_T = knn(train[, 1:14], test[, 1:14], train$Churn, k = 7)

> Conf\_matrix\_T = table(KNN\_Predictions\_T,test$Churn)

> #Accuracy

> sum(diag(Conf\_matrix\_T))/nrow(test)

[1] 0.8920216

Here, the accuracy for the test case is 89.2% and False Negative Rate is 18.6%.

5. Naïve Bayes:- It works on Bayes theorem of probability to predict the class of unknown data set. Bayes' theorem with strong (naive) independence assumptions between the features.

> #naive Bayes

> library(e1071)

> #Develop model

> NB\_model = naiveBayes(Churn ~ ., data = train)

> #predict on test cases #raw

> NB\_Predictions = predict(NB\_model, validation[,1:14], type = 'class')

> #Look at confusion matrix

> Conf\_matrix\_V = table(observed = validation[,15], predicted = NB\_Predictions)

> confusionMatrix(Conf\_matrix\_V)

Confusion Matrix and Statistics

predicted

observed 1 2

1 457 12

2 48 33

Accuracy : 0.8909

95% CI : (0.8618, 0.9157)

No Information Rate : 0.9182

P-Value [Acc > NIR] : 0.9899

Kappa : 0.4678

Mcnemar's Test P-Value : 6.228e-06

Sensitivity : 0.9050

Specificity : 0.7333

Pos Pred Value : 0.9744

Neg Pred Value : 0.4074

Prevalence : 0.9182

Detection Rate : 0.8309

Detection Prevalence : 0.8527

Balanced Accuracy : 0.8191

'Positive' Class : 1

Here, the accuracy for the validation data is 89.1 and False Negative Rate is 59.3%

Now, for the test case:-

> #Prediction for test case

> NB\_Predictions\_T = predict(NB\_model, test[,1:14], type = 'class')

> #Look at confusion matrix

> Conf\_matrix\_Te = table(observed = test[,15], predicted = NB\_Predictions\_T)

> confusionMatrix(Conf\_matrix\_Te)

Confusion Matrix and Statistics

predicted

observed 1 2

1 1394 49

2 156 68

Accuracy : 0.877

95% CI : (0.8603, 0.8924)

No Information Rate : 0.9298

P-Value [Acc > NIR] : 1

Kappa : 0.3378

Mcnemar's Test P-Value : 1.328e-13

Sensitivity : 0.8994

Specificity : 0.5812

Pos Pred Value : 0.9660

Neg Pred Value : 0.3036

Prevalence : 0.9298

Detection Rate : 0.8362

Detection Prevalence : 0.8656

Balanced Accuracy : 0.7403

'Positive' Class : 1

Here, the accuracy is 87.7% and False Negative rate is 69.6%.

## 6. K-Fold Cross Validation

**K-Fold Cross Validation** is a common type of cross validation that is widely used in machine learning.

K-fold cross validation is performed as per the following steps:

1. Partition the original training data set into k equal subsets. Each subset is called a **fold**. Let the folds be named as f1, f2, …, fk .
2. For i = 1 to i = k
   1. Keep the fold fi as Validation set and keep all the remaining *k-1* folds in the Cross validation training set.
   2. Train your machine learning model using the cross validation training set and calculate the accuracy of your model by validating the predicted results against the validation set.
3. Estimate the accuracy of your machine learning model by averaging the accuracies derived in all the*k* cases of cross validation.

In the k-fold cross validation method, all the entries in the original training data set are used for both training as well as validation. Also, each entry is used for validation just once.

Generally, the value of *k* is taken to be 10, but it is not a strict rule, and *k* can take any value.

1. K-folds for Decision Tree:- here, we use c5.0 model.
2. > #K-folda cross-validation for Decision Tree
3. > # define training control
4. > train\_control<- trainControl(method="cv", number=10)
5. > # train the model
6. > model<- C5.0(Churn~., data=train, trControl=train\_control, method="class")
7. > # make predictions
8. > predictions<- predict(model,validation)
9. > # append predictions
10. > validation\_D<- cbind(validation,predictions)
11. > # summarize results
12. > confusionMatrix<- confusionMatrix(validation\_D$predictions,validation$Churn)
13. > confusionMatrix
14. Confusion Matrix and Statistics
15. Reference
16. Prediction 1 2
17. 1 463 21
18. 2 6 60
20. Accuracy : 0.9509
21. 95% CI : (0.9294, 0.9674)
22. No Information Rate : 0.8527
23. P-Value [Acc > NIR] : 1.496e-13
25. Kappa : 0.7883
26. Mcnemar's Test P-Value : 0.007054
28. Sensitivity : 0.9872
29. Specificity : 0.7407
30. Pos Pred Value : 0.9566
31. Neg Pred Value : 0.9091
32. Prevalence : 0.8527
33. Detection Rate : 0.8418
34. Detection Prevalence : 0.8800
35. Balanced Accuracy : 0.8640
37. 'Positive' Class : 1

