Churn Reduction Model

Akansha Jain

2 March 2019

Contents

Chapter 1

Introduction

1.1 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. This problem statement is targeted at enabling churn reduction using analytics concepts.

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

The predictors provided are as follows:

- Account length
- International plan
- Voicemail plan
- Number of voicemail messages
- total day minutes used
- Day calls made
- Total day charge
- Total evening minutes
- Total evening calls
- Total evening charge

- total night minutes
- Total night calls
- Total night charge
- Total international minutes used
- Total international calls made
- Total international charge
- Number of customer service calls made

Target Variable : move: if the customer has moved (1=yes; 0 = no)

1.2 Data

Our task is to build classification models which will predict whether the customer is going to churn or not which means Is he going to leave a company or not. Given below is a sample of the data set that we are using to predict the behavior of customer: Table

Table 1.1 churn reduction sample data (columns 1 to 8)

state	account.length	area.code	phone.number	international.plan	voice.mail.plan	number.vmail.messages	total.day.minutes
AK	1	408	373-1028	no	no	0	175.2
AK	36	408	341-9764	no	yes	30	146.3
AK	36	415	399-1526	yes	yes	19	171.9
AK	41	415	378-7733	no	no	0	159.3
AK	42	415	375-4450	no	no	0	171.0
AK	48	415	389-7073	no	yes	37	211.7
AK	50	408	362-8331	no	no	0	183.6

Table 1.2 churn reduction sample data (columns 9 to 14)

total.day.calls	total.day.charge	total.eve.minutes	total.eve.calls	total.eve.charge	total.night.minutes
74	29.78	151.7	79	12.89	230.5
128	24.87	162.5	80	13.81	129.3
96	29.22	198.4	111	16.86	321.7
66	27.08	125.9	75	10.70	261.9
129	29.07	183.9	96	15.63	130.2
115	35.99	159.9	84	13.59	144.1

Table 1.3 churn reduction sample data (columns 15 to 19)

total.night.calls	total.night.charge	total.intl.minutes	total.intl.calls	total.intl.charge
109	10.37	5.3	3	1.43
109	5.82	14.5	6	3.92
76	14.48	10.5	1	2.84
76	11.79	11.1	5	3.00
90	5.86	4.6	6	1.24

Table 1.4 churn reduction sample data (columns 20 to 21)

number.customer.service.calls	Churn
1	False.
0	False.
1	True.
1	False.
0	False.
1	False.
1	False.

Chapter 2 Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. It include techniques which help to clean and manage data into proper shape. To start this process we will see in the dataset if there is any missing value or not. Now we check if missing value percentage is greater than 30% we compute the values using various methods but if it is less than it we drop it.

2.1.1 Missing Value Treatment:

Missing value means the data points which are not present in the master data. In data first we need to check we have missing value in data or not because there are so many algorithms which are applicable on missing data. To check missing value in my data

First we apply function in R

- table(is.na(data))

Function we apply in python check missing value
data. isnull (). sum ()

In our data there is no missing value so we don't need to apply missing value treatment method like we have method

- mean ()- fill NA with the mean of the variable
- median () – fill NA with the median of the variable
- knn imputation – its algorithm which fill NA by prediction

2.1.2 Variable Transformation:

In this variable transformation we need to change our categorical data in numeric data because maximum classification algorithm only take numeric data some of the algorithm like decision tree can take categorical data too. In decision tree we don't need to apply variable transformation method

Output

Before :the dataset we have before the variable transformation we can see that we can only transform the categorical variable into the numeric dataset.

```
> head(churn_data)
  state account.length area.code phone.number international.plan voice.mail.plan number.vmail.messages
1    AK             1       408     373-1028                 no              no                      0
2    AK            36       408     341-9764                 no             yes                     30
3    AK            36       415     399-1526                yes             yes                     19
4    AK            41       415     378-7733                 no              no                      0
5    AK            42       415     375-4450                 no              no                      0
6    AK            48       415     389-7073                 no             yes                     37
  total.day.minutes total.day.calls total.day.charge total.eve.minutes total.eve.calls total.eve.charge
1             175.2              74            29.78             151.7              79            12.89
2             146.3             128            24.87             162.5              80            13.81
3             171.9              96            29.22             198.4             111            16.86
4             159.3              66            27.08             125.9              75            10.70
5             171.0             129            29.07             183.9              96            15.63
6             211.7             115            35.99             159.9              84            13.59
  total.night.minutes total.night.calls total.night.charge total.intl.minutes total.intl.calls
1               230.5               109              10.37               5.3                3
2               129.3               109               5.82              14.5                6
3               321.7                76              14.48              10.5                1
4               261.9                76              11.79              11.1                5
5               130.2                90               5.86               4.6                6
6               144.1                80               6.48              12.2                1
  total.intl.charge number.customer.service.calls   Churn
1              1.43                             1  False.
2              3.92                             0  False.
3              2.84                             1   True.
4              3.00                             1  False.
5              1.24                             0  False.
6              3.29                             1  False.
>
```

The data set after Variable transformation is

```
>
> ############################################## Explore data analysis ################################
>
> ## Univariate Analysis and Variable Consolidation
> ##Data Manupulation; convert string categories into factor numeric
> for(i in 1:ncol(churn_data)){
+   if(class(churn_data[,i]) == 'factor'){
+      churn_data[,i] = factor(churn_data[,i],
+      labels=(1:length(levels(factor(churn_data[,i])))))
+   }
+ }
> head(churn_data)
  state account.length area.code phone.number international.plan voice.mail.plan number.vmail.messages
1     1             1       408         3279                  1               1                     0
2     1            36       408         2185                  1               2                    30
3     1            36       415         1225                  2               2                    19
4     1            41       415         3470                  1               1                     0
5     1            42       415          792                  1               1                     0
6     1            48       415         3851                  1               2                    37
  total.day.minutes total.day.calls total.day.charge total.eve.minutes total.eve.calls total.eve.charge
1             175.2              74            29.78             151.7              79            12.89
2             146.3             128            24.87             162.5              80            13.81
3             171.9              96            29.22             198.4             111            16.86
4             159.3              66            27.08             125.9              75            10.70
5             171.0             129            29.07             183.9              96            15.63
6             211.7             115            35.99             159.9              84            13.59
  total.night.minutes total.night.calls total.night.charge total.intl.minutes total.intl.calls
1               230.5               109              10.37               5.3                3
2               129.3               109               5.82              14.5                6
3               321.7                76              14.48              10.5                1
4               261.9                76              11.79              11.1                5
5               130.2                90               5.86               4.6                6
6               144.1                80               6.48              12.2                1
  total.intl.charge number.customer.service.calls Churn
1              1.43                             1     1
2              3.92                             0     1
3              2.84                             1     2
4              3.00                             1     1
```

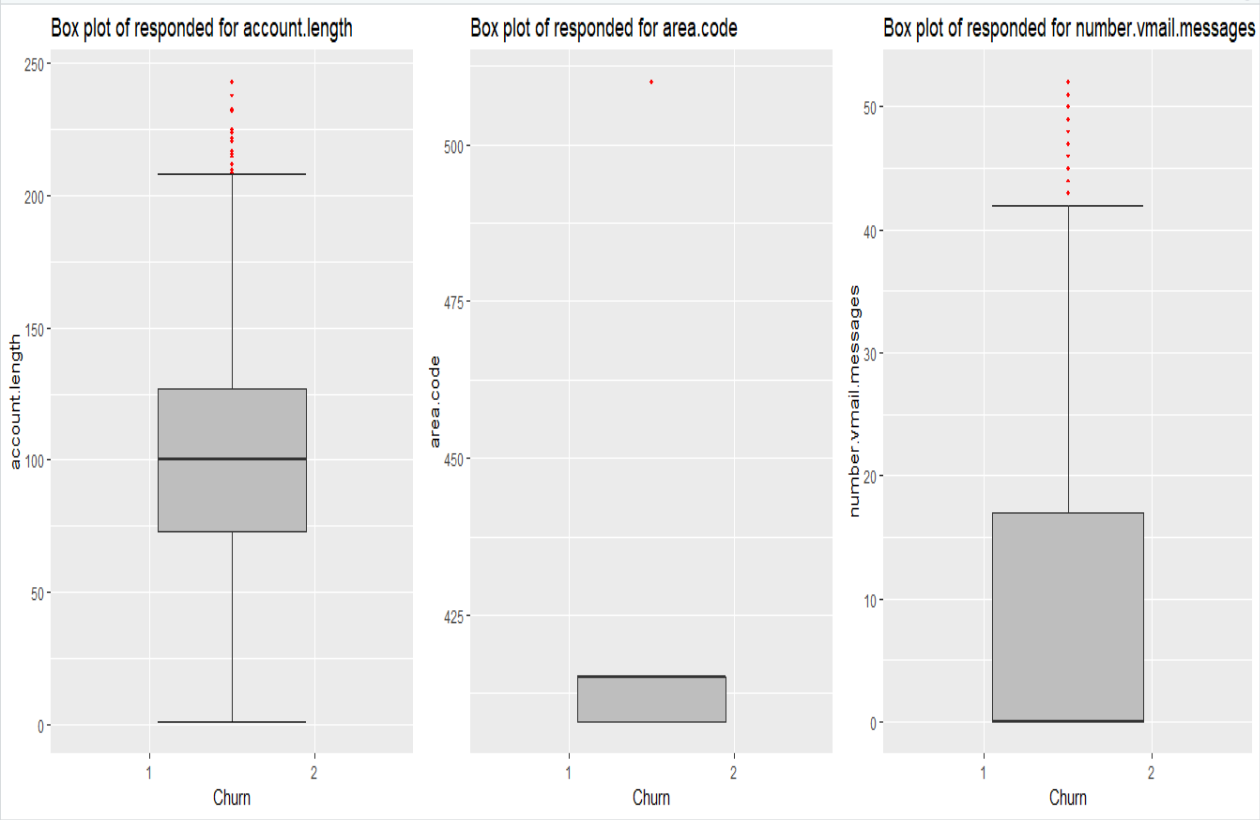
2.1.3 Outlier Analysis:

