CHURN ANALYSIS

SHALINI KRISHNAN

CONTENTS

1. PROJECT OBJECTIVE

The main objective of this predictive modelling project is to predict the churn rate of the customers in Telecom sector. The project simulates one such case of customer churn where we work on a data of postpaid customers with a contract. Predictive models are built using the following, and the same is been compared for the best performance among them.

- Logistic Regression
- K-Nearest Neighbor
- Naïve Bayes

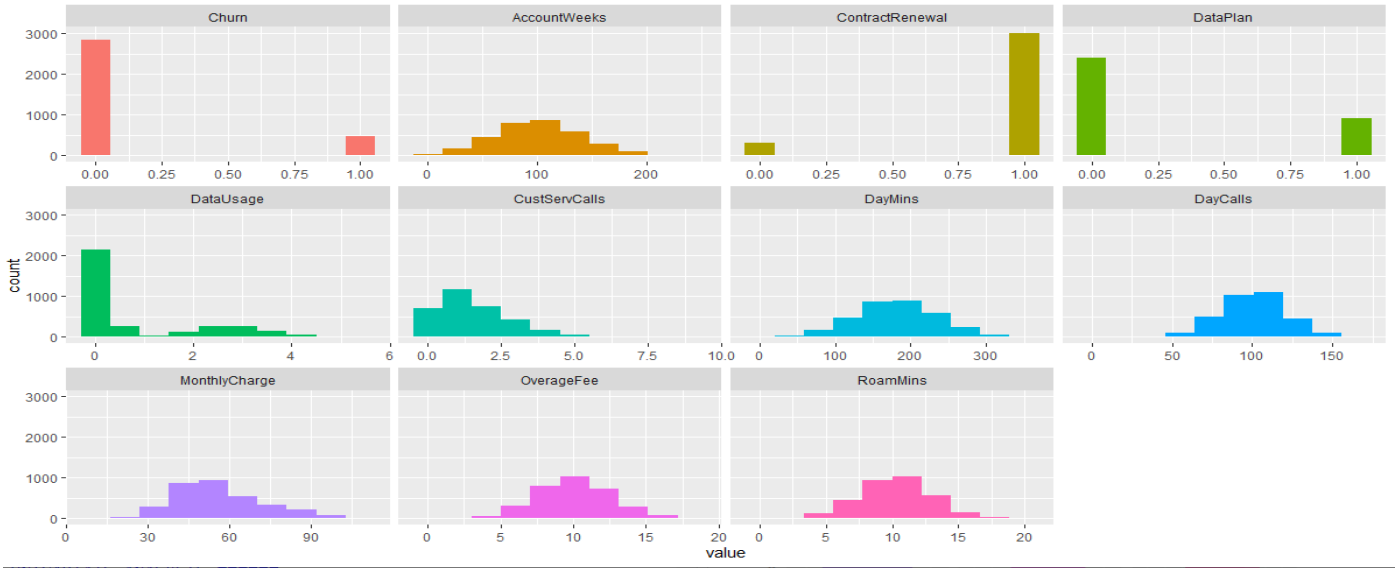
2. EXPLORATORY DATA ANALYSIS

2.1 BASIC DATA SUMMARY

- The provided dataset consists of 3333 rows and 11 columns.
- The dataset contains both categorical and continuous variables.
- The variable names are “Churn”, “AccountWeeks”, “DataPlan”, “DataUsage”, “ContractRenewal”, “CustServCalls”, ”DayMins”, “DayCalls”, “MonthlyCharge”, “OverageFee”, “RoamMins”.
- The 5 point summary is calculated for all the variables. The measures of Central Tendency and measures of Dispersion are also examined. Refer source code.
- Few variables are not of appropriate datatype. So we correct them as and when necessary according to the assumptions of the models to be built.