Here, the accuracy for validation data is 95.1% and False Negative Rate is 9.1%.

Now, for the test data.

> # make predictions for test cases

> predictions\_T<- predict(model,test)

> # append predictions

> test\_D<- cbind(test,predictions\_T)

> # summarize results

> confusionMatrix\_T<- confusionMatrix(test\_D$predictions,test$Churn)

> confusionMatrix\_T

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 1430 69

2 13 155

Accuracy : 0.9508

95% CI : (0.9393, 0.9607)

No Information Rate : 0.8656

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7636

Mcnemar's Test P-Value : 1.25e-09

Sensitivity : 0.9910

Specificity : 0.6920

Pos Pred Value : 0.9540

Neg Pred Value : 0.9226

Prevalence : 0.8656

Detection Rate : 0.8578

Detection Prevalence : 0.8992

Balanced Accuracy : 0.8415

'Positive' Class : 1

Here, the accuracy for the test data is 95.1% and False Negative Rate is 7.74%.

b. k-folds for Random forest:- Here, we use randomForest model to model the data.

> #K-folda cross-validation for Random Forest

> # define training control

> train\_control1<- trainControl(method="cv", number=10)

> # train the model

> model1<- randomForest(Churn~., data=train, trControl1=train\_control1, method="class")

> # make predictions

> predictions\_R<- predict(model1,validation)

> # append predictions

> validation\_R<- cbind(validation,predictions\_R)

> # summarize results

> confusionMatrix\_R<- confusionMatrix(validation\_R$predictions\_R,validation$Churn)

> confusionMatrix\_R

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 469 0

2 0 81

Accuracy : 1

95% CI : (0.9933, 1)

No Information Rate : 0.8527

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 1

Mcnemar's Test P-Value : NA

Sensitivity : 1.0000

Specificity : 1.0000

Pos Pred Value : 1.0000

Neg Pred Value : 1.0000

Prevalence : 0.8527

Detection Rate : 0.8527

Detection Prevalence : 0.8527

Balanced Accuracy : 1.0000

'Positive' Class : 1

Here, the accuracy is 100% on validation data and False Negative Rate is 0%.

Now, for the test case:-

> # make predictions for test cases

> predictions\_RF<- predict(model1,test)

> # append predictions

> test\_R<- cbind(test,predictions\_RF)

> # summarize results

> confusionMatrix\_DT<- confusionMatrix(test\_R$predictions\_RF,test$Churn)

> confusionMatrix\_DT

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 1410 67

2 33 157

Accuracy : 0.94

95% CI : (0.9275, 0.9509)

No Information Rate : 0.8656

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7245

Mcnemar's Test P-Value : 0.0009668

Sensitivity : 0.9771

Specificity : 0.7009

Pos Pred Value : 0.9546

Neg Pred Value : 0.8263

Prevalence : 0.8656

Detection Rate : 0.8458

Detection Prevalence : 0.8860

Balanced Accuracy : 0.8390

'Positive' Class : 1

Here, the accuracy is 94% for test data and False Negative Rate is 17.4%.

**MODEL SELECTION:-** We have used various machine learning model to train the data and to get the highest accuracy and minimum False Negative Rate. So, after analyzing the model the model k-folds cross validation Decision Tree model is best model in respect with Accuracy and False Negative Rate.