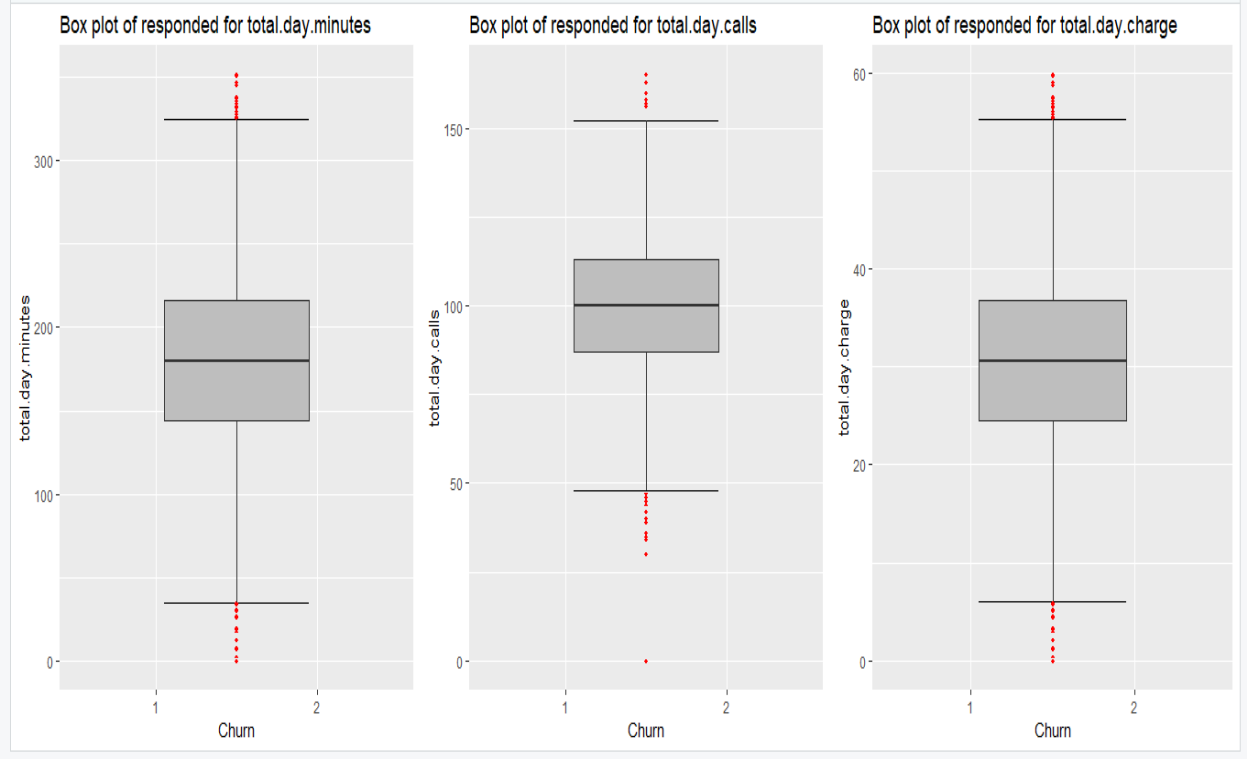
Outliers are points in a dataset that lie far away from the estimated value of the center of the dataset. This estimated center could be either the mean, or median, or percentile. Outlier detection is an important aspect of machine learning algorithms of any sophistication. Because of the fact that outliers can throw off a machine learning algorithm or deflate an assumption about the dataset.

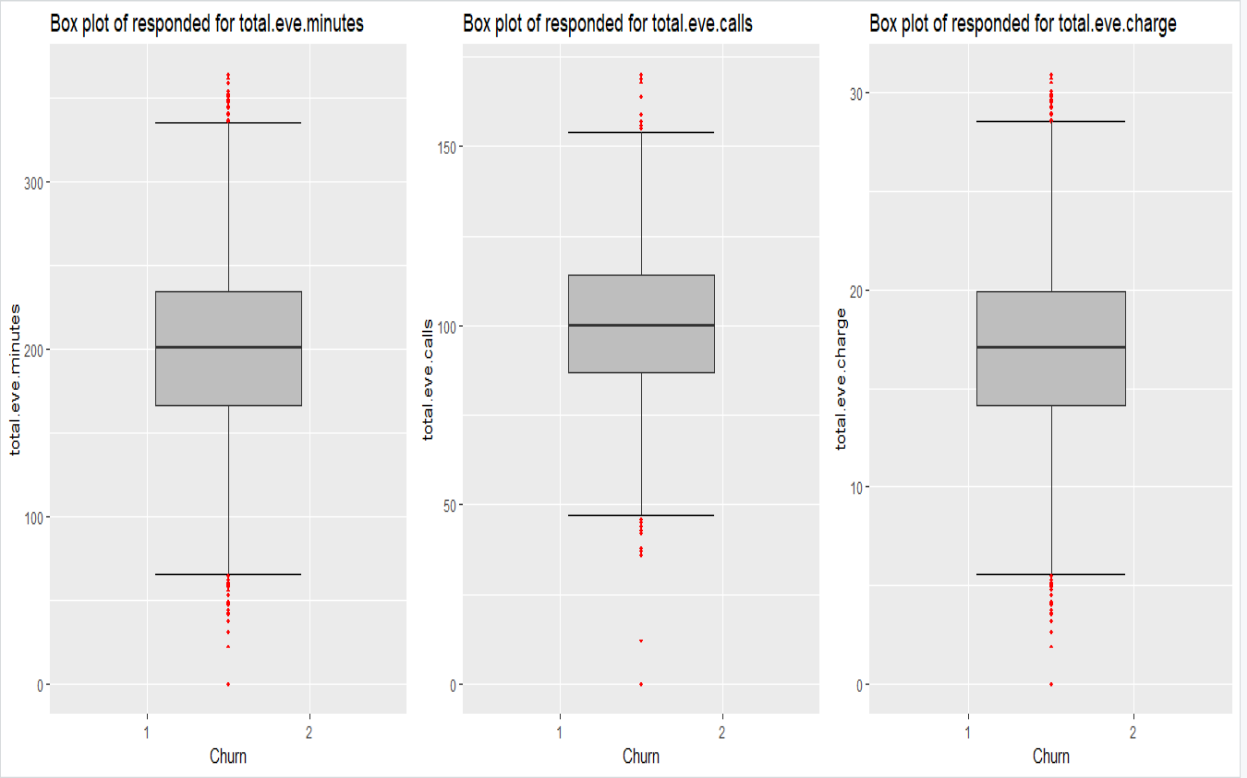
Outliers will be available only in a continuous variable.