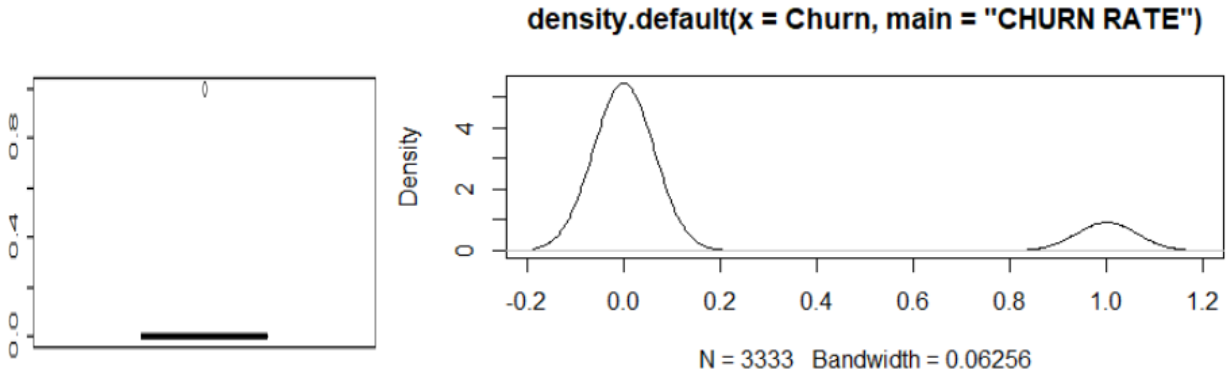
2.2 UNIVARIATE ANALYSIS

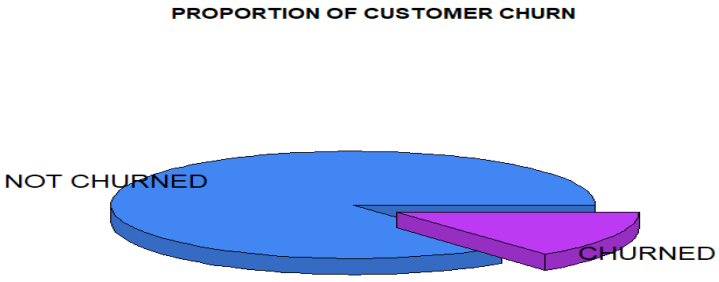
The following shows the univariate analysis of the variables to learn more about its nature.



CHURN

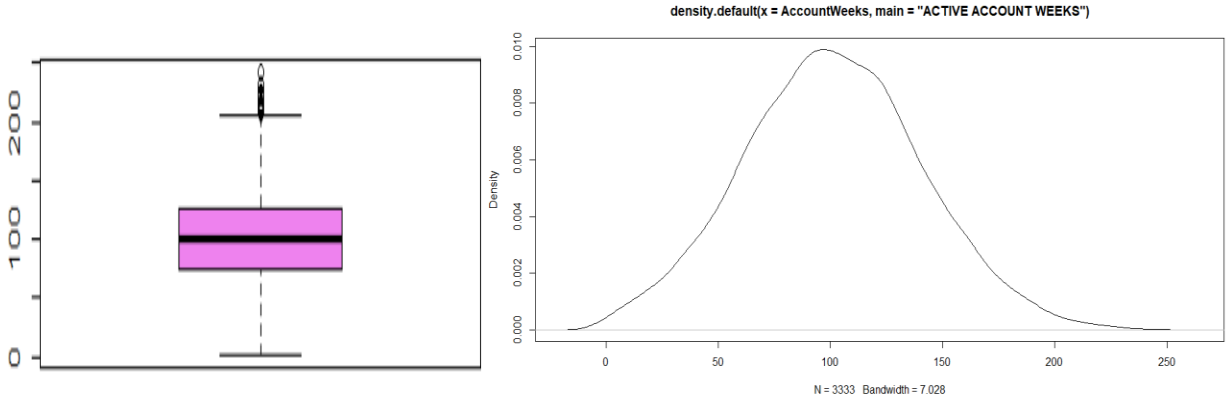
It is the proportion of the customers churned or not churned.'1' is churned and '0' is not churned in the given dataset. It seems that proportion of customers not churned is more than the proportion of customers who have churned.