**COMPLETE R CODE:-**

#Remove all stored object

rm(list = ls())

#Set working directory

setwd("E:/")

#Load the libraries

x = c("plyr","dplyr","ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",

"MASS", "rpart", "gbm","corrplot","cowplot","gmodels", "ROSE", 'sampling', 'DataCombine', 'inTrees')

lapply(x, require, character.only = TRUE)

rm(x)

#Load the Data

train=read.csv("Train\_data.csv",header = T,na.strings = c(" ", "", "NA"))

test=read.csv("Test\_data.csv",header = T,na.strings = c(" ", "", "NA"))

#Understanding the data

dim(train)

dim(test)

colnames(train)

colnames(test)

str(train)

str(test)

##################EXPLORATORY DATA ANALYSIS###############################################

##################UNIVARIATE ANALYSIS#####################################################

train$state=as.factor(train$state)

train$phone.number=as.factor(train$phone.number)

train$international.plan=as.factor(train$international.plan)

train$voice.mail.plan=as.factor(train$voice.mail.plan)

train$Churn=as.factor(train$Churn)

test$state=as.factor(test$state)

test$phone.number=as.factor(test$phone.number)

test$international.plan=as.factor(test$international.plan)

test$voice.mail.plan=as.factor(test$voice.mail.plan)

test$Churn=as.factor(test$Churn)

prop.table(table(train$Churn))

prop.table(table(test$Churn))

ggplot(train %>% group\_by(Churn) %>% summarise(Count = n())) +

geom\_bar(aes(Churn, Count), stat = "identity", fill = "coral1") +

xlab("") +

geom\_label(aes(Churn, Count, label = Count), vjust = 0.5) +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

ggtitle("Churn")

# plot for state

p1 = ggplot(train %>% group\_by(state) %>% summarise(Count = n())) +

geom\_bar(aes(state, Count), stat = "identity", fill = "coral1") +

xlab("") +

geom\_label(aes(state, Count, label = Count), vjust = 0.5) +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

ggtitle("State")

# plot for international plan

p2 = ggplot(train %>% group\_by(international.plan) %>% summarise(Count = n())) +

geom\_bar(aes(international.plan, Count), stat = "identity", fill = "coral1") +

geom\_label(aes(international.plan, Count, label = Count), vjust = 0.5) +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

ggtitle("International Plan")

# plot for voice mail plan

p3 = ggplot(train %>% group\_by(voice.mail.plan) %>% summarise(Count = n())) +

geom\_bar(aes(voice.mail.plan, Count), stat = "identity", fill = "coral1") +

xlab("") +

geom\_label(aes(voice.mail.plan, Count, label = Count), vjust = 0.5) +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

ggtitle("Voice Mail Plan")

second\_row = plot\_grid(p2, p3, nrow = 1)

plot\_grid(p1, second\_row, ncol = 1)

p4 = ggplot(train) + geom\_histogram(aes(account.length), binwidth = 10, fill = "blue")

p5 = ggplot(train) + geom\_histogram(aes(number.vmail.messages), binwidth = 5, fill = "blue")

p6 = ggplot(train) + geom\_histogram(aes(total.day.minutes), binwidth = 10, fill = "blue")

p7 = ggplot(train) + geom\_histogram(aes(total.day.calls), binwidth = 10, fill = "blue")

plot\_grid(p4, p5, p6,p7, nrow = 2) # plot\_grid() from cowplot package

p8 = ggplot(train) + geom\_histogram(aes(total.day.charge), binwidth = 5, fill = "blue")

p9 = ggplot(train) + geom\_histogram(aes(total.eve.minutes), binwidth = 10, fill = "blue")

p10 = ggplot(train) + geom\_histogram(aes(total.eve.calls), binwidth = 10, fill = "blue")

p11 = ggplot(train) + geom\_histogram(aes(total.eve.charge), binwidth = 1, fill = "blue")

plot\_grid(p8, p9, p10,p11, nrow = 2) # plot\_grid() from cowplot package

p12 = ggplot(train) + geom\_histogram(aes(number.customer.service.calls), binwidth = 1, fill = "blue")

p13 = ggplot(train) + geom\_histogram(aes(total.night.minutes), binwidth = 10, fill = "blue")

p14 = ggplot(train) + geom\_histogram(aes(total.night.calls), binwidth = 10, fill = "blue")

p15 = ggplot(train) + geom\_histogram(aes(total.night.charge), binwidth = 1, fill = "blue")

plot\_grid(p12, p13, p14,p15, nrow = 2) # plot\_grid() from cowplot package

p16 = ggplot(train) + geom\_histogram(aes(total.intl.minutes), binwidth = 1, fill = "blue")

p17 = ggplot(train) + geom\_histogram(aes(total.intl.calls), binwidth = 1, fill = "blue")

p18 = ggplot(train) + geom\_histogram(aes(total.intl.charge), binwidth = 0.5, fill = "blue")

plot\_grid(p16, p17, p18, nrow = 2) # plot\_grid() from cowplot package

#########################MULTIVARIATE ANALYSIS##############################

#USING crosstable function from gmodels

CrossTable(train$state,train$Churn)

