NAME : KRISHNA KUMAR

PROJECT NAME : CHURN REDUCTION

DATE OF SUBMISSION : 20 – 09 - 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

Churn (loss of customers to competition) is a problem for telecom companies or any other company because it is more expensive to acquire a new customer than to keep your existing one from leaving. The aim of this project is to predict customer behavior. That is predicting whether a particular customer will churn or not .The main objective or the expected outcome of the project is churn reduction .

1.2 Why does Churn occur

Churn may occur due to various reasons. It could be due to lower tariffs of a rival company , dissatisfaction of the customer and much more . It is true that it is more expensive to acquire a new customer than to retain your existing customer. So to avoid churn we must first find the reason for churn . It is possible to retain customers if proper actions are taken . Most of the customers who are going to churn can be prevented if actions are taken .

Besides , Churn also affects the growth of the company. Churn is inversely proportional to the growth of the company . One research shows that churn could range anywhere between 10 percent at the lowest to as much as 60 percent highest . This can drastically affect the profit and growth of the company as well as the reputation of the company.

1.3 Data

Let's take a look at our dataset. Our dataset consists of 3333 observations and 21 variables in the train dataset . In test dataset it contains 1667 observations and 21 variables. Of these variables, 20 are independent variables and 1 is our dependent variable, that is target variable . So we take a look at the sample data of our original dataset.

Important Note

Please Note : For convenience and ease of operation the two data sets that is provided as part of the project that is Test_data.csv and Train_data.csv has been merged into a single dataset (row binded). This is done only for the sake of avoiding complexities and confusion and also to make easier the pre processing steps. Then later after the pre – processing is done , that is outlier analysis , feature selection same rows of the Test_data.csv and Train_data.csv has been manually selected from the merged dataset and split into the same train and test for feeding into the model . This can be done since we row binded . Then the modeling is done as usual .

Table 1.1 Churn sample data (Columns 1-7) - Predictor Variables

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages
KS	128	415	382-4657	no	yes	25
OH	107	415	371-7191	no	yes	26
NJ	137	415	358-1921	no	no	0
OH	84	408	375-9999	yes	no	0
OK	75	415	330-6626	yes	no	0
AL	118	510	391-8027	yes	no	0
MA	121	510	355-9993	no	yes	24

Table 1.2 Churn sample data (Columns 8-14) - Predictor Variables

total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes
265.1	110	45.07	197.4	99	16.78	244.7
161.6	123	27.47	195.5	103	16.62	254.4
243.4	114	41.38	121.2	110	10.3	162.6
299.4	71	50.9	61.9	88	5.26	196.9
166.7	113	28.34	148.3	122	12.61	186.9
223.4	98	37.98	220.6	101	18.75	203.9
218.2	88	37.09	348.5	108	29.62	212.6

Table 1.3 Churn sample data (Columns 15-21)

Column 21 – Dependent variable (target)

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

From the above sample data we can have an idea about the dataset. As the dataset is related to telecom, in this dataset we can mostly see the data related to number of minutes , number of calls , charge , the area they belong to and much more .

We can also note that this dataset does not consist of any missing values. Also the dataset we are given consists of two files that is the train and test dataset.

Of the 20 predictor variables (independent variables), there is a combination of both continuous variables and categorical variables. The continuous and categorical variables are listed below.

Continuous variables:

- Account length
- Number of voicemail messages
- total day minutes used
- total day calls
- 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

Categorical variables :

- State
- international plan
- voicemail plan
- phone number
- area code

Target variable (Dependent variable) :

- **Churn (True , False)**

We had a basic look at the data so let us proceed to the next step.

Chapter 2

Methodology

2.1 Pre Processing

Before proceeding to process the data there are a few steps left. We have to first clean the data. This is because the data may contain missing values , outliers , highly correlated variables etc . Here we use both numerical and graphical techniques to:

- Visualize the data
- Detect the missing values
- Detect outliers
- Remove outliers
- Find interesting patterns in the data
- Finding the trends
- Relationship between the variables
- Feature selection / Feature engineering
- Dimensionality reduction

In modeling or machine learning there is something called **Garbage In - Garbage Out (GIGO)** . What this means is if there is impurity or noise in the data then our model will have a very worst accuracy . Any garbage that goes in will come out as garbage .

2.1.1 Missing value analysis

One of the very first steps of data pre – processing is missing value analysis . Even though some models can deal with missing values it is important to properly deal with missing values in our dataset as it may affect the accuracy of the model . Like we have seen previously that our data **does not contain any missing values** . So we can proceed further .

2.1.2 Data types

So after performing missing value analysis we have to make sure that the data types are in correct format. That is continuous variables should have numeric values . Categorical variables can be label encoded to have levels .

2.1.3 Outlier analysis

Next thing in our process is Outlier analysis. In statistics, an outlier is an observation point that is distant from other observations. An outlier can cause serious problems in statistical analyses. Outliers can affect the accuracy of the model and the outcome. Although some models are resistant to outliers it is a good practice to deal with outliers beforehand. Here we have used the boxplot method to detect the outliers .

Fig 1.1 R Boxplot - Account length , number of voicemail messages, total day minutes