ACCOUNT WEEKS

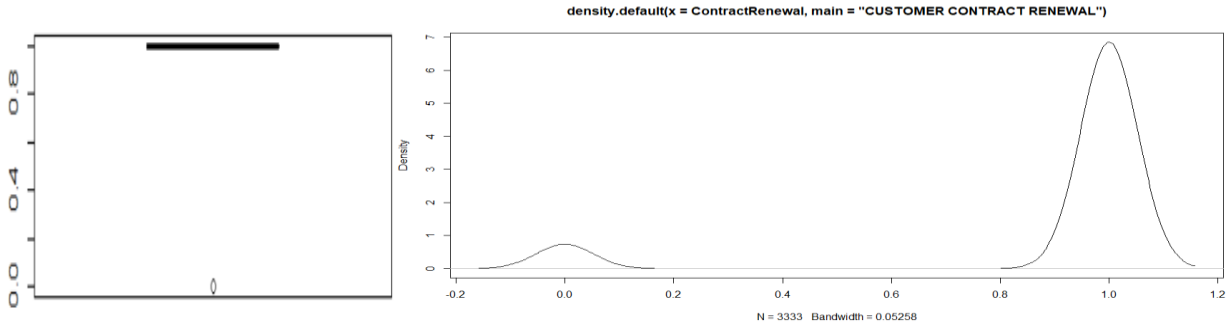
It is the number of weeks customers have had an active account.

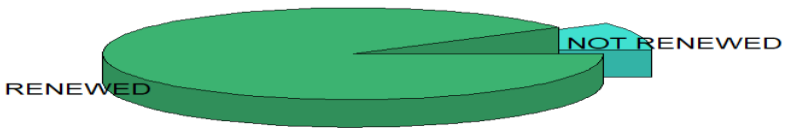


The distribution is slightly right skewed with the presence of outliers on the positive side.

CONTRACT RENEWAL

It is the proportion of the customers who recently renewed the contract or not.’1’ if customer renewed the contract and’0’ if not.

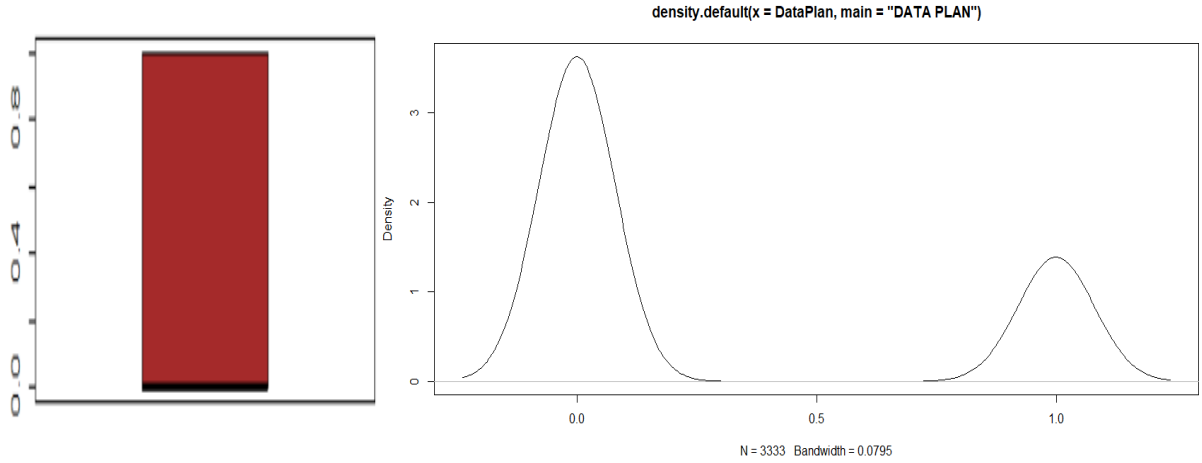




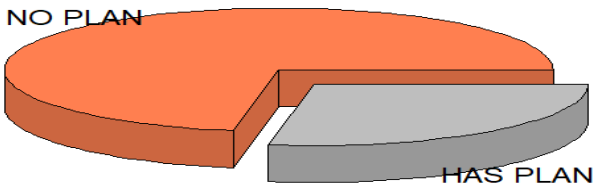
It seems the proportion of the customers recently renewed the contract is more than the customers who have not renewed.

DATA PLAN

It shows the proportion of customers having a data plan or not.’1’ if the customer has a data plan and ‘0’ if not.