#plot a stacked barcharts

ggplot(train,aes(state,fill=Churn))+geom\_bar()+labs(title="State vs Churn",x="state",y="count")+theme\_bw()

ggplot(train,aes(international.plan,fill=Churn))+geom\_bar()+labs(title="Intl plan vs Churn",

x="international plan",y="count")+theme\_bw()

ggplot(train,aes(voice.mail.plan,fill=Churn))+geom\_bar()+labs(title="Voice plan vs Churn",

x="Voice mail plan",y="count")+theme\_bw()

##################Missing Value###############

colSums(is.na(train))

colSums(is.na(train))

sum(is.na(train))

sum(is.na(test))

############################################Outlier Analysis#############################################

# ## BoxPlots - Distribution and Outlier Check

numeric\_index = sapply(train,is.numeric) #selecting only numeric

numeric\_data = train[,numeric\_index]

cnames = colnames(numeric\_data)

for(i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = "Churn"), data = subset(train))+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="Churn")+

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

}

# ## Plotting plots together

gridExtra::grid.arrange(gn1,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,gn15,ncol=3)

gridExtra::grid.arrange(gn16,ncol=1)

##################################Feature Selection################################################

## Correlation Plot

corrgram(train[,numeric\_index], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

## Chi-squared Test of Independence

factor\_index = sapply(train,is.factor)

factor\_data = train[,factor\_index]

for (i in 1:5)

{

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

print(chisq.test(table(factor\_data[,i])))

}

test1=test

## Dimension Reduction

train = subset(train,

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

phone.number))

test = subset(test,

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

phone.number))

##Data Manupulation; convert string categories into factor numeric

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

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

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

}

}

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

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

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

}

}

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

#Divide data into train and test using stratified sampling method

set.seed(1234)

train.index = createDataPartition(train$Churn, p = .80, list = FALSE)

train = train[ train.index,]

validation = train[-train.index,]

##Decision tree for classification

#Develop Model on training data

C50\_model = C5.0(Churn ~., train, trials = 100, rules = TRUE)

#Summary of DT model

summary(C50\_model)

#Lets predict for validation cases

C50\_Predictions = predict(C50\_model, validation[,-15], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_C50 = table(validation$Churn, C50\_Predictions)

confusionMatrix(ConfMatrix\_C50)

#Accuracy=98.9%

#FNR=7.7%

#Lets Predict for test case

test\_D = predict(C50\_model, test[,-15], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_test\_D = table(test$Churn, test\_D)

confusionMatrix(ConfMatrix\_test\_D)

#Accuracy=95.4

#FNR=29.4

###Random Forest

RF\_model = randomForest(Churn ~ .,train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest

#transform rf object to an inTrees' format

treeList = RF2List(RF\_model)

#

# #Extract rules

exec = extractRules(treeList, train[,-15]) # R-executable conditions

#

# #Visualize some rules

exec[1:2,]

#

# #Make rules more readable:

readableRules = presentRules(exec, colnames(train))

readableRules[1:2,]

#

# #Get rule metrics

ruleMetric = getRuleMetric(exec, train[,-15], train$Churn) # get rule metrics

#

# #evaulate few rules

ruleMetric[1:2,]

#Presdict validation data using random forest model

RF\_Predictions = predict(RF\_model, validation[,-15])

##Evaluate the performance of classification model

ConfMatrix\_RF = table(validation$Churn, RF\_Predictions)

confusionMatrix(ConfMatrix\_RF)

#Accuracy=100%

#FNR=0%

#Random forest for test case

RF\_Test=predict(RF\_model,test[,-15])

##Evaluate the performance of classification model

ConfMatrix\_RF\_Test = table(test$Churn, RF\_Test)

confusionMatrix(ConfMatrix\_RF\_Test)

#Accuracy=94.5

#FNR=30.4

#Logistic Regression

logit\_model = glm(Churn ~ ., data = train, family = "binomial")

#summary of the model

summary(logit\_model)