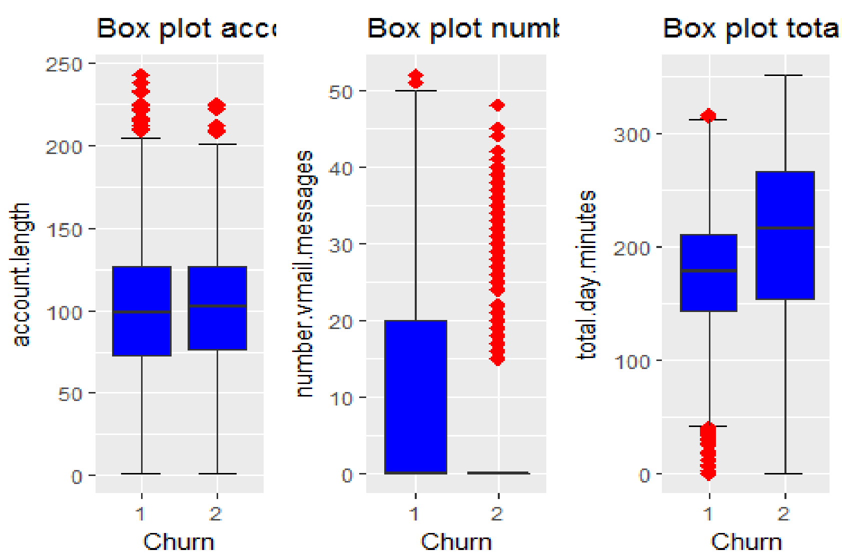


Fig 1.2 R Boxplot – total day calls , total day charge , total day minutes

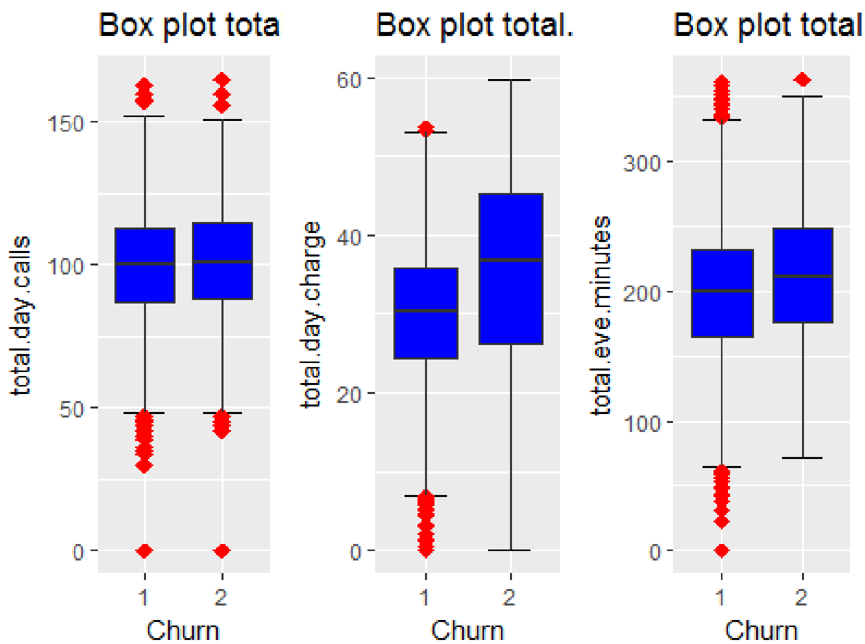


Fig 1.3 R Boxplot – total eve calls , total eve charge , total night minutes

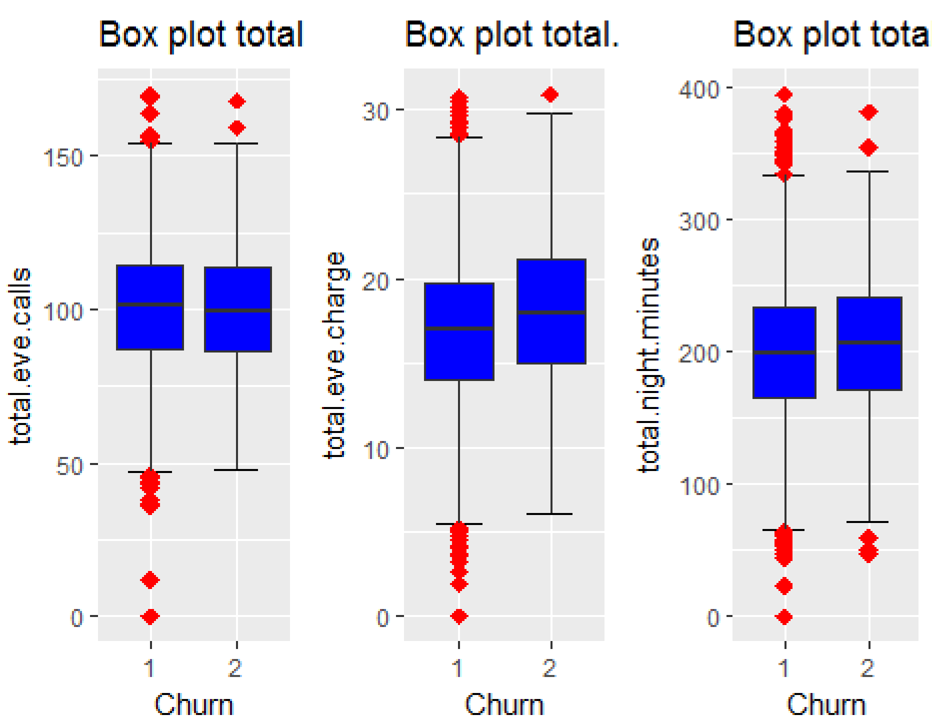


Fig 1.4 R – Boxplot – total night calls , total night charge , total international minutes