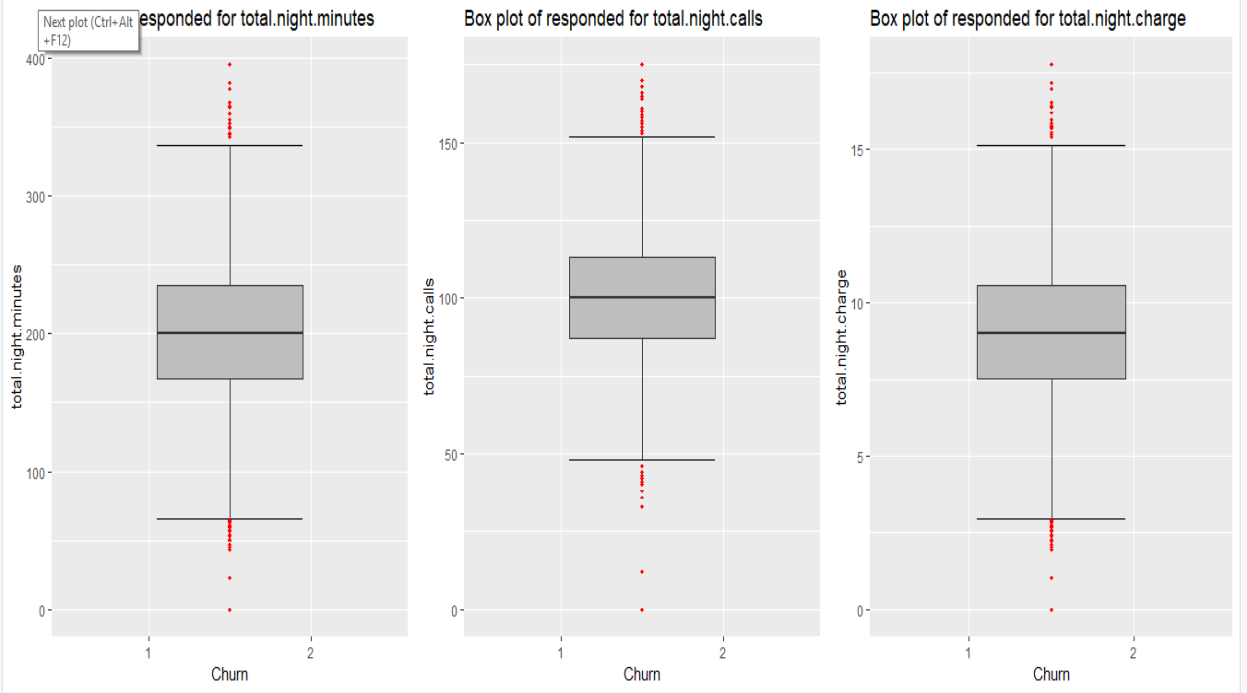
Here we remove the values which are falling beyond outer and inner fence. Boxplot technique help to detect and delete outliers in the dataset. After detecting outliers we remove them by either drop them or replace them by NA and then imputation.

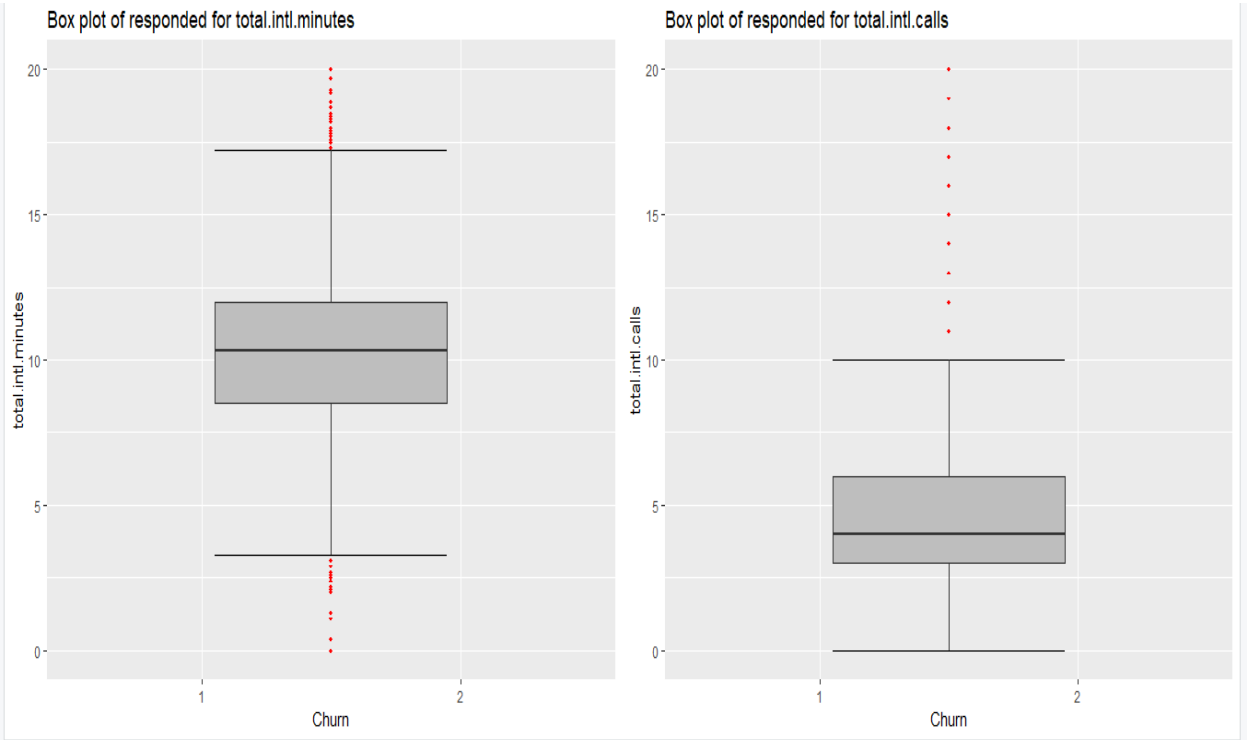
Quality boxplots for each predictor

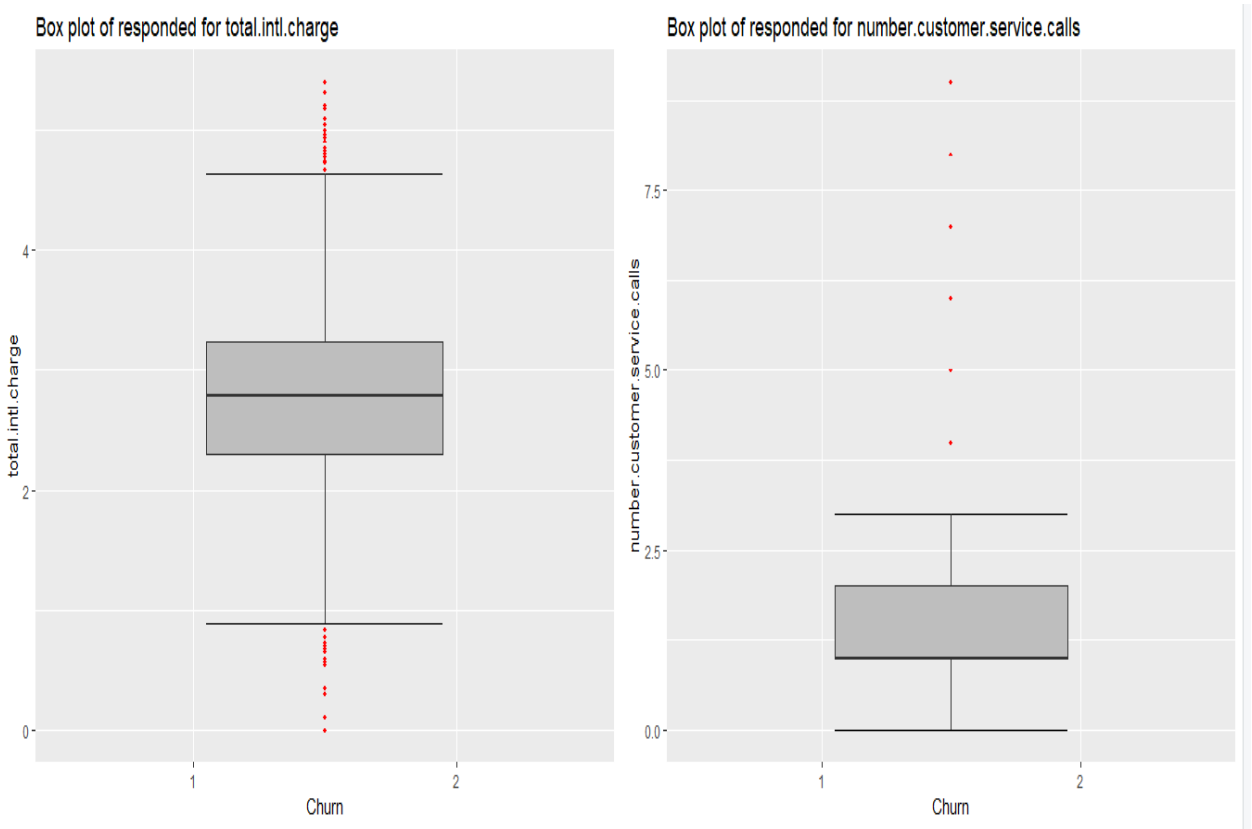

Box plot of responded for account.length
account.length
250
200
150
100
50
0
1
2
Churn
Box plot of responded for area.code
area.code
500
475
450
425
1
2
Churn
Box plot of responded for number.vmail.messages
number.vmail.messages
50
40
30
20
10
0
1
2
Churn


Box plot of responded for total.day.minutes
total.day.minutes
300
200
100
0
1
2
Churn
Box plot of responded for total.day.calls
total.day.calls
150
100
50
0
1
2
Churn
Box plot of responded for total.day.charge
total.day.charge
60
40
20
0
1
2
Churn


Box plot of responded for total.eve.minutes
total.eve.minutes
300
200
100
0
1
2
Churn
Box plot of responded for total.eve.calls
total.eve.calls
150
100
50
0
1
2
Churn
Box plot of responded for total.eve.charge
total.eve.charge
30
20
10
0
1
2
Churn


Next plot (Ctrl+Alt +F12)
esponded for total.night.minutes
total.night.minutes
Churn
Box plot of responded for total.night.calls
total.night.calls
Churn
Box plot of responded for total.night.charge
total.night.charge
Churn


Box plot of responded for total.intl.minutes
total.intl.minutes
Churn
Box plot of responded for total.intl.calls
total.intl.calls
Churn



Above which are in the red color circles are the outliers we have to detect them and remove them by either drop or replace with NA followed by imputation.