#predict using logistic regression

logit\_Predictions = predict(logit\_model, newdata = validation, type = "response")

#convert prob

logit\_Predictions = ifelse(logit\_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model

ConfMatrix\_RF = table(validation$Churn, logit\_Predictions)

#Accuracy

sum(diag(ConfMatrix\_RF))/nrow(validation)

#Accuracy: 86.5

#FNR: 72.8

#prediction for test cases

logit\_Predictions\_Test = predict(logit\_model, newdata = test, type = "response")

#convert prob

logit\_Predictions\_Test = ifelse(logit\_Predictions\_Test > 0.5, 1, 0)

##Evaluate the performance of classification model

ConfMatrix\_RF\_T = table(test$Churn, logit\_Predictions\_Test)

#Accuracy

sum(diag(ConfMatrix\_RF\_T))/nrow(test)

#Accuracy: 86.7

#FNR: 75.4

##KNN Implementation

library(class)

#Predict test data

KNN\_Predictions = knn(train[, 1:14], validation[, 1:14], train$Churn, k = 7)

#Confusion matrix

Conf\_matrix = table(KNN\_Predictions, validation$Churn)

#Accuracy

sum(diag(Conf\_matrix))/nrow(validation)

#Accuracy = 90.4

#FNR = 6.25

#Prediction for test case

KNN\_Predictions\_T = knn(train[, 1:14], test[, 1:14], train$Churn, k = 7)

Conf\_matrix\_T = table(KNN\_Predictions\_T,test$Churn)

#Accuracy

sum(diag(Conf\_matrix\_T))/nrow(test)

#Accuracy=89.2

#FNR=18.6

#naive Bayes

library(e1071)

#Develop model

NB\_model = naiveBayes(Churn ~ ., data = train)

#predict on test cases #raw

NB\_Predictions = predict(NB\_model, validation[,1:14], type = 'class')

#Look at confusion matrix

Conf\_matrix\_V = table(observed = validation[,15], predicted = NB\_Predictions)

confusionMatrix(Conf\_matrix\_V)

#Accuracy = 89.1

#FNR = 59.3

#Prediction for test case

NB\_Predictions\_T = predict(NB\_model, test[,1:14], type = 'class')

#Look at confusion matrix

Conf\_matrix\_Te = table(observed = test[,15], predicted = NB\_Predictions\_T)

confusionMatrix(Conf\_matrix\_Te)

#Accuracy = 87.7

#FNR = 69.6

#K-folda cross-validation for Decision Tree

# define training control

train\_control<- trainControl(method="cv", number=10)

# train the model

model<- C5.0(Churn~., data=train, trControl=train\_control, method="class")

# make predictions

predictions<- predict(model,validation)

# append predictions

validation\_D<- cbind(validation,predictions)

# summarize results

confusionMatrix<- confusionMatrix(validation\_D$predictions,validation$Churn)

confusionMatrix

#Accuracy=95.1

#FNR=9.1%

# make predictions for test cases

predictions\_T<- predict(model,test)

# append predictions

test\_D<- cbind(test,predictions\_T)

# summarize results

confusionMatrix\_T<- confusionMatrix(test\_D$predictions,test$Churn)

confusionMatrix\_T

#Accuracy=95.1

#FNR=7.74

#K-folda cross-validation for Random Forest

# define training control

train\_control1<- trainControl(method="cv", number=10)

# train the model

model1<- randomForest(Churn~., data=train, trControl1=train\_control1, method="class")

# make predictions

predictions\_R<- predict(model1,validation)

# append predictions

validation\_R<- cbind(validation,predictions\_R)

# summarize results

confusionMatrix\_R<- confusionMatrix(validation\_R$predictions\_R,validation$Churn)

confusionMatrix\_R

#Accuracy=100

#FNR=0%

# make predictions for test cases

predictions\_RF<- predict(model1,test)

# append predictions

test\_R<- cbind(test,predictions\_RF)

# summarize results

confusionMatrix\_DT<- confusionMatrix(test\_R$predictions\_RF,test$Churn)

confusionMatrix\_DT

#Accuracy=94

#FNR=17.4

test1$Churn=predictions\_T

test1$Churn=as.numeric(test1$Churn)

test1$Churn[test1$Churn==1]="False"

test1$Churn[test1$Churn==2]="True"

write.csv(test1,file = "Sample.csv",row.names = F)