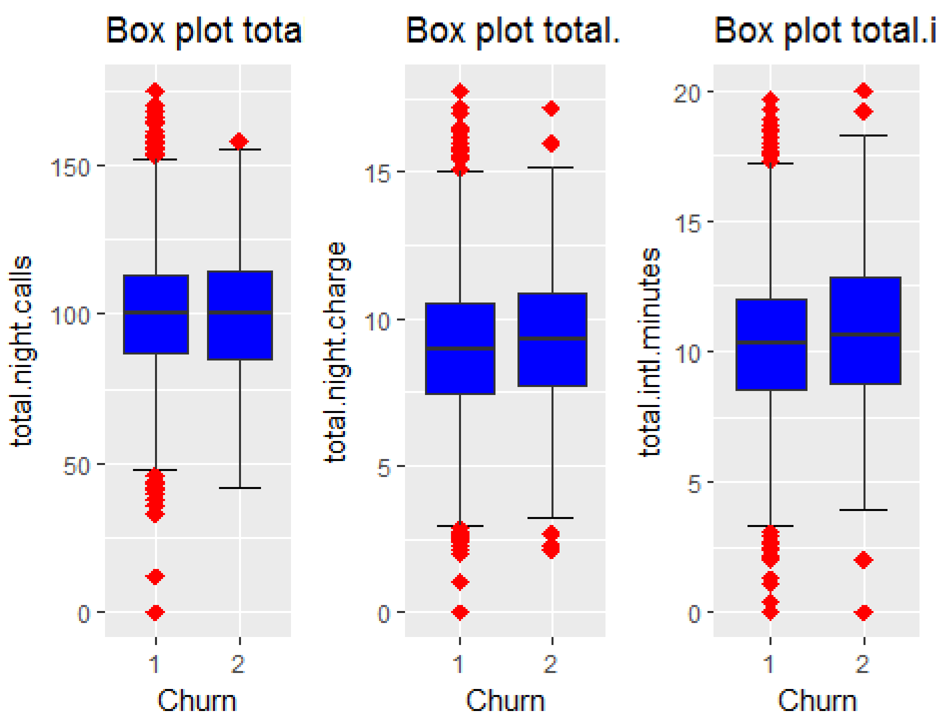
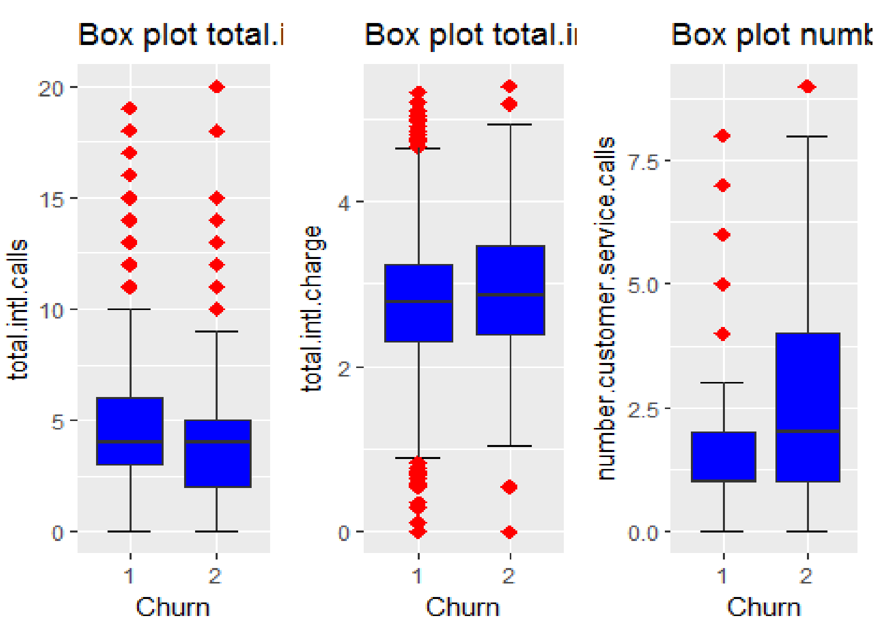


Fig 1.5 R Boxplot – total intl calls , total intl charge , number of customer service calls



In the above figures **" red dots "** indicate the outliers . Outliers are those extreme values present in our data . They are represented by red dots in the above figures. As we can see, that majority of the variables consist of outliers. This is because of large number of continuous variables .

So before proceeding it is necessary to remove the outliers . The code for removing the outliers in R and python can be found in the code file itself. Below are the figure of boxplot after removing the outliers .