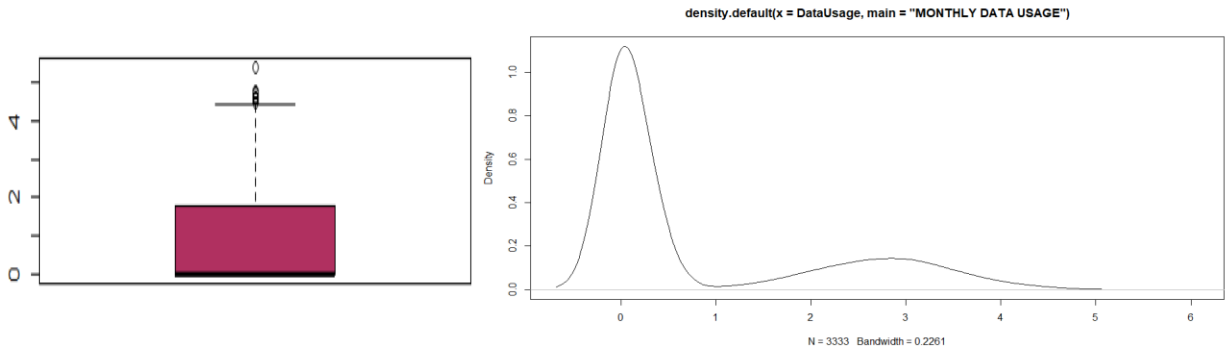
CUSTOMER WITH OR WITHOUT DATAPLAN



The proportion of the customers without a data plan is more than the customers with a data plan.

DATA USAGE

It is the monthly data usage of the customer in gigabytes.

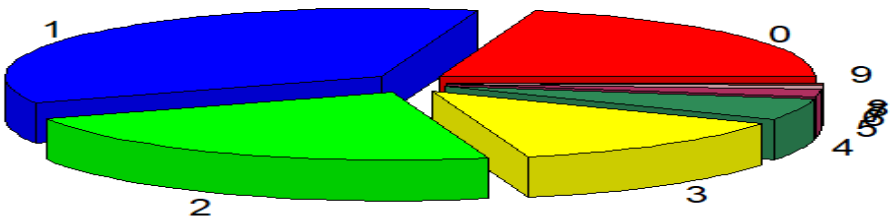


The distribution is skewed with the presence of outliers on the positive side.

CUSTOMER SERVICE CALLS

It is the number of calls customer had made to the customer service center.

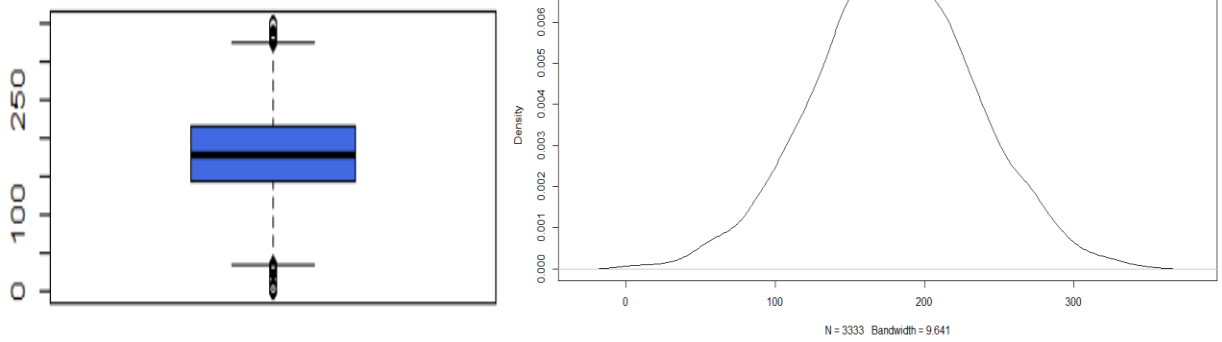
PROPORTION OF CUSTOMER SERVICE CALLS



The number of calls ranges from 0 – 9.The proportion of ‘1’ call is more followed by ‘2’ calls and then ‘0’ (no call).