**COMPLETE PYTHON CODE:-**

import os

os.chdir("E:")

import pandas as pd

import numpy as np # For mathematical calculations

import seaborn as sns # For data visualization

import matplotlib as mpl

print('Matplotlib version: ', mpl.\_\_version\_\_) # >= 2.0.0

import matplotlib.pyplot as plt # For plotting graphs

%matplotlib inline

import warnings # To ignore any warnings

warnings.filterwarnings("ignore")

print(plt.style.available)

mpl.style.use(['ggplot']) # optional: for ggplot-like style

train=pd.read\_csv("Train\_data.csv",sep=',')

test=pd.read\_csv("Test\_data.csv",sep=',')

train.shape, test.shape

train.info()

test.info()

train['Churn'].value\_counts()

train['Churn'].value\_counts(normalize=True)

train.head()

train['Churn'].value\_counts(normalize=True).plot.bar()

train['state'].value\_counts(normalize=True).plot.bar(figsize=(30,20))

train['international plan'].value\_counts(normalize=True).plot.bar()

train['voice mail plan'].value\_counts(normalize=True).plot.bar()

count,bin\_edges=np.histogram(train['account length'])

train['account length'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['number vmail messages'])

train['number vmail messages'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total day minutes'])

train['total day minutes'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total day calls'])

train['total day calls'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total day charge'])

train['total day charge'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total eve calls'])

train['total eve calls'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total eve minutes'])

train['total eve minutes'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.8,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total eve charge'])

train['total eve charge'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total intl minutes'])

train['total intl minutes'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total intl calls'])

train['total intl calls'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total intl charge'])

train['total intl charge'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

train['number customer service calls'].value\_counts(normalize=True).plot.bar()

count,bin\_edges=np.histogram(train['number customer service calls'])

train['number customer service calls'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total night minutes'])

train['total night minutes'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total night calls'])

train['total night calls'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

count,bin\_edges=np.histogram(train['total night charge'])

train['total night charge'].plot(kind='hist',figsize=(10,6),bins=10,alpha=0.6,

xticks=bin\_edges,

color=['coral', 'darkslateblue', 'mediumseagreen'],

stacked=True)

print(count)

print(bin\_edges)

State=pd.crosstab(train['state'],train['Churn'])

State.div(State.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(20,8))

Intl=pd.crosstab(train['international plan'],train['Churn'])

Intl.div(Intl.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))

Vmp=pd.crosstab(train['voice mail plan'],train['Churn'])

Vmp.div(Vmp.sum(1).astype(float), axis=0).plot(kind="bar", stacked=True, figsize=(4,4))

train.groupby('Churn')['total day calls'].count().plot.bar()

train.groupby('Churn')['total eve calls'].count().plot.bar()

train.groupby('Churn')['total night calls'].count().plot.bar()

train.isnull().sum()

test.isnull().sum()

#Select the Continous Variable

cnames=["account length","area code","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"]

##Correlation analysis

#Correlation plot

df\_corr = train.loc[:,cnames]

#Set the width and hieght of the plot

f, ax = plt.subplots(figsize=(15, 8))

#Generate correlation matrix

corr = df\_corr.corr()

#Plot using seaborn library

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),

square=True, ax=ax)

#Chisquare test of independence

#Save categorical variables

cat\_names=["state","international plan","voice mail plan"]

#loop for chi square values

from scipy.stats import chi2\_contingency

for i in cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(train['Churn'], train[i]))

print(p)

train=train.drop(['area code','phone number','total day charge','total eve charge','total night charge',

'total intl charge'],axis=1)

features = train.drop(["Churn"], axis=1).columns

test1= test.copy()

test=test.drop(['phone number','area code','total day charge','total eve charge','total night charge',

'total intl charge'],axis=1)

train.shape, test.shape

train.head()

from sklearn.preprocessing import LabelEncoder

number=LabelEncoder()

train['international plan']=number.fit\_transform(train['international plan'].astype('str'))

train['voice mail plan']=number.fit\_transform(train['voice mail plan'].astype('str'))

train['Churn']=number.fit\_transform(train['Churn'].astype('str'))

train['state']=number.fit\_transform(train['state'].astype('str'))

test['international plan']=number.fit\_transform(test['international plan'].astype('str'))

test['voice mail plan']=number.fit\_transform(test['voice mail plan'].astype('str'))

test['Churn']=number.fit\_transform(test['Churn'].astype('str'))

test['state']=number.fit\_transform(test['state'].astype('str'))

test.head()