I have imputed it using mean and median method as it give more accuracy then knn imputation.

2.1.4 Feature Selection:

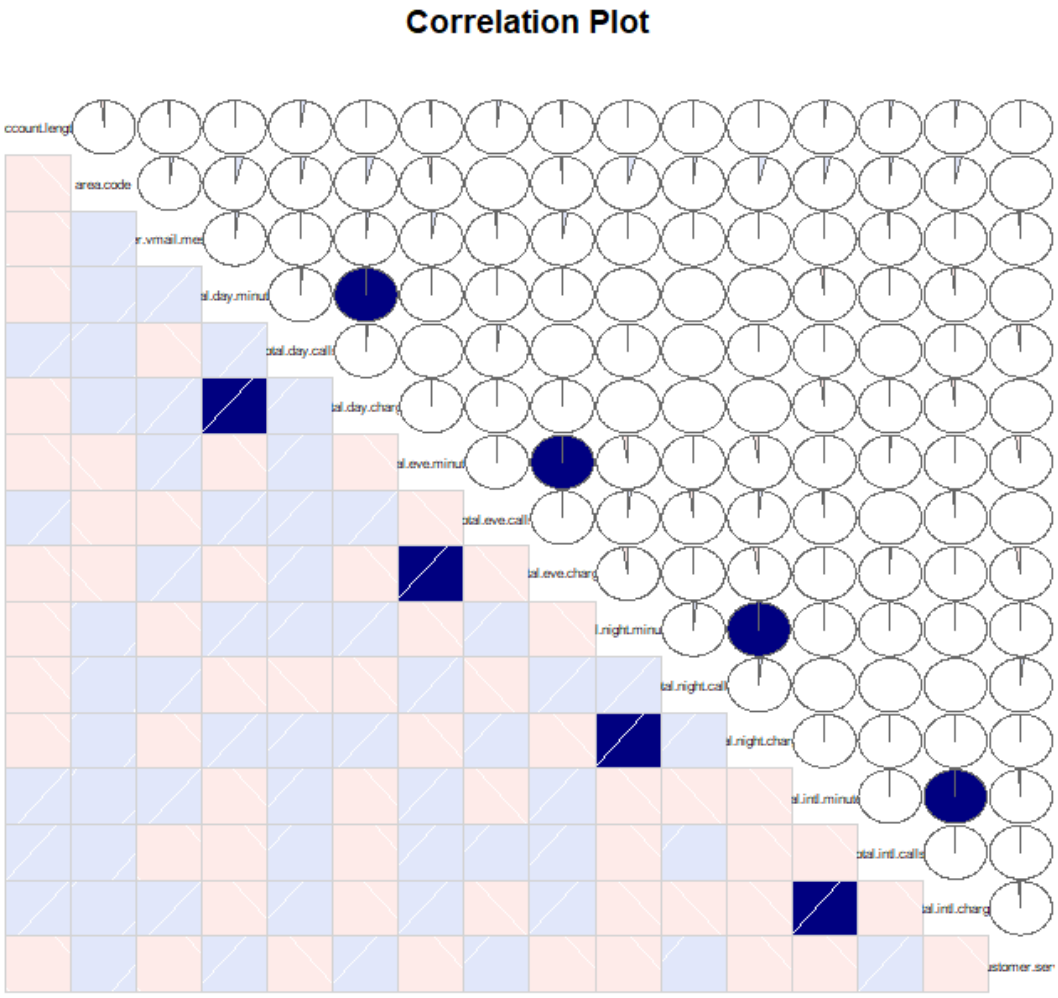
Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. It is a process of finding out the best subset of attributes which better explains the relationship of independent variable with target variables.

Here based on domain experience we have state, area code and phone number which do not have high impact on target variable churn. As leaving of customers from company have nothing to do with these attributes.

There are several methods of doing that. Here we perform two task a. Correlation plot b. Chi-square test .

a. Correlation Plot:

Correlation plot for categorical data we find out the correlation between dependent variable and categorical variable



In this visualization

Extremely orange color indicate that the variable is highly negatively correlated with each other

Extremely navy blue indicate that the variable is highly positively correlated with each other

b. Chi-square test:

It is the best method to measure dependencies between two categorical variables. We find out the p value of the numeric dataset if p-value is greater than>0.05 we neglect that independent variable from the dataset.

This test will convert two categorical variables to contingency table for better representation.

```
>
>  ## Chi-squared Test of Independence
>   factor_index = sapply(churn_data,is.factor)
>  factor_data = churn_data[,factor_index]
>  for (i in 1:4)
+  {
+    print(names(factor_data)[i])
+    print(chisq.test(table(factor_data$Churn,factor_data[,i])),simulate.p.value = TRUE)
+   }
[1] "state"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 96.899, df = 50, p-value = 7.851e-05

[1] "phone.number"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 5000, df = 4999, p-value = 0.4934

[1] "international.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 333.19, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 60.552, df = 1, p-value = 7.165e-15
```

In above we can say that independent variable “phone number” have p- value >0.05

We easily remove them from the dataset. So we remove three variables from dataset state, area code and phone number based on domain knowledge and visualization.

2.1.5 Feature Scaling:

This is very important technique in data preprocessing. It is very important in dataset in which when dealing with different parameters of different units and scales. It limit the range of the variables so they can compare on a common ground and it is performed on a continuous variables. In feature scaling we can perform two method. a. Normalization b. Standardization

a. Normalization: In this we normalize the dataset we impute the numeric dataset between 0 and 1.

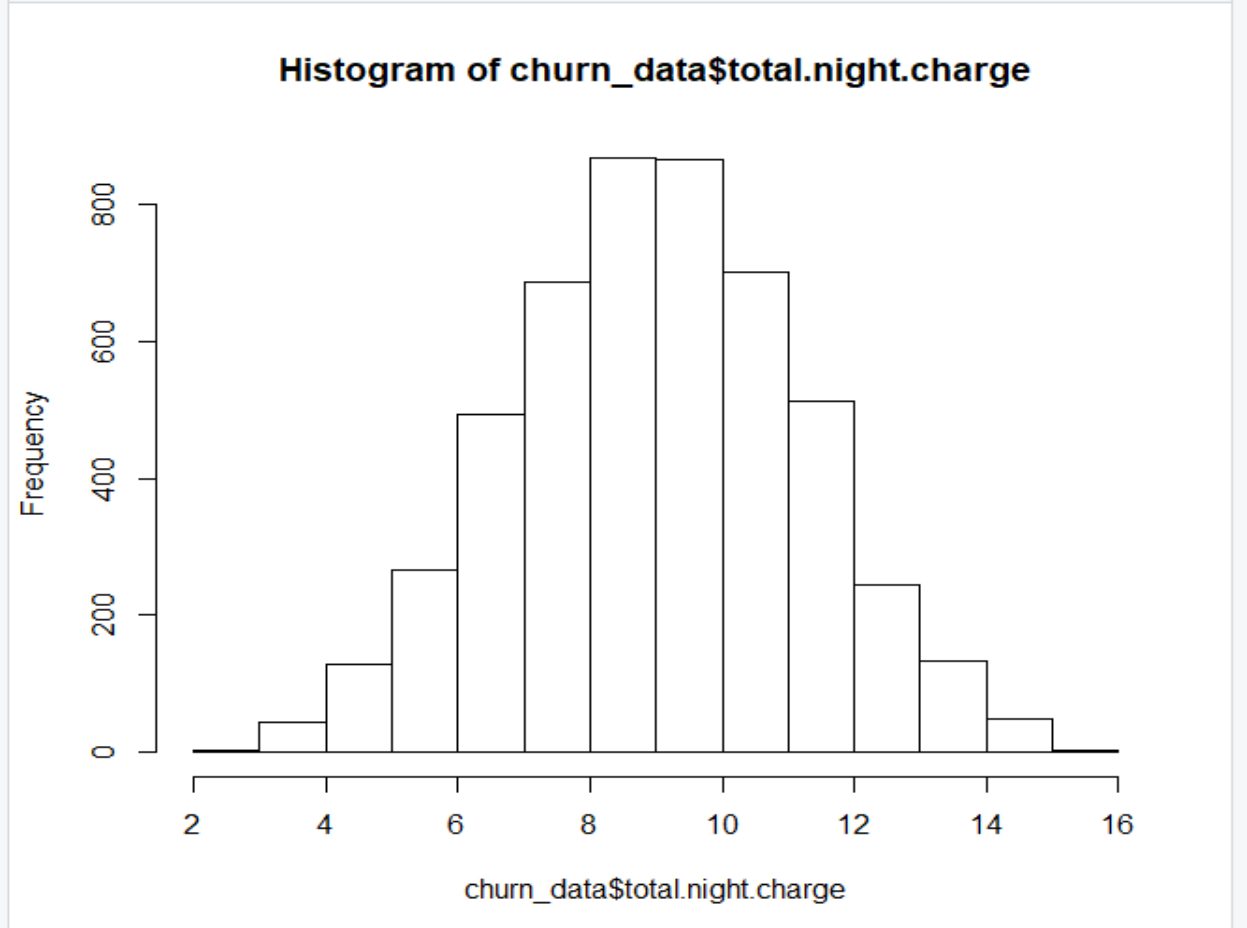
b. Standardization: standardization refers to the shifting the distribution of each attribute to have a mean of zero and a standard deviation of 1. It is done when the data is normally distributed we go for standardization.