Fig 1.6 Boxplot after outlier removal

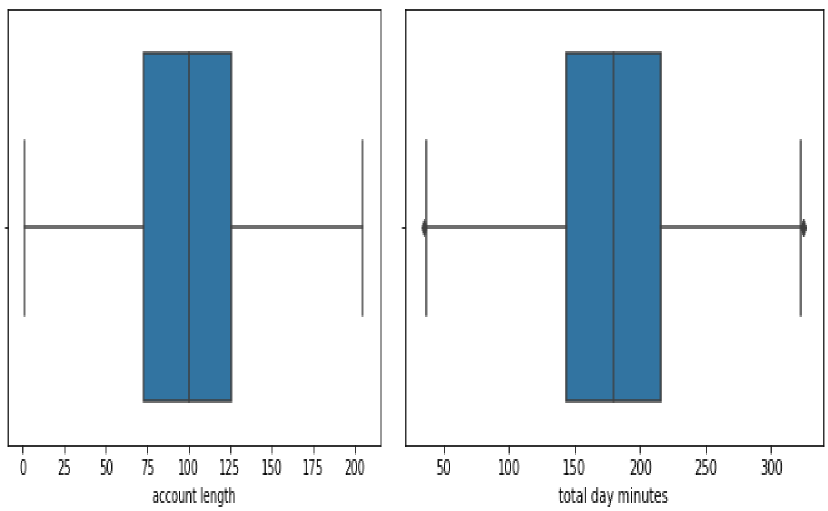


Fig 1.7 Boxplot after outlier removal

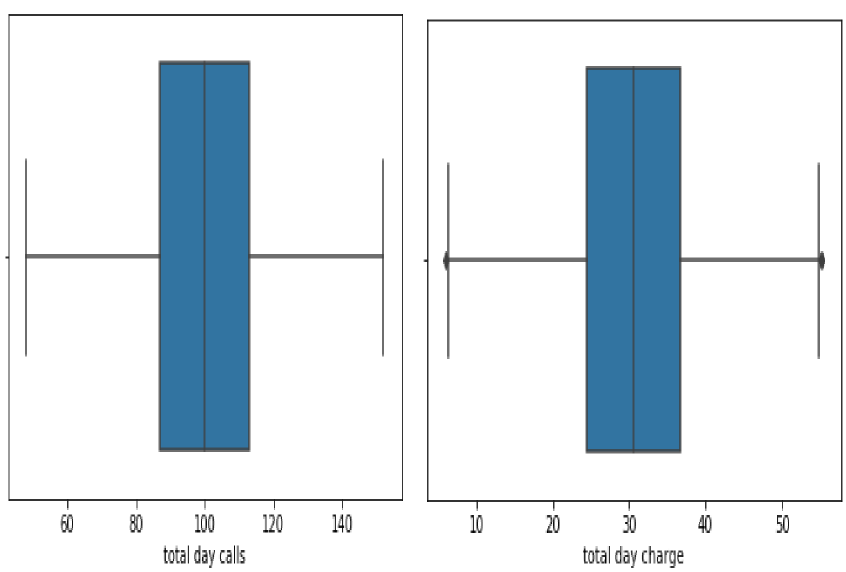


Fig 1.8 Boxplot after outlier removal

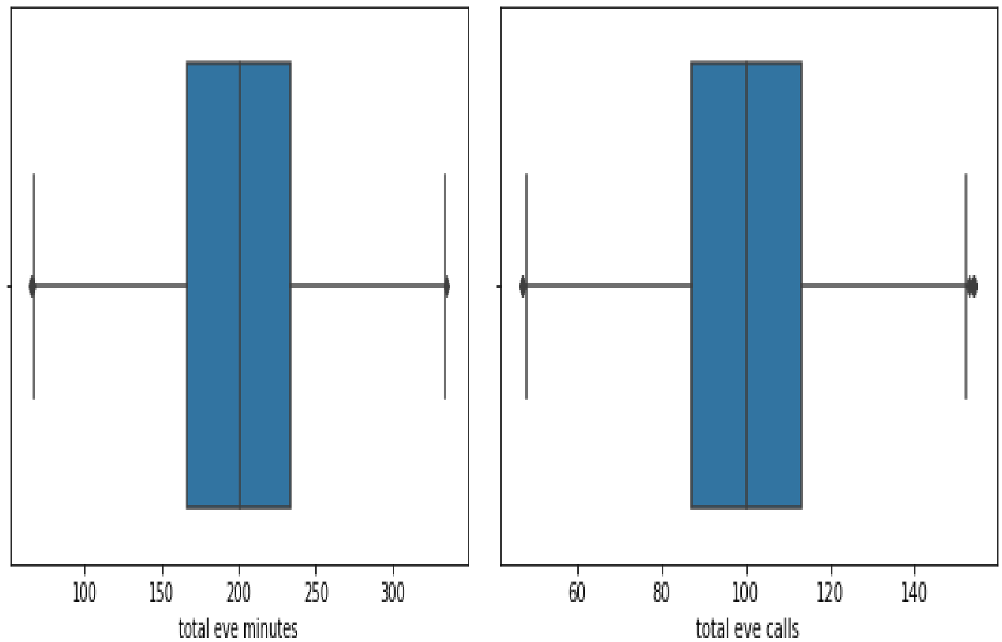


Fig 1.9 Boxplot after outlier removal

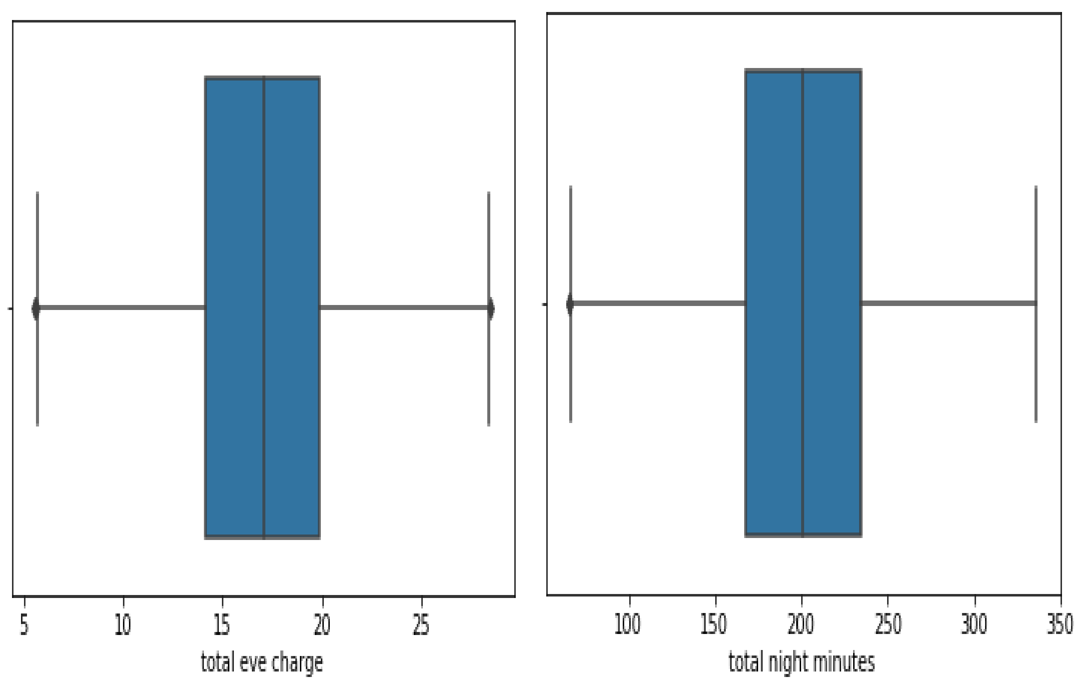


Fig 2.0 Boxplot after outlier removal

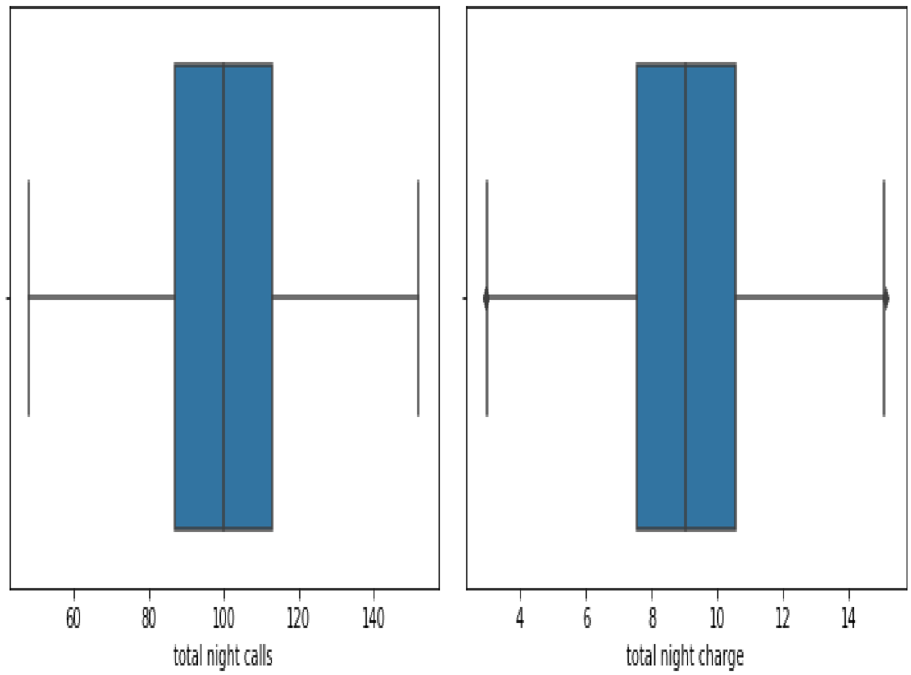


Fig 2.1 Boxplot after outlier removal

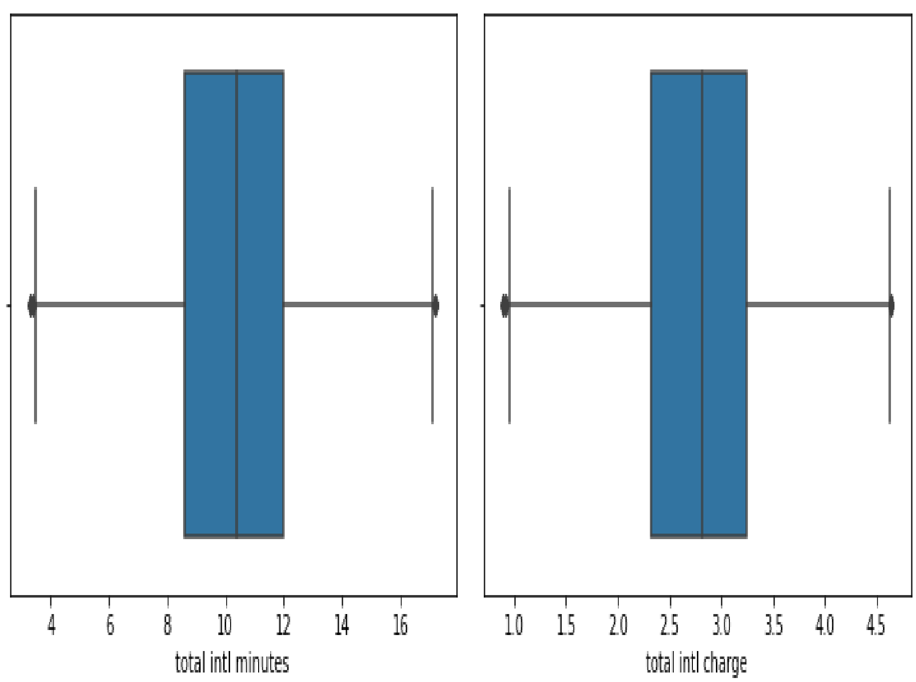
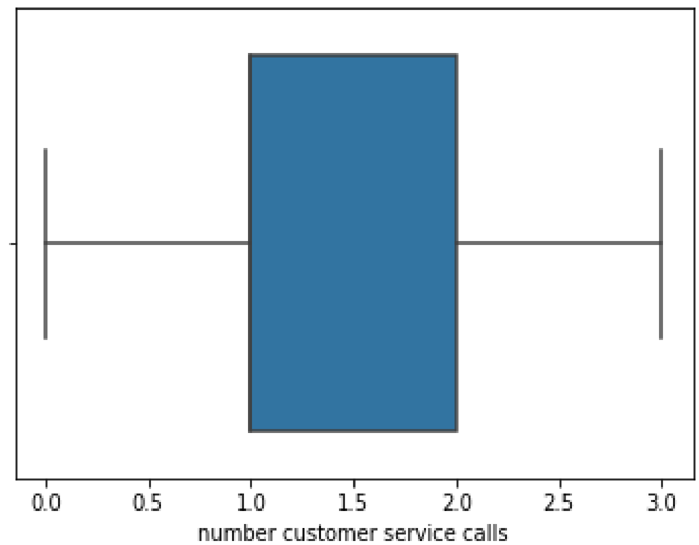


Fig 2.1 Boxplot after outlier removal



As we can see that there no outliers in our data now.

2.1.4 Feature Selection