DAYMINS

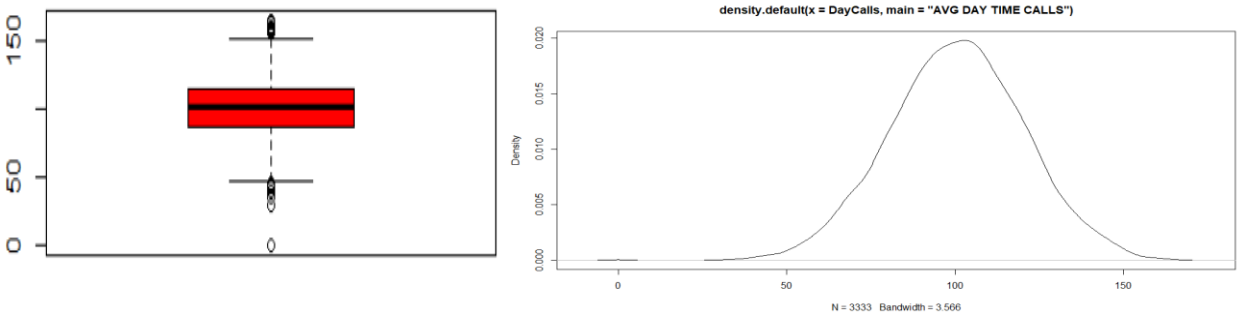
It is the average minutes of calls made by the customers per month.



The distribution is slightly skewed on left tail and shows the presence of the outbound values on both the sides.

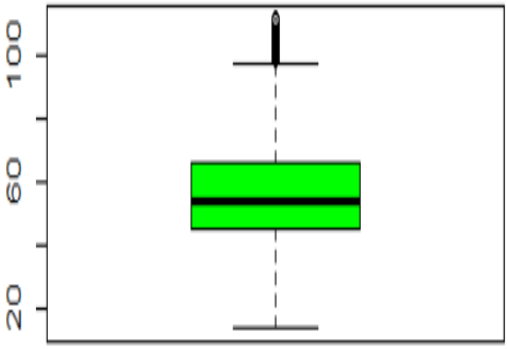
DAY CALLS

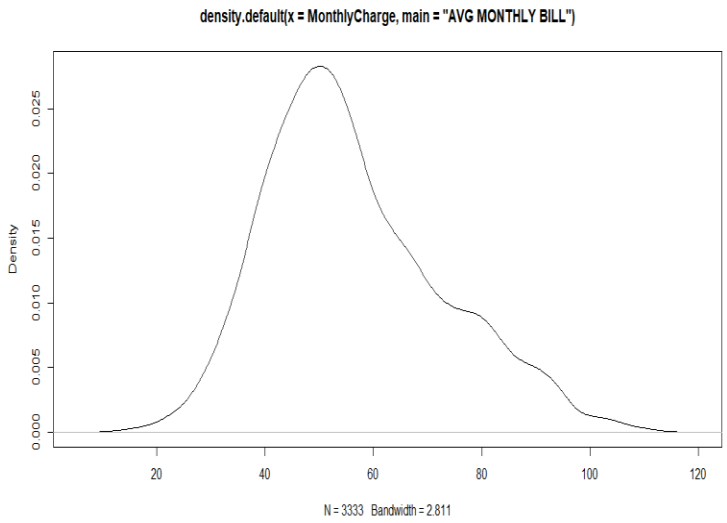
It is the average number of day time calls made by the customer. The distribution is skewed on left side with the presence of outbound values on both positive and negative tails.



MONTHLY CHARGE

It is the average monthly bill generated for the customers.

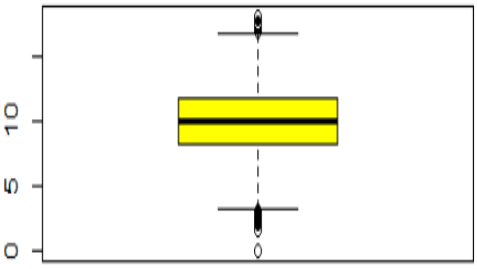


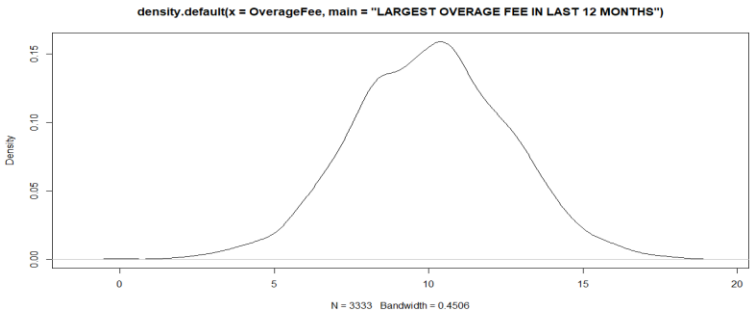


The distribution seems to be right skewed with the presence of outliers on the positive side.