X = train.drop('Churn',1)

y = train.Churn

x1=test.drop('Churn',1)

y1=test.Churn

from sklearn.model\_selection import train\_test\_split

x\_train, x\_cv, y\_train, y\_cv = train\_test\_split(X,y, test\_size =0.25)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_curve

from IPython.display import display, HTML

# Set up our RandomForestClassifier instance and fit to data

clf = RandomForestClassifier(n\_estimators=30)

clf.fit(x\_train,y\_train)

# Make predictions

predictions = clf.predict(x\_cv)

probs = clf.predict\_proba(x\_cv)

score = clf.score(x\_cv, y\_cv)

print("Accuracy: ", score)

get\_ipython().magic('matplotlib inline')

confusion\_matrix = pd.DataFrame(

confusion\_matrix(y\_cv, predictions),

columns=["Predicted False", "Predicted True"],

index=["Actual False", "Actual True"]

)

display(confusion\_matrix)

# Calculate the fpr and tpr for all thresholds of the classification

fpr, tpr, threshold = roc\_curve(y\_cv, probs[:,1])

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

#FNR=36

fig = plt.figure(figsize=(20, 18))

ax = fig.add\_subplot(111)

df\_f = pd.DataFrame(clf.feature\_importances\_, columns=["importance"])

df\_f["labels"] = features

df\_f.sort\_values("importance", inplace=True, ascending=False)

display(df\_f.head(5))

index = np.arange(len(clf.feature\_importances\_))

bar\_width = 0.5

rects = plt.barh(index , df\_f["importance"], bar\_width, alpha=0.4, color='b', label='Main')

plt.yticks(index, df\_f["labels"])

plt.show()

predictions\_test = clf.predict(x1)

#build confusion matrix

# from sklearn.metrics import confusion\_matrix

# CM = confusion\_matrix(x1, y1)

CM = pd.crosstab(y1, predictions\_test)

#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

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 94.7

#FNR: 35.7

from sklearn.model\_selection import StratifiedKFold

from sklearn.metrics import accuracy\_score

i=1

kf = StratifiedKFold(n\_splits=5,random\_state=1,shuffle=True)

for train\_index,test\_index in kf.split(X,y):

print('\n{} of kfold {}'.format(i,kf.n\_splits))

xtr,xvl = X.loc[train\_index],X.loc[test\_index]

ytr,yvl = y[train\_index],y[test\_index]

model = RandomForestClassifier(random\_state=1, max\_depth=10)

model.fit(xtr, ytr)

pred\_test = model.predict(xvl)

score = accuracy\_score(yvl,pred\_test)

print('accuracy\_score',score)

i+=1

pred\_test = model.predict(x1)

# CM = confusion\_matrix(y1, y1)

CM1 = pd.crosstab(y1, pred\_test)

#let us save TP, TN, FP, FN

TN = CM1.iloc[0,0]

FN = CM1.iloc[1,0]

TP = CM1.iloc[1,1]

FP = CM1.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 93.5

#FNR: 43.3

#Import Libraries for decision tree

from sklearn import tree

#Decision Tree

C50\_model = tree.DecisionTreeClassifier(criterion='entropy').fit(x\_train, y\_train)

#predict new test cases

C50\_Predictions = C50\_model.predict(x\_cv)

# CM = confusion\_matrix(y\_cv, C50\_Predictions)

CM2 = pd.crosstab(y\_cv, C50\_Predictions)

#let us save TP, TN, FP, FN

TN = CM2.iloc[0,0]

FN = CM2.iloc[1,0]

TP = CM2.iloc[1,1]

FP = CM2.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 92.4

#FNR: 28

C50\_Test=C50\_model.predict(x1)

# CM = confusion\_matrix(y1, C50\_Test)

CM3 = pd.crosstab(y1, C50\_Test)

#let us save TP, TN, FP, FN

TN = CM3.iloc[0,0]

FN = CM3.iloc[1,0]

TP = CM3.iloc[1,1]