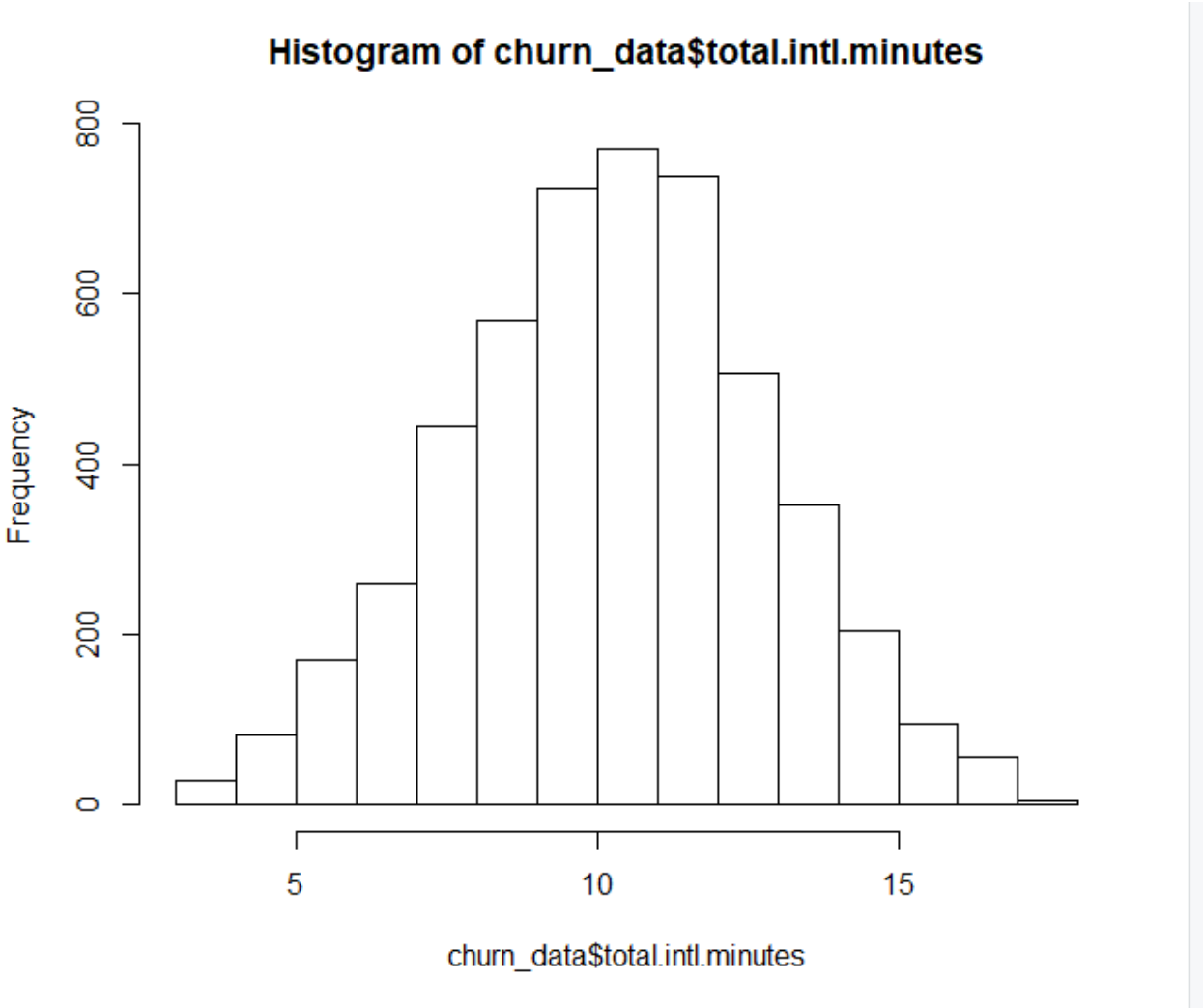
It is hard to know whether rescaling your data will improve the performance of your algorithms before you apply them. If often can, but not always.

So first we check the normality of variables by plotting distribution graph of the variables.

```
> qqnorm(churn_data$account.length)
>  hist(churn_data$account.length)
>  hist(churn_data$number.vmail.messages)
>  hist(churn_data$total.day.minutes)
>  hist(churn_data$total.day.calls)
>  hist(churn_data$total.day.charge)
>  hist(churn_data$total.eve.minutes)
>  hist(churn_data$total.eve.calls)
>  hist(churn_data$total.eve.charge)
>  hist(churn_data$total.night.minutes)
>  hist(churn_data$total.night.calls)
>  hist(churn_data$total.night.charge)
>  hist(churn_data$total.intl.minutes)
>  hist(churn_data$total.intl.calls)
>  hist(churn_data$total.intl.charge)
>  hist(churn_data$number.vmail.messages)
>
```

Output of some of these variables are given below





Hence from these figure we can depict the variables are normally distributed.

So instead of normalization we will go for standardization.

Code in R for normalization and standardization is given below.

```
#normalisation method
#for(i in cnames){
 #print(i)
  #churn_data[,i] = (churn_data[,i] - min(churn_data[,i]))/(max(churn_data[,i] - min(churn_data[,i])))
}

## as data is normally distributed so we will go for standardisation method

cnames = c("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")
#Standardisation
for(i in cnames){
  #print(i)
  churn_data[,i] = (churn_data[,i] - mean(churn_data[,i]))/
    sd(churn_data[,i])
}
head(churn_data)
```

Output of data after performing standardization

```
> head(churn_data)
  account.length international.plan voice.mail.plan number.vmail.messages total.day.minutes total.day.calls
1      -2.543301                  1               1             -0.5660574       -0.09796268      -1.3670946
2      -1.641223                  1               2              1.7609155       -0.65091869       1.4532915
3      -1.641223                  2               2              0.9076921       -0.16110298      -0.2180484
4      -1.512354                  1               1             -0.5660574       -0.40218415      -1.7849296
5      -1.486581                  1               1             -0.5660574       -0.17832307       1.5055209
6      -1.331939                  1               2              2.3038759        0.60040737       0.7743096
  total.day.charge total.eve.minutes total.eve.calls total.eve.charge total.night.minutes total.night.calls
1      -0.09847564       -1.00837208      -1.1004167      -1.00951712            0.618023          0.4794244
2      -0.65109771       -0.78621648      -1.0485945      -0.78687662           -1.462826          0.4794244
3      -0.16150382       -0.04775484       0.5578936      -0.04877497            2.493255         -1.2450297
4      -0.40236151       -1.53907710      -1.3077054      -1.53949831            1.263662         -1.2450297
5      -0.17838637       -0.34601929      -0.2194393      -0.34643564           -1.444321         -0.5134431
6       0.60046186       -0.83969839      -0.8413057      -0.84011674           -1.158512         -1.0360050
  total.night.charge total.intl.minutes total.intl.calls total.intl.charge number.customer.service.calls Churn
1          0.6168286        -1.96048247       -0.5996902       -1.96289487                    -0.3336128     1
2         -1.4621898         1.64955969        0.8600177        1.65624463                    -1.4040121     1
3          2.4947991         0.07997614       -1.5728288        0.08649737                    -0.3336128     2
4          1.2656651         0.31541367        0.3734484        0.31905252                    -0.3336128     1
5         -1.4439127        -2.23515959        0.8600177       -2.23905411                    -1.4040121     1
6         -1.1606179         0.74704915       -1.5728288        0.74055873                    -0.3336128     1
>
```

2.1.6 Feature Sampling:

In this we select the dataset from the whole dataset which have impact on the whole dataset to analyze the dataset we make a subset of the whole data which is called the data sampling. In our model we are using stratified sampling, it is based on stratum. Stratum is a subset of population that shares at least one common characteristic. In our data churn is 14.14% and not churn is 85% which lead to data imbalance issue not provide exact accuracy

```
##############################################Sampling#############################################
# ##Simple Random Sampling
data_sample = churn_data[sample(nrow(churn_data), 3000, replace = F), ]
#
# ##Stratified Sampling
 stratas = strata(churn_data, c("international.plan"), size = c(100, 199, 10, 5), method = "srswor")
 stratified_data = getdata(churn_data, stratas)
#
# ##Systematic sampling
# #Function to generate Kth index
   sys.sample = function(N,n)
 {
   k = ceiling(N/n)
   r = sample(1:k, 1)
   sys.samp = seq(r, r + k*(n-1), k)
  }
#
  lis = sys.sample(5000, 400) #select the repective rows
#
# #Create index variable in the data
    churn_data$index = 1:7414
#
# #Extract subset from whole data
  systematic_data = churn_data[which(churn_data$index %in% lis),]
```