OVERAGE FEE

It is the largest overage fee paid by the customers in the last 12 months.

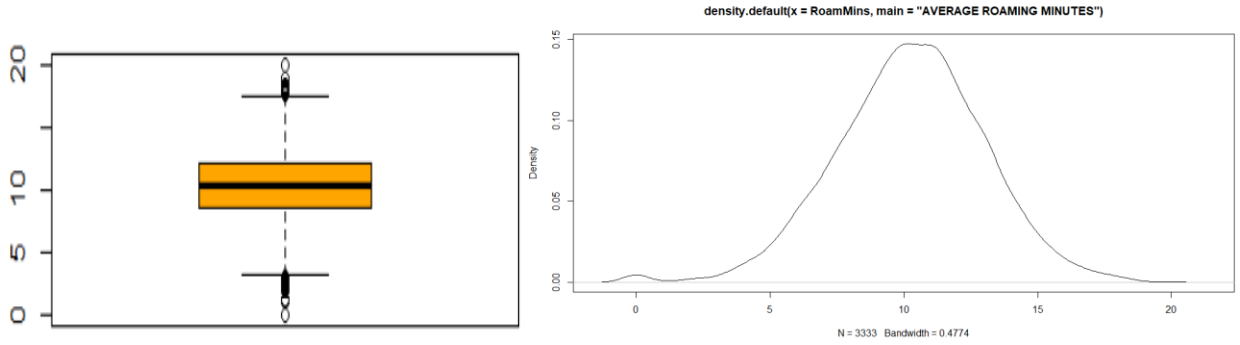




The distribution is left skewed with the presence of outliers on both the tails.

ROAM MINS

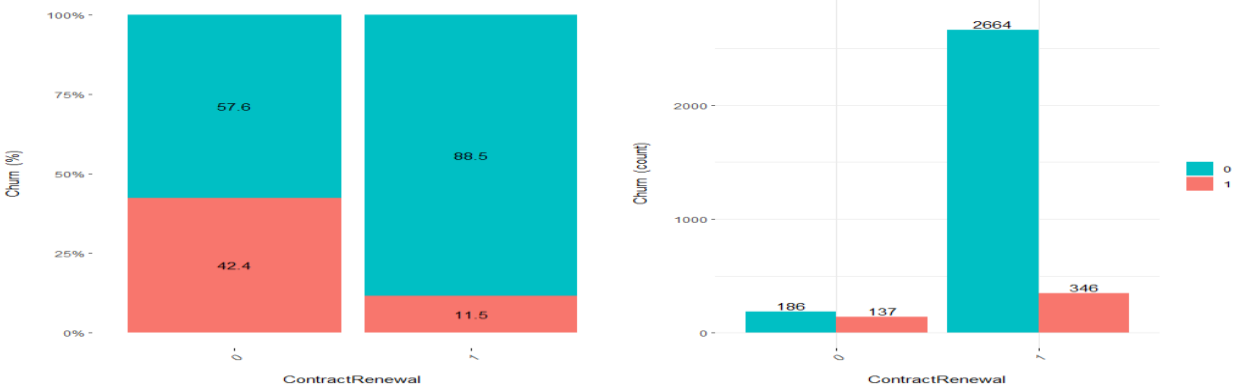
It is the average of roaming minutes utilized by the customer.

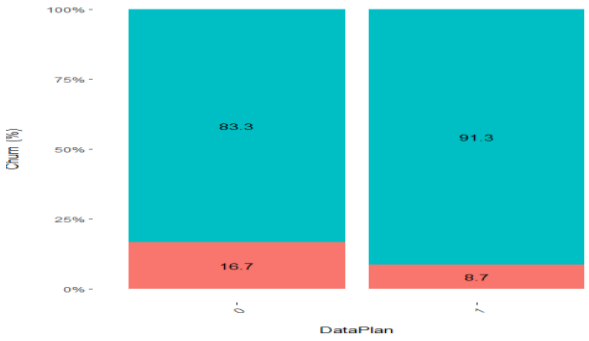


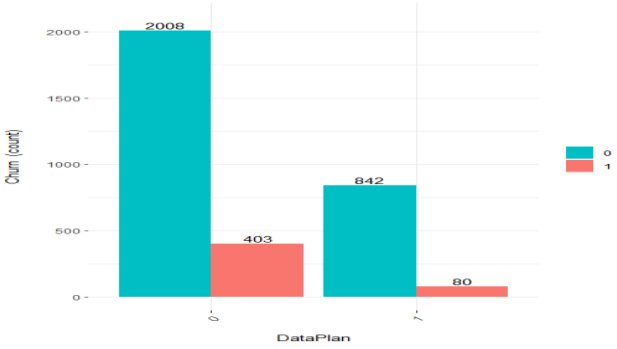
The distribution is slightly left skewed with the presence of outliers on both positive and negative tails.