FP = CM3.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 92.0

#FNR: 33

i=1

kf = StratifiedKFold(n\_splits=5,random\_state=1,shuffle=True)

for train\_index,test\_index in kf.split(X,y):

print('\n{} of kfold {}'.format(i,kf.n\_splits))

xtr,xvl = X.loc[train\_index],X.loc[test\_index]

ytr,yvl = y[train\_index],y[test\_index]

model = tree.DecisionTreeClassifier(random\_state=1)

model.fit(xtr, ytr)

pred\_test = model.predict(xvl)

score = accuracy\_score(yvl,pred\_test)

print('accuracy\_score',score)

i+=1

pred\_test1 = model.predict(x1)

# CM = confusion\_matrix(y1, pred\_test1)

CM4 = pd.crosstab(y1, pred\_test1)

#let us save TP, TN, FP, FN

TN = CM4.iloc[0,0]

FN = CM4.iloc[1,0]

TP = CM4.iloc[1,1]

FP = CM4.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 91

#FNR: 30.4

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(x\_train, y\_train)

pred\_cv = model.predict(x\_cv)

accuracy\_score(y\_cv,pred\_cv)

# CM = confusion\_matrix(y\_cv, pred\_cv)

CM5 = pd.crosstab(y\_cv, pred\_cv)

#let us save TP, TN, FP, FN

TN = CM5.iloc[0,0]

FN = CM5.iloc[1,0]

TP = CM5.iloc[1,1]

FP = CM5.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 84.9

#FNR: 90.4

pred\_L = model.predict(x1)

# CM = confusion\_matrix(y1, pred\_L)

CM6 = pd.crosstab(y1, pred\_L)

#let us save TP, TN, FP, FN

TN = CM6.iloc[0,0]

FN = CM6.iloc[1,0]

TP = CM6.iloc[1,1]

FP = CM6.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 87.4

#FNR: 83.5

# CM = confusion\_matrix(y1, pred\_L)

CM6 = pd.crosstab(y1, pred\_L)

#let us save TP, TN, FP, FN

TN = CM6.iloc[0,0]

FN = CM6.iloc[1,0]

TP = CM6.iloc[1,1]

FP = CM6.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 87.4

#FNR: 83.5

i=1

kf = StratifiedKFold(n\_splits=5,random\_state=1,shuffle=True)

for train\_index,test\_index in kf.split(X,y):

print('\n{} of kfold {}'.format(i,kf.n\_splits))

xtr,xvl = X.loc[train\_index],X.loc[test\_index]

ytr,yvl = y[train\_index],y[test\_index]

model = LogisticRegression(random\_state=1)

model.fit(xtr, ytr)

pred\_test = model.predict(xvl)

score = accuracy\_score(yvl,pred\_test)

print('accuracy\_score',score)

i+=1

pred\_LV = model.predict(x1)

pred=model.predict\_proba(xvl)[:,1]

# CM = confusion\_matrix(y1, pred\_LV)

CM6 = pd.crosstab(y1, pred\_LV)

#let us save TP, TN, FP, FN

TN = CM6.iloc[0,0]

FN = CM6.iloc[1,0]

TP = CM6.iloc[1,1]

FP = CM6.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Results

#Accuracy: 87.34

#FNR: 82.14

#Naive Bayes

from sklearn.naive\_bayes import GaussianNB

#Naive Bayes implementation

NB\_model = GaussianNB().fit(x\_train, y\_train)

#predict x\_cv

NB\_Predictions = NB\_model.predict(x\_cv)

#Build confusion matrix

CM7 = pd.crosstab(y\_cv, NB\_Predictions)

#let us save TP, TN, FP, FN

TN = CM7.iloc[0,0]

FN = CM7.iloc[1,0]

TP = CM7.iloc[1,1]

FP = CM7.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Accuracy: 84.5

#FNR: 64.8

#predict Test case

NB\_Test=NB\_model.predict(x1)

#Build confusion matrix

CM8 = pd.crosstab(y1, NB\_Test)

#let us save TP, TN, FP, FN

TN = CM8.iloc[0,0]

FN = CM8.iloc[1,0]

TP = CM8.iloc[1,1]

FP = CM8.iloc[0,1]

#check accuracy of model

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

#False Negative rate

print((FN\*100)/(FN+TP))

#Accuracy: 86

#FNR: 61

test1["Churn"]=predictions\_test

test1["Churn"]=test1["Churn"].replace(1,'True')

test1["Churn"]=test1["Churn"].replace(0,'False')

# Writing a csv (output)

test1.to\_csv("TestP.csv", index = False)