2.1.7 Feature Splitting:

In this we divide the dataset in two parts like train and test dataset its ratio like 80:20 80% train and 20% test dataset. We build the model on training data set and apply same model on test dataset to check the performance of model. We calculate confusion metric parameters of each model in dataset. In this we use stratified sampling method to handle **data imbalance** issue ,it make sure that sample contains the same proportion of each class so that the sample becomes representative. Data imbalance issue refers to disparity encountered with dependent variable, here in this case 85% customers are not going to churn and left 14% is going to churn so when data exhibits an unequal distribution between its classes is considered to be imbalanced.

```
> ###################################Model Development#########################################
> #Clean the environment
> rmExcept("churn_data")
Removed the following objects:

cnames, data1, data2, df, factor_data, factor_index, gn1, gn10, gn11, gn12, gn13, gn14, gn15, gn16, gn2, gn3,
gn4, gn5, gn6, gn7, gn8, gn9, i, missing_val, numeric_data, numeric_index, val
>
> #Divide data into train and test using stratified sampling method
> #this code will divide the data into train and test set
> #we will be using 80% training and 20%testing split is there
> set.seed(1234)
> train.index = createDataPartition(churn_data$Churn, p = .80, list = FALSE)
> train = churn_data[ train.index,]
> test  = churn_data[-train.index,]
```

2.2 Modelling

2.2.1 Model Selection:

According to the dataset we select our model if we have categorical dependent variable like yes or no, We perform classification that dataset else if we have continuous dependent variable like in numeric we perform regression on that dataset. Here we have to predict value is yes or no so we have classification model on that we perform classification analysis. You always start your model building from the most simplest to more complex

2.2.2 Decision Tree Classification:

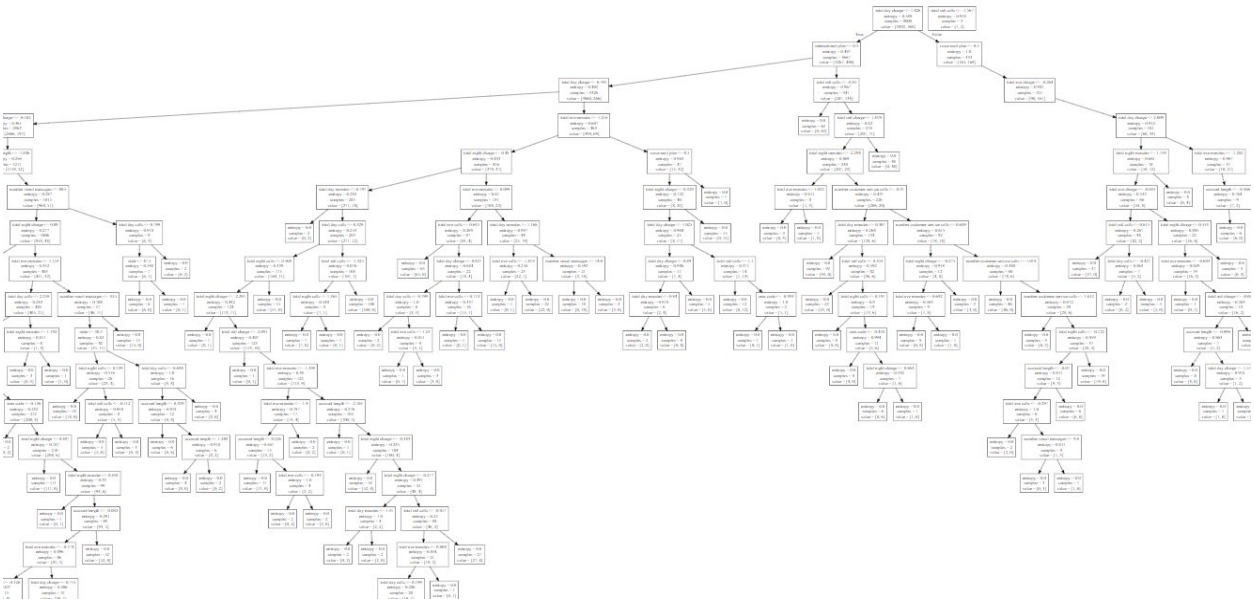
A predictive model based on a branching series of Boolean tests. It can be used for both classification and regression. There are no. of different types of decision trees that can be used in MLA. It is to create training model/ predictive model used for predict the class or value of target variable by the new distinct input rules from historical or past data.

This algorithm used tree like graph in which each leaf represent a class label and each node represent an attribute so basically decision tree is a rule in which each branch connects node with “and” and multiple branches are connected by “or”.

Here is the R code of decision tree classifier

```
#############################################################################################################
######  MODEL 1 DECISION TREE CLASSIFIER
#############################################################################################################
##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., train, trials = 80, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-18], type = "class")

##Evaluate the performance of classification model
ConfMatrix_C50 = table(test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#False Negative rate
FNR = 39*102(39+102)

#Accuracy: 95.2%
#FNR: 27.65%
```

The digraph generated of this dataset should be in this format to see it more clearly and precisely you need to go to http://webgraphviz.com/ and copy the rule from output and paste on its screen and click on generate digraph. In this way decision tree will be generated.



2.2.3 Random forest:

Another powerful MLA after decision tree is Random Forest. Idea behind it is to build 'n' no. of trees to have more accuracy in the dataset. Can be used for both classification and regression. It combines Breiman's "bagging" idea and the random selection of features.

Random Forest is an ensemble that consist of many decision trees algorithm. We built multiple decision trees to build accuracy and reduce misclassification error rate. It is a trial and error method. To build any decision tree in random forest we use Kart algorithm. RF will apply kart algorithm on training data, once RF built decision tree using Kart on training data, it will apply that training data on test data internally and check accuracy and false negative rate.

R code of random forest is

```
#############################################################################################
######  MODEL 2 RANDOM FOREST
#############################################################################################
RF_model = randomForest(Churn ~ ., train, importance = TRUE, ntree =33)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
#treeList = RF2List(RF_model)
#
# #Extract rules
# exec = extractRules(treeList, train[,-20])  # 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[,-20], train$Churn)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-18])

##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
#False Negative rate
FNR = 39*100/(39+100)
#Accuracy = 95.7%
#FNR = 27.6%
```

2.2.4 Logistic Regression:

It is a classification model which develop a model using regression coefficients. Input can be continuous or categorical. Possible outcomes could be class and probabilities. It predicts the probability of particular outcomes. Use logistic function to estimate the prediction. It follow same idea of development like linear regression. It is used client need output in terms of probabilities we can use this model.

R code of logistic regression is

```
#############################################################################################
######  MODEL 3 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 = test, type = "response")

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


##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, logit_Predictions)

##Accuracy=TP+TN/TP+TN+FP+FN
accuracy=(848+20)/(848+20+121+10)

#False Negative rate= FN/FN+TP
FNR = 121*100/(121+20)

#Accuracy = 86.8%
#FNR = 85.81%
```

2.2.5 KNN model:

KNN stands for K- nearest neighbor. It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measures. It predicts future test cases depends on nearest neighbor. Can be used for both classification and regression. It is also called as **lazy learning** because it never stores pattern from the training data. It is local heuristic. In classification it will apply majority and minority rule. In KNN classifier we select odd value of k to build the model

R code of KNN model

```
#################################################################################################
######  MODEL 4 KNN IMPLEMENTATION
#################################################################################################

#Predict test data
KNN_Predictions = knn(train[, 1:17], test[, 1:17], train$Churn, k = 17)

#Confusion matrix
Conf_matrix = table(KNN_Predictions, test$Churn)

#Accuracy
sum(diag(Conf_matrix))/nrow(test)

#False Negative rate= FN/FN+TP
FNR= 1*100/(25+1)

#Accuracy = 88.28%
#FNR = 17.39%
```

2.2.6 Naïve Bayes:

Classification algorithm based on Bayes i.e. called as probability. It is one of the most practical learning method. Works on probabilistic classification. It works on Bayes theorem of probability to predict class of unknown data set. Attribute values are conditionally independent give the target value.

Naïve Bayes assume that in a data set all variables or regressors are independent. It assume presence of particular feature is unrelated to present of other features. All probabilities should independently contribute to all probabilities. It is also proven in text mining.

It calculate probability of each target class and then on basis of MAPE rule it will assign target value to test case depend on probabilities.