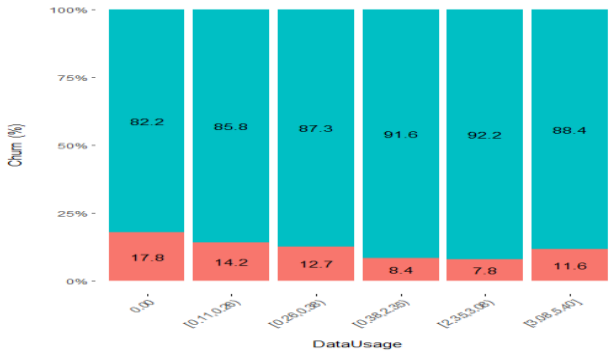
2.3 BI-VARIATE ANALYSIS

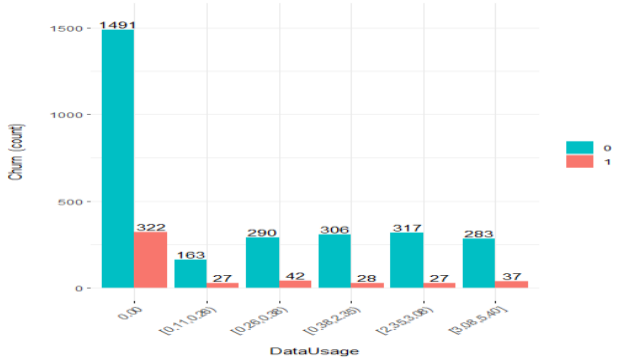
The following shows the bivariate analysis of the target variable "Churn" with all the other predictor variables.