Feature selection is the next step of our project . It is also called Dimensionality reduction . This is important because there may be many highly correlated independent variables in our dataset . This is called multicollinearity. If not taken care of it may affect the model . Also many variables may not be necessary for the model .

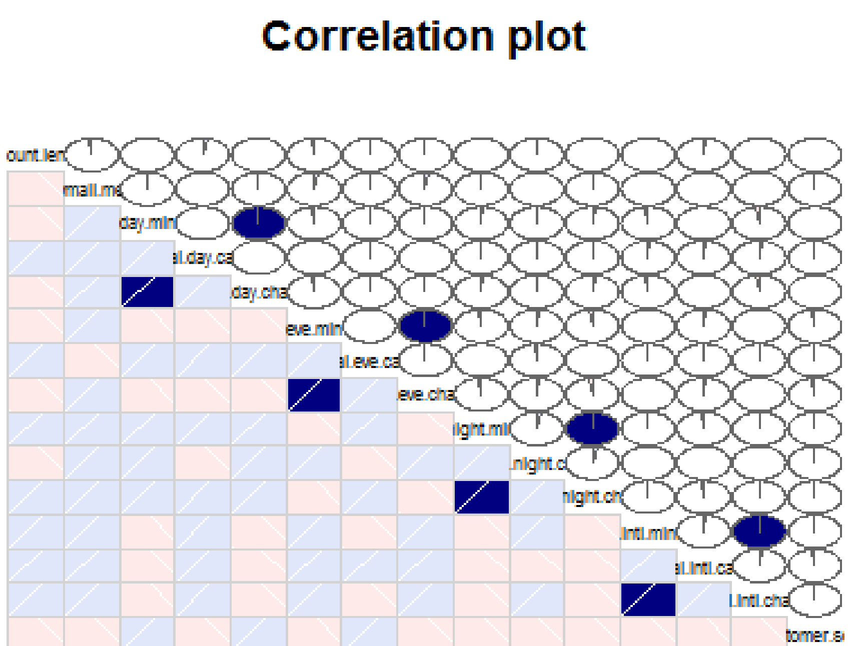
There are many techniques for feature selection .In the basic level we can perform correlation analysis for continuous variables to detect the highly correlated variables.

Similarly chi square test can be performed on categorical variables to know which variables have high prediction power and we can ignore the rest of the variables.