R code for Naïve Bayes

```
#################################################################################################
######  MODEL 5 NAIVE BAYES
#################################################################################################

#Develop model
NB_model = naiveBayes(Churn ~ ., data = train)

#predict on test cases #raw
NB_Predictions = predict(NB_model, test[,1:17], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = test[,18], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)

#FNR = FN*100/(FN+TP)
FNR= 87*100/(87+54)

#Accuracy: 88.89%
#FNR: 61.70%
```

Chapter 3 Conclusion

3.1 Model Evaluation with help of confusion matrix

Confusion matrix is a summary of prediction results on a classification problem.
The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.
The confusion matrix shows the ways in which your classification model is confused when it makes predictions.It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

3.1.1 Decision tree

```
#############################################################################################
######  MODEL 1 DECISION TREE CLASSIFIER
#############################################################################################
##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., train, trials = 80, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-18], type = "class")

##Evaluate the performance of classification model
ConfMatrix_C50 = table(test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#False Negative rate
FNR = 39*102(39+102)

#Accuracy: 95.2%
#FNR: 27.65%
```

Output

```
  72         46  404(10.1%)
  73         14  266( 6.6%)
  74         42  388( 9.7%)
  75         26  506(12.6%)
  76         29  214( 5.3%)
  77         33  270( 6.7%)
  78         24  238( 5.9%)
  79         36  369( 9.2%)
boost             79( 2.0%)   <<


           (a)   (b)    <-classified as
          ----  ----
          3435          (a): class 1
            79   487    (b): class 2


        Attribute usage:

        100.00% account.length
        100.00% international.plan
        100.00% voice.mail.plan
        100.00% number.vmail.messages
        100.00% total.day.minutes
        100.00% total.day.calls
        100.00% total.day.charge
        100.00% total.eve.minutes
        100.00% total.eve.calls
        100.00% total.eve.charge
        100.00% total.night.calls
        100.00% total.night.charge
        100.00% total.intl.minutes
        100.00% total.intl.calls
        100.00% number.customer.service.calls
         99.43% total.night.minutes
         91.05% total.intl.charge


> #write rules into disk
> write(capture.output(summary(C50_model)), "c50Rules.txt")
>
> #Lets predict for test cases
> C50_Predictions = predict(C50_model, test[,-18], type = "class")
>
> ##Evaluate the performance of classification model
> ConfMatrix_C50 = table(test$Churn, C50_Predictions)
> confusionMatrix(ConfMatrix_C50)
Confusion Matrix and Statistics

   C50_Predictions
      1   2
  1 849   9
  2  39 102

               Accuracy : 0.952
                 95% CI : (0.9368, 0.9644)
    No Information Rate : 0.8889
    P-Value [Acc > NIR] : 1.437e-12

                  Kappa : 0.7825
 Mcnemar's Test P-Value : 2.842e-05

            Sensitivity : 0.9561
            Specificity : 0.9189
         Pos Pred Value : 0.9895
         Neg Pred Value : 0.7234
             Prevalence : 0.8889
         Detection Rate : 0.8498
   Detection Prevalence : 0.8589
      Balanced Accuracy : 0.9375

       'Positive' Class : 1

>
```

3.1.2 RANDOM FOREST

```
#############################################################################################
######  MODEL 2 RANDOM FOREST
#############################################################################################
RF_model = randomForest(Churn ~ ., train, importance = TRUE, ntree =33)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
#treeList = RF2List(RF_model)
#
# #Extract rules
# exec = extractRules(treeList, train[,-20])  # 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[,-20], train$Churn)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-18])

##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
#False Negative rate
FNR = 39*100/(39+100)
#Accuracy = 95.7%
#FNR = 27.6%
```

```
> #
> # #evaulate few rules
> # ruleMetric[1:2,]
>
> #Presdict test data using random forest model
> RF_Predictions = predict(RF_model, test[,-18])
>
> ##Evaluate the performance of classification model
> ConfMatrix_RF = table(test$Churn, RF_Predictions)
> confusionMatrix(ConfMatrix_RF)
Confusion Matrix and Statistics

    RF_Predictions
       1   2
  1 854   4
  2  39 102

                Accuracy : 0.957
                  95% CI : (0.9425, 0.9687)
     No Information Rate : 0.8939
     P-Value [Acc > NIR] : 3.475e-13

                   Kappa : 0.8019
 Mcnemar's Test P-Value : 2.161e-07

             Sensitivity : 0.9563
             Specificity : 0.9623
          Pos Pred Value : 0.9953
          Neg Pred Value : 0.7234
              Prevalence : 0.8939
          Detection Rate : 0.8549
    Detection Prevalence : 0.8589
       Balanced Accuracy : 0.9593

        'Positive' Class : 1

>
```

3.1.3 LOGISTIC REGRESSION

```
#############################################################################################################
######  MODEL 3 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 = test, type = "response")

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


##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, logit_Predictions)

##Accuracy=TP+TN/TP+TN+FP+FN
accuracy=(848+20)/(848+20+121+10)

#False Negative rate= FN/FN+TP
FNR = 121*100/(121+20)

#Accuracy = 86.8%
#FNR = 85.81%
```

```
>
> #predict using logistic regression
> logit_Predictions = predict(logit_model, newdata = test, type = "response")
>
> #convert prob
> logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)
>
>
> ##Evaluate the performance of classification model
> ConfMatrix_RF = table(test$Churn, logit_Predictions)
> ConfMatrix_RF
   logit_Predictions
      0   1
  1 848  10
  2 121  20
>
```

3.1.4 KNN ALGORITHM

```
#############################################################################################################
######  MODEL 4 KNN IMPLEMENTATION
#############################################################################################################

#Predict test data
KNN_Predictions = knn(train[, 1:17], test[, 1:17], train$Churn, k = 17)

#Confusion matrix
Conf_matrix = table(KNN_Predictions, test$Churn)

#Accuracy
sum(diag(Conf_matrix))/nrow(test)

#False Negative rate= FN/FN+TP
FNR= 1*100/(25+1)

#Accuracy = 88.28%
#FNR = 17.39%
```

```
> #Predict test data
> KNN_Predictions = knn(train[, 1:17], test[, 1:17], train$Churn, k = 17)
>
> #Confusion matrix
> Conf_matrix = table(KNN_Predictions, test$Churn)
> confusionMatrix(Conf_matrix)
Confusion Matrix and Statistics


KNN_Predictions   1   2
              1 857 116
              2   1  25

               Accuracy : 0.8829
                 95% CI : (0.8613, 0.9022)
    No Information Rate : 0.8589
    P-Value [Acc > NIR] : 0.01471

                  Kappa : 0.2672
 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.9988
            Specificity : 0.1773
         Pos Pred Value : 0.8808
         Neg Pred Value : 0.9615
             Prevalence : 0.8589
         Detection Rate : 0.8579
   Detection Prevalence : 0.9740
      Balanced Accuracy : 0.5881

       'Positive' Class : 1

>
> #Accuracy
> sum(diag(Conf_matrix))/nrow(test)
[1] 0.8828829
>
```

3.1.5 NAÏVE BAYES

```
#############################################################################################
######  MODEL 5 NAIVE BAYES
#############################################################################################

#Develop model
NB_model = naiveBayes(Churn ~ ., data = train)

#predict on test cases #raw
NB_Predictions = predict(NB_model, test[,1:17], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = test[,18], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)

#FNR = FN*100/(FN+TP)
FNR= 87*100/(87+54)

#Accuracy: 88.89%
#FNR: 61.70%
```

3.2 Model Selected

Hence from all the above models we save **random forest** as a best selection because of its high accuracy and less false negative rate. This algorithm can handle high dimensional spaces as well as large number of training examples. Although knn is also good but for this algorithm, we have to determine the value of parameter K (number of nearest neighbors) and the type of distance to be used. The computation time is also very much as we need to compute distance of each query instance to all training samples.