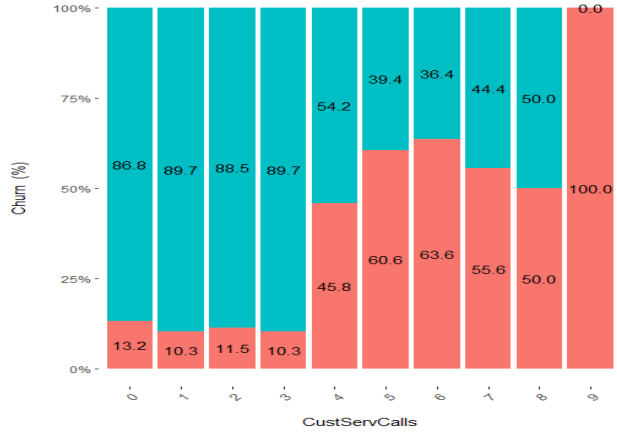


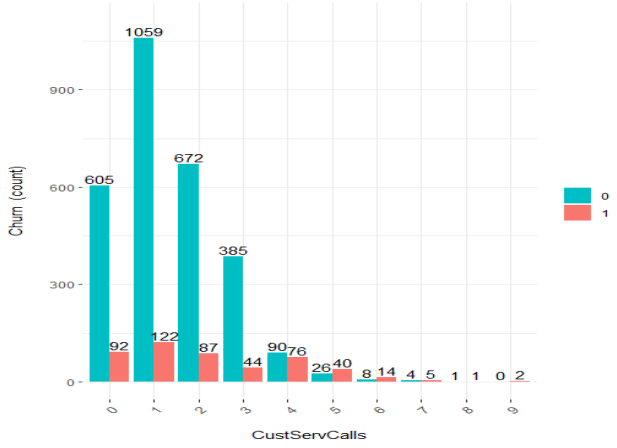

Churn (%)
100%
75%
50%
25%
0%
83.3
16.7
91.3
8.7
0
1
DataPlan

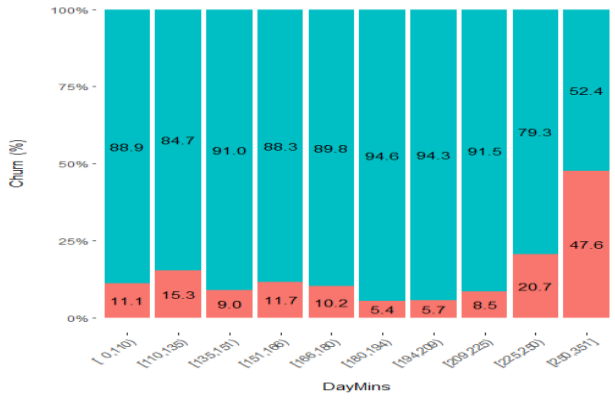

Churn (count)
2000
1500
1000
500
0
2008
403
842
80
0
1
DataPlan

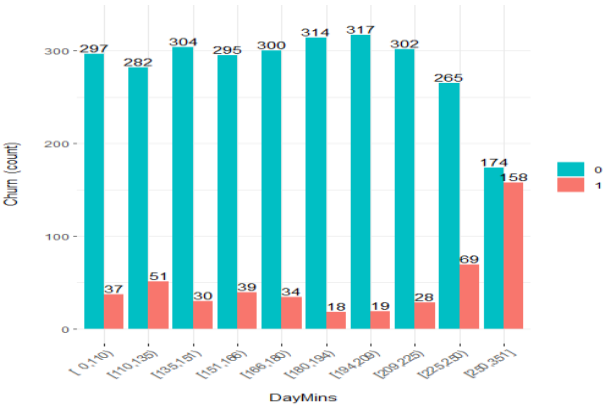

Churn (%)
100%
75%
50%
25%
0%
82.2
17.8
85.8
14.2
87.3
12.7
91.6
8.4
92.2
7.8
88.4
11.6
0.00
[0.11,0.26)
[0.26,0.38)
[0.38,2.35)
[2.35,3.06)
[3.06,5.40)
DataUsage

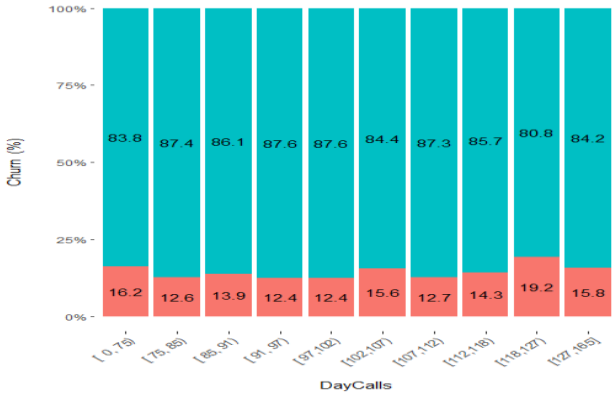

Churn (count)
1500
1000
500
0
1491
322
163
27
290
42
306
28
317
27
283
37
0.00
[0.11,0.26)
[0.26,0.38)
[0.38,2.35)
[2.35,3.06)
[3.06,5.40)
DataUsage
0
1

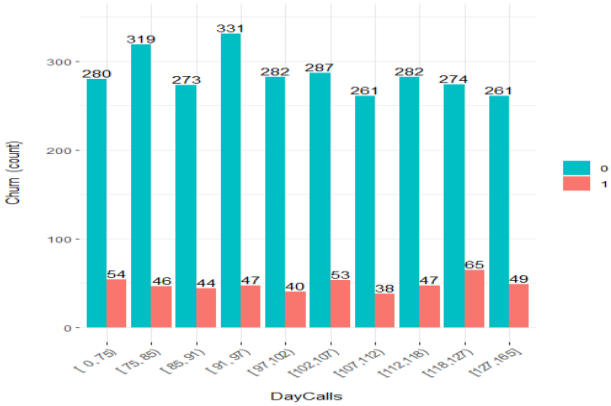

Churn (%)
100%
75%
50%
25%
0%
86.8
13.2
89.7
10.3
88.5
11.5
89.7
10.3
54.2
45.8
39.4
60.6
36.4
63.6
44.4
55.6
50.0
50.0
0.0
100.0
0
1
2
3
4
5
6
7
8
9
CustServCalls