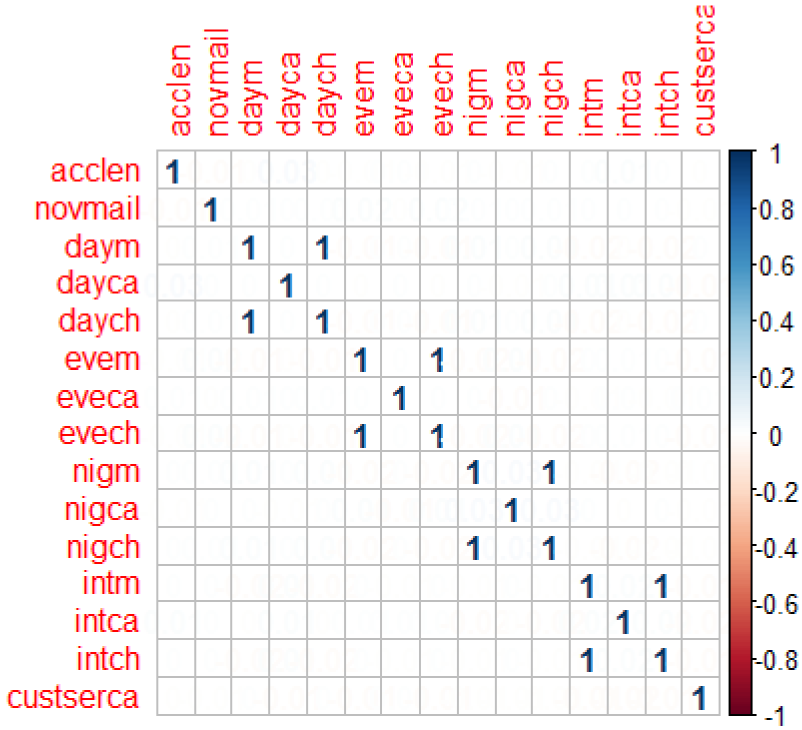
But if there are large number of variables present in our dataset then it can be difficult to interpret correlation analysis and chi square test. So in that case we can go for **Principal Component Analysis (PCA)**. It helps to reduce large number of variables. Then on the remaining variables we can perform correlation analysis .

```
> corrgram(Z_1 , order = F , upper.panel = panel.pie , text.panel = panel.txt
,
+         main = "Correlation plot")
```

Fig 2.2 Correlation plot




```
> corrplot(X2 ,   method="number" )
```




```
> symnum(cor(Z_2))
          a nv dym dayc dych evm evec evch ngm nigc ngch intm intca intch c
acclen    1
novmail     1
daym           1
dayca              1
daych          1        1
evem                         1
eveca                            1
evech                        1        1
nigm                                       1
nigca                                          1
nigch                                      1        1
intm                                                     1
intca                                                         1
intch                                                    1          1
custserca                                                                 1
attr(,"legend")
[1] 0 ‘ ’ 0.3 ‘.’ 0.6 ‘,’ 0.8 ‘+’ 0.9 ‘*’ 0.95 ‘B’ 1
```

```
> vif(Z_1[,1:15])
                    Variables          VIF
1              account.length 1.001602e+00
2         number.vmail.messages 1.001171e+00
3             total.day.minutes 1.021520e+07
4               total.day.calls 1.001590e+00
5              total.day.charge 1.021520e+07
6             total.eve.minutes 2.224771e+06
7               total.eve.calls 1.001393e+00
8              total.eve.charge 2.224773e+06
9           total.night.minutes 6.317695e+05
10            total.night.calls 1.001704e+00
11           total.night.charge 6.317674e+05
12           total.intl.minutes 6.825133e+04
13             total.intl.calls 1.002948e+00
14            total.intl.charge 6.825163e+04
15 number.customer.service.calls 1.002040e+00
```

```
> vifcor(Z_1[,1:15], th = 0.9)
4 variables from the 15 input variables have collinearity problem:

total.day.charge total.eve.charge total.night.minutes total.intl.charge

After excluding the collinear variables, the linear correlation coefficients
ranges between:
min correlation ( total.intl.calls ~ number.vmail.messages ):  0.0001243302
max correlation ( total.day.calls ~ account.length ):  0.02824023

---------- VIFs of the remained variables --------
                    Variables      VIF
1              account.length 1.001434
2         number.vmail.messages 1.000725
3             total.day.minutes 1.000753
4               total.day.calls 1.001278
5             total.eve.minutes 1.001315
6               total.eve.calls 1.000500
7             total.night.calls 1.001318
8            total.night.charge 1.001607
9            total.intl.minutes 1.001066
10             total.intl.calls 1.001341
11 number.customer.service.calls 1.001039
```

From VIF test and corrgram and corrplot plots we can come to a conclusion that total day charge , total eve charge , total night minutes and total international charge have a correlation coefficient of 1 . So these variables are dropped.

2.2 Modeling

2.2.1 Model Selection

Model selection depends on many factors . No model is 100 % perfect or accurate. Each model has its own strength and weakness . Our main criteria for model selection here is **Accuracy** and **False Negative Rate (FNR)** . The main things that companies expect from us is accuracy of the model . Next thing that the company requires is False Negative Rate (FNR).

False Negative Rate can be obtained from the Confusion matrix. Formula for FNR is given by :

$$\textbf{FNR} = \frac{FN}{FN+TP}$$

Why FNR is important ? Just take this scenario . The company wants to predict which customers churn and who will not. So if our model wrongly predicted that n number of customers won't churn but actually those customers churn then it is a loss to the company . So many companies focus on Accuracy and FNR . Here we are also going to show FPR that is the False Positive Rate. But the main things are accuracy and FNR .

Formula for False Positive Rate is given by

$$\textbf{FPR} = \frac{FP}{FP+TN}$$

Here we are going to try the following models. All of the models we use are supervised machine learning models. The models we try are :

- Decision Tree Classifier
- Random Forest Classifier
- KNN Classifier
- Logistic Regression
- Naïve Bayes

Then based on the accuracy and FNR we are going to select the best model.

2.2.2 Decision Tree (Classification)

First model that we are going to try is Decision Tree Classification. The results are below :

```
> DT

Call:
C5.0.formula(formula = Churn ~ ., data = train, trials = 100, rules = TRUE)

Rule-Based Model
Number of samples: 3333
Number of predictors: 15

Number of boosting iterations: 100
Average number of rules: 17.9

Non-standard options: attempt to group attributes


> summary(DT)



          (a)   (b)    <-classified as
         ----  ----
         2849     1    (a): class 1
          102   381    (b): class 2
```

```
Attribute usage:

100.00% state
100.00% international.plan
100.00% voice.mail.plan
100.00% number.vmail.messages
100.00% total.day.minutes
100.00% total.eve.minutes
100.00% total.night.charge
100.00% total.intl.minutes
100.00% total.intl.calls
100.00% number.customer.service.calls
 99.94% account.length
 99.94% total.eve.calls
 99.91% total.day.calls
 99.79% total.night.calls
 97.30% area.code
```

2.2.3 Random Forest (Classification)

Next we perform Random forest Regression on our dataset . It is shown below :

```
> RF

Call:
 randomForest(formula = Churn ~ ., data = train, importance = TRUE,      ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 8.58%
Confusion matrix:
     1   2 class.error
1 2826  24 0.008421053
2  262 221 0.542443064
```

2.2.4 Naïve Bayes

We perform Naïve Bayes next . The results are :

```
> NB

Naive Bayes Classifier for Discrete Predictors
```

```
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        1         2 
0.8550855 0.1449145 

Conditional probabilities:
   state
Y             1           2           3           4           5           6
7           8
  1 0.017192982 0.025263158 0.015438596 0.021052632 0.008771930 0.020000000
0.021754386 0.017192982
  2 0.006211180 0.016563147 0.022774327 0.008281573 0.018633540 0.018633540
0.024844720 0.010351967
   state
Y             9          10          11          12          13          14
15          16
  1 0.018245614 0.019298246 0.016140351 0.017543860 0.014385965 0.022456140
0.018596491 0.021754386
  2 0.018633540 0.016563147 0.016563147 0.006211180 0.006211180 0.018633540
0.010351967 0.018633540
   state
Y            17          18          19          20          21          22
23          24
  1 0.020000000 0.017894737 0.016491228 0.018947368 0.018596491 0.017192982
0.020000000 0.024210526
  2 0.026915114 0.016563147 0.008281573 0.022774327 0.035196687 0.026915114
0.033126294 0.031055901
   state
Y            25          26          27          28          29          30
31          32
  1 0.019649123 0.017894737 0.018947368 0.020000000 0.019649123 0.019649123
0.016491228 0.017543860
  2 0.014492754 0.028985507 0.028985507 0.022774327 0.012422360 0.010351967
0.018633540 0.037267081
   state
Y            33          34          35          36          37          38
39          40
  1 0.019649123 0.018245614 0.023859649 0.023859649 0.018245614 0.023508772
0.012982456 0.020701754
  2 0.012422360 0.028985507 0.031055901 0.020703934 0.018633540 0.022774327
0.016563147 0.012422360
   state
Y            41          42          43          44          45          46
47          48
  1 0.016140351 0.018245614 0.016842105 0.018947368 0.021754386 0.025263158
0.022807018 0.018245614
  2 0.028985507 0.016563147 0.010351967 0.037267081 0.020703934 0.010351967
0.016563147 0.028985507
   state
Y            49          50          51
  1 0.024912281 0.033684211 0.023859649
  2 0.014492754 0.020703934 0.018633540

   account.length
```

```
Y        [,1]     [,2]
  1 100.2278 39.08702
  2 101.7049 38.07804

   area.code
Y           1         2         3
  1 0.2512281 0.4978947 0.2508772
  2 0.2525880 0.4886128 0.2587992

   international.plan
Y            1          2
  1 0.93473684 0.06526316
  2 0.71635611 0.28364389

   voice.mail.plan
Y           1         2
  1 0.7045614 0.2954386
  2 0.8343685 0.1656315

   number.vmail.messages
Y        [,1]     [,2]
  1 8.266894 13.42101
  2 4.861321 11.31752

   total.day.minutes
Y        [,1]     [,2]
  1 175.9248 49.04906
  2 203.6455 65.66908

   total.day.calls
Y        [,1]     [,2]
  1 100.4016 19.22174
  2 101.8672 19.91935

   total.eve.minutes
Y       [,1]     [,2]
  1 199.540 48.62416
  2 210.737 49.46702

   total.eve.calls
Y        [,1]     [,2]
  1 100.1427 19.32389
  2 100.2619 19.30840

   total.night.calls
Y         [,1]     [,2]
  1  99.91995 18.85981
  2 100.30714 19.78597

   total.night.charge
Y        [,1]     [,2]
  1 9.001332 2.197745
  2 9.233660 2.074083

   total.intl.minutes
Y        [,1]    [,2]
  1 10.22484 2.55913
```

```
  2 10.60326 2.61956

   total.intl.calls
Y        [,1]     [,2]
  1 4.331204 2.066883
  2 3.951661 2.087634

   number.customer.service.calls
Y        [,1]      [,2]
  1 1.307145 0.9516122
  2 1.286457 0.8627472
```

2.2.5 KNN Classifier

We have tried the KNN Classifier .

```
KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(C_train ,
D_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
             metric_params=None, n_jobs=1, n_neighbors=9, p=2,
             weights='uniform')

KNN_Pred

array(['No', 'No', 'Yes', ..., 'No', 'No', 'No'], dtype=object)
```

2.2.6 Logistic Regression

We try Logistic Regression

```
> summary(LR)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8403  -0.5438  -0.3872  -0.2391   2.9745

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -7.3951633  0.9425014  -7.846 4.28e-15 ***
state2                         0.3866902  0.7462238   0.518  0.60432
```

state3	1.2309748	0.7353124	1.674	0.09411	.
state4	0.1248106	0.8350688	0.149	0.88119	
state5	1.7041886	0.7780449	2.190	0.02850	*
state6	0.9136893	0.7434416	1.229	0.21907	
state7	0.9295897	0.7185633	1.294	0.19578	
state8	0.4871016	0.8043126	0.606	0.54477	
state9	0.7508702	0.7407791	1.014	0.31076	
state10	0.5882310	0.7492109	0.785	0.43237	
state11	0.9482560	0.7574355	1.252	0.21060	
state12	-0.1215110	0.8843042	-0.137	0.89071	
state13	0.3775483	0.8788406	0.430	0.66749	
state14	0.9555782	0.7346442	1.301	0.19335	
state15	-0.3336505	0.8179853	-0.408	0.68335	
state16	0.6750611	0.7331638	0.921	0.35718	
state17	0.9706802	0.7170385	1.354	0.17582	
state18	0.8765842	0.7511625	1.167	0.24322	
state19	0.5483372	0.8247001	0.665	0.50612	
state20	0.9483025	0.7296424	1.300	0.19371	
state21	1.3014402	0.7027277	1.852	0.06403	.
state22	1.4052555	0.7152533	1.965	0.04945	*
state23	1.3905361	0.7011241	1.983	0.04733	*
state24	1.0852241	0.7016885	1.547	0.12196	
state25	0.5874331	0.7654287	0.767	0.44281	
state26	1.4019528	0.7156599	1.959	0.05012	.
state27	1.7449429	0.7090230	2.461	0.01385	*
state28	0.7302174	0.7334863	0.996	0.31947	
state29	0.1433235	0.7800795	0.184	0.85423	
state30	0.3265818	0.7938065	0.411	0.68077	
state31	1.0564204	0.7487985	1.411	0.15830	
state32	1.5931603	0.6968849	2.286	0.02225	*
state33	0.3484025	0.7775607	0.448	0.65410	
state34	1.1791144	0.7150223	1.649	0.09914	.
state35	1.2546673	0.7031799	1.784	0.07438	.
state36	0.7152704	0.7284949	0.982	0.32618	
state37	1.0141622	0.7368330	1.376	0.16870	
state38	0.8314878	0.7206808	1.154	0.24860	
state39	0.9130234	0.7700761	1.186	0.23577	
state40	0.0906503	0.7910572	0.115	0.90877	
state41	1.7630830	0.7176538	2.457	0.01402	*
state42	0.7443243	0.7508666	0.991	0.32155	
state43	0.2228598	0.8121268	0.274	0.78377	
state44	1.6417131	0.6989247	2.349	0.01883	*
state45	0.8974920	0.7339902	1.223	0.22142	
state46	-0.2619864	0.8012927	-0.327	0.74370	
state47	0.4230338	0.7482408	0.565	0.57182	
state48	1.3493456	0.7129642	1.893	0.05841	.
state49	0.3286607	0.7599121	0.432	0.66538	
state50	0.4817264	0.7247202	0.665	0.50624	
state51	0.3225021	0.7422603	0.434	0.66394	
account.length	0.0011946	0.0013955	0.856	0.39198	
area.code2	-0.0482982	0.1345560	-0.359	0.71964	
area.code3	-0.0371079	0.1541762	-0.241	0.80980	
international.plan2	1.9552917	0.1432726	13.647	< 2e-16	***
voice.mail.plan2	-1.6964311	0.5447467	-3.114	0.00184	**
number.vmail.messages	0.0268652	0.0180461	1.489	0.13657	
total.day.minutes	0.0105440	0.0010633	9.916	< 2e-16	***
total.day.calls	0.0038948	0.0028090	1.387	0.16558	

```
total.eve.minutes             0.0058149  0.0011399   5.101 3.37e-07 ***
total.eve.calls               0.0005799  0.0028599   0.203  0.83933
total.night.calls             0.0004034  0.0028563   0.141  0.88768
total.night.charge            0.0815238  0.0250419   3.255  0.00113 **
total.intl.minutes            0.0580760  0.0215356   2.697  0.00700 **
total.intl.calls             -0.1159273  0.0274564  -4.222 2.42e-05 ***
number.customer.service.calls  0.0135372  0.0574848   0.235  0.81383
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2758.3  on 3332  degrees of freedom
Residual deviance: 2285.0  on 3267  degrees of freedom
AIC: 2417

Number of Fisher Scoring iterations: 5
```

Chapter 3

Conclusion

3.1 Model Evaluation

After training the model and predicting the outcomes it is now time to evaluate the performance of the model. We have tried a total of 5 machine learning models . All these models are supervised machine learning models. We are going to evaluate the models based on the **Accuracy** and **False Negative Rate (FNR)** . Although Recall and False Positive Rate are also important here we are going to concentrate only on these two parameters .

3.1.1 Error Metrics & Accuracy

As said earlier we are going to evaluate the model based on Accuracy and False Negative Rate .

$$\text{Accuracy} = \frac{((\boldsymbol{TP}+\boldsymbol{TN})*\mathbf{100})}{(\boldsymbol{TP}+\boldsymbol{TN}+\boldsymbol{FP}+\boldsymbol{FN})}$$

$$\text{FNR} = \frac{\boldsymbol{FP}}{\boldsymbol{FP}+\boldsymbol{TN}}$$

$$\text{FPR} = \frac{FN}{FN+TP}$$

$$\text{Recall} = \frac{(TP*100)}{TP+FN}$$

3.1.2 Decision Tree Classification

As said above for evaluating the model we are focusing on only Accuracy and False Negative Rate (FNR) . We are leaving out False Positive Rate and Recall .

ACCURACY = 93.7

FNR = 45.5

3.1.3 Random Forest Classification

Accuracy and FNR given below

ACCURACY = 92

FNR = 53.1

3.1.4 Naïve Bayes

Accuracy and FNR given below

ACCURACY = 88

FNR = 77.6

3.1.5 KNN Classifier

Accuracy and FNR given below

ACCURACY = 89

FNR = 19.4

3.1.6 Logistic Regression

Accuracy and FNR given below

ACCURACY = 87.3

FNR = 80.8

3.2 Model Selection

From the above results KNN Classifier produced the best results. Since FNR is also important other than accuracy so we choose the model which has significantly less FNR compared to other models . Decision Tree and Random Forest comes next to KNN as good models . But Naïve BAyes and Logistic Regression performed the worst . They are ranked as below :

1. **KNN – Best**
2. **Decision Tree – Better**

3. **Random Forest – Good**
4. **Naïve Bayes – Bad**
5. **Logistic Regression – Worst**

3.3 Output

Example of output with a sample input.

Output for Decision Tree :

```
> DT_pred
   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  [56] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [111] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1
 [166] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1
 [221] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [276] 2 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [331] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
 [386] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [441] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
 [496] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [551] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [606] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1
 [661] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [716] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [771] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
 [826] 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [881] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1
 [936] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 1 1
1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
 [991] 1 1 1 1 1 2 1 2 1 2
 [ reached getOption("max.print") -- omitted 667 entries ]
Levels: 1 2
```

Output for Naïve Bayes :

```
    > NBPred
   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1
  [56] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [111] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2
 [166] 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2 1 1
 [221] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
 [276] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [331] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
 [386] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [441] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
 [496] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
 [551] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [606] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1
1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1
 [661] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
 [716] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [771] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [826] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [881] 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
 [936] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1
1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
 [991] 1 2 1 1 1 2 1 1 1 2
 [ reached getOption("max.print") -- omitted 667 entries ]
Levels: 1 2
```

Remaining output for other models can be found in the code files .

Notes

1. Please note that the train and test dataset that is provided has been combined that is merged (row binded) into a single dataset for easier data pre processing (data cleaning). Later after data pre processing same rows of the train and test data has been manually selected for train and test data for feeding into the model.
2. The example output can be found in the code files itself . Besides output of two models are provided here for reference .
3. The full code for the project can be found in R and python files. (Project 2.R and Project2.ipynb)
4. R code and python code are not provided here since they can be found in the code files itself.
5. In codes the necessary information or notes (if required) is provided in comments (#) .
6. The code file submitted is the final code.
7. Full care has been taken to ensure that the code runs properly. So if any error is encountered during code execution kindly re - run the code.

R Code

Please find the code file attached with this project report (Project 2.R)

Python Code

Please find the code file attached with this project report (Project 2.ipynb)