Complete R code file

```
##############################.....CHURN PREDICTION MODEL.....######################

#remove all objects stored
rm(list=ls(all=T))

#set current working directory
setwd("C:/Users/admin/akansha")
getwd()

#load libraries for data preprocessing and visualisations
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071",
      "Information","MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','class')
#library(ggplot2)          for boxplot visualisations
#library(corrgram)         for correlation plot
#library(DMwR)             for knn imputation
#library(caret)            for partitioning data into train and test
#library(randomForest)     for implementing random forest model
#library(unbalanced)       for rebalancing the dataset
#library(C50)              for implementing decision tree
#library(dummies)          for performing confusion metrics
#library(e1071)            for implementing naive bayes
#library(Information)      for implementing naives and logistic regression
#library(MASS)             for performing chi-square test
#library(rpart)            for implementing cart algorithm
#library(gbm)              for implementing gradient boosting algorithm
#library(ROSE)             Random over sampling in binary classification
#library(sampling)         for performing sampling
#library(DataCombine)      for combining and cleansing dataset
#library(inTrees)          interpreting tree ensembles extracting,summarising rules
#library(class)            for knn implementattion


#install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)
```

```
#install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)
#read data
data1=read.csv("Test_data.csv",header=T,na.strings=c("","","NA"))
data2=read.csv("Train_data.csv",header=T,na.strings=c("","","NA"))

#merging two dataset
churn_data=merge(data1,data2,all.x = TRUE, all.y = TRUE)

###############exploring the data #############
view(churn_data)
head(churn_data)
str(churn_data)

############################################## Explore data analysis ################################

## Univariate Analysis and Variable Consolidation
##Data Manupulation; convert string categories into factor numeric
for(i in 1:ncol(churn_data)){
  if(class(churn_data[,i]) == 'factor'){
    churn_data[,i] = factor(churn_data[,i],
                            labels=(1:length(levels(factor(churn_data[,i])))))
  }
}

#check for missing values in dataset
sapply(churn_data, function(x) sum(is.na(x)))


#################################### outlier analysis ##################################3
# ## BoxPlots - Distribution and Outlier Check
numeric_index = sapply(churn_data,is.numeric) #selecting only numeric

numeric_data = churn_data[,numeric_index]
```

```
numeric_data = churn_data[,numeric_index]
cnames=colnames(numeric_data)
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn",group=1), data = subset(churn_data))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames[i],x="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,ncol=2)
gridExtra::grid.arrange(gn15,gn16,ncol=2)

### Remove outliers using boxplot method

df=churn_data  #saving raw data for experiment just to avoid complexity

# # #loop to remove from all variables
#for(i in cnames){
  #print(i)
  #val = churn_data[,i][churn_data[,i] %in% boxplot.stats(churn_data[,i])$out]
  #print(length(val))
  #churn_data = churn_data[which(!churn_data[,i] %in% val),]
#}

#Replace all outliers with NA and impute as missing value % was more than 30%
for(i in cnames){
  val = churn_data[,i][churn_data[,i] %in% boxplot.stats(churn_data[,i])$out]
  #print(length(val))
```

```
  #print(length(val))
  churn_data[,i][churn_data[,i] %in% val] = NA
}

missing_val = data.frame(apply(churn_data,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] =  "Missing_percentage"
missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(churn_data)) * 100
missing_val = missing_val[order(-missing_val$Missing_percentage),]
row.names(missing_val) = NULL
missing_val = missing_val[,c(2,1)]
write.csv(missing_val, "Miising_perc.csv", row.names = F)
sum(missing_val$Missing_percentage)

# imputing missing values using knn method
#churn_data=knnImputation(churn_data,k=3)
#Mean Method
#Mean Method
churn_data$account.length[is.na(churn_data$account.length)] = mean(churn_data$account.length, na.rm = T)
churn_data$area.code[is.na(churn_data$area.code)] = mean(churn_data$area.code, na.rm = T)
churn_data$number.vmail.messages[is.na(churn_data$number.vmail.messages)] = mean(churn_data$number.vmail.messages, na.rm = T)
churn_data$total.day.minutes[is.na(churn_data$total.day.minutes)] = mean(churn_data$total.day.minutes, na.rm = T)
churn_data$total.day.calls[is.na(churn_data$total.day.calls)] = mean(churn_data$total.day.calls, na.rm = T)
churn_data$total.day.charge[is.na(churn_data$total.day.charge)] = mean(churn_data$total.day.charge, na.rm = T)
churn_data$total.eve.minutes[is.na(churn_data$total.eve.minutes)] = mean(churn_data$total.eve.minutes, na.rm = T)
churn_data$total.eve.calls[is.na(churn_data$total.eve.calls)] = mean(churn_data$total.eve.calls, na.rm = T)
churn_data$total.eve.charge[is.na(churn_data$total.eve.charge)] = mean(churn_data$total.eve.charge, na.rm = T)
churn_data$total.night.minutes[is.na(churn_data$total.night.minutes)] = mean(churn_data$total.night.minutes, na.rm = T)
churn_data$total.night.calls[is.na(churn_data$total.night.calls)] = mean(churn_data$total.night.calls, na.rm = T)
churn_data$total.night.charge[is.na(churn_data$total.night.charge)] = mean(churn_data$total.night.charge, na.rm = T)
churn_data$total.intl.minutes[is.na(churn_data$total.intl.minutes)] = mean(churn_data$total.intl.minutes, na.rm = T)
churn_data$total.intl.calls[is.na(churn_data$total.intl.calls)] = mean(churn_data$total.intl.calls, na.rm = T)
churn_data$total.intl.charge[is.na(churn_data$total.intl.charge)] = mean(churn_data$total.intl.charge, na.rm = T)
churn_data$number.customer.service.calls[is.na(churn_data$number.customer.service.calls)] = mean(churn_data$number.customer.service.calls,
```

```
############################### Feature Selection ###############################################
## Correlation Plot
corrgram(churn_data[,numeric_index], order = F,
         upper.panel=panel.pie,text.panel=panel.txt, main = "Correlation Plot")

## Chi-squared Test of Independence
factor_index = sapply(churn_data,is.factor)
factor_data = churn_data[,factor_index]
for (i in 1:4)
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn,factor_data[,i])),simulate.p.value = TRUE)
}

############################ Dimension Reduction ########################################
churn_data= subset(churn_data,
                   select = -c(phone.number,state,area.code))

################################Feature Scaling#############################################
#Normality check
qqnorm(churn_data$account.length)
hist(churn_data$account.length)
hist(churn_data$number.vmail.messages)
hist(churn_data$total.day.minutes)
hist(churn_data$total.day.calls)
hist(churn_data$total.day.charge)
hist(churn_data$total.eve.minutes)
hist(churn_data$total.eve.calls)
hist(churn_data$total.eve.charge)
hist(churn_data$total.night.minutes)
hist(churn_data$total.night.calls)
hist(churn_data$total.night.charge)
hist(churn_data$total.intl.minutes)
hist(churn_data$total.intl.calls)
hist(churn_data$total.intl.charge)
```

```
#for(i in cnames){
 #print(i)
  #churn_data[,i] = (churn_data[,i] - min(churn_data[,i]))/(max(churn_data[,i] - min(churn_data[,i])))
#}

## as data is normally distributed so we will go for standardisation method

cnames = c("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")
#Standardisation
for(i in cnames){
  #print(i)
  churn_data[,i] = (churn_data[,i] - mean(churn_data[,i]))/
    sd(churn_data[,i])
}
head(churn_data)

########################################Sampling########################################
# ##Simple Random Sampling
#data_sample = churn_data[sample(nrow(churn_data), 3000, replace = F), ]
#
# ##Stratified Sampling
#stratas = strata(churn_data, c("international.plan"), size = c(100, 199, 10, 5), method = "srswor")
#stratified_data = getdata(churn_data, stratas)
#
# ##Systematic sampling
# #Function to generate Kth index
#sys.sample = function(N,n)
#{
  #k = ceiling(N/n)
  #r = sample(1:k, 1)
```

```
#################################################################################################
######  MODEL 1 DECISION TREE CLASSIFIER
#################################################################################################
##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., train, trials = 80, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-18], type = "class")

##Evaluate the performance of classification model
ConfMatrix_C50 = table(test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#False Negative rate
FNR = 39*102(39+102)

#Accuracy: 95.2%
#FNR: 27.65%
```

```
#################################################################################################
######  MODEL 2 RANDOM FOREST
#################################################################################################
RF_model = randomForest(Churn ~ ., train, importance = TRUE, ntree =33)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
#treeList = RF2List(RF_model)
#
# #Extract rules
# exec = extractRules(treeList, train[,-20])  # 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[,-20], train$Churn)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-18])

##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
#False Negative rate
FNR = 39*100/(39+100)
#Accuracy = 95.7%
#FNR = 27.6%
```

```
#############################################################################################
######  MODEL 3 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 = test, type = "response")

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


##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, logit_Predictions)

##Accuracy=TP+TN/TP+TN+FP+FN
accuracy=(848+20)/(848+20+121+10)

#False Negative rate= FN/FN+TP
FNR = 121*100/(121+20)

#Accuracy = 86.8%
#FNR = 85.81%


#############################################################################################
######  MODEL 4 KNN IMPLEMENTATION
#############################################################################################

#Predict test data
KNN_Predictions = knn(train[, 1:17], test[, 1:17], train$Churn, k = 17)

#Confusion matrix
Conf_matrix = table(KNN_Predictions, test$Churn)

#Accuracy
sum(diag(Conf_matrix))/nrow(test)

#False Negative rate= FN/FN+TP
FNR= 1*100/(25+1)

#Accuracy = 88.28%
#FNR = 17.39%


#############################################################################################
######  MODEL 5 NAIVE BAYES
#############################################################################################

#Develop model
NB_model = naiveBayes(Churn ~ ., data = train)

#predict on test cases #raw
NB_Predictions = predict(NB_model, test[,1:17], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = test[,18], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)

#FNR = FN*100/(FN+TP)
FNR= 87*100/(87+54)

#Accuracy: 88.89%
#FNR: 61.70%
```

```
#########################################################################################################################
######### Save the model to a file
#########################################################################################################################
save(RF_model,file="churnmodel.rda")


# the random forest model accuarcy is very high and also false negative rate is very low so this can handle high dimensional spaces
# and also large number of training examples.
# as compare to knn model it is also good but its computing time is very much and we have to determine the value of parameter k
# where k is number of nearest neighbors and the type of distance to be used
```

In this way we save random forest as best model that can easily work on test data set

We trained model on certain set of information, We test the algorithm and derived insights from it .Now we can do deployment of it . Specifically in big data environment we save the model so we can apply it on the future test cases and can do churn attrition of customers as Churn is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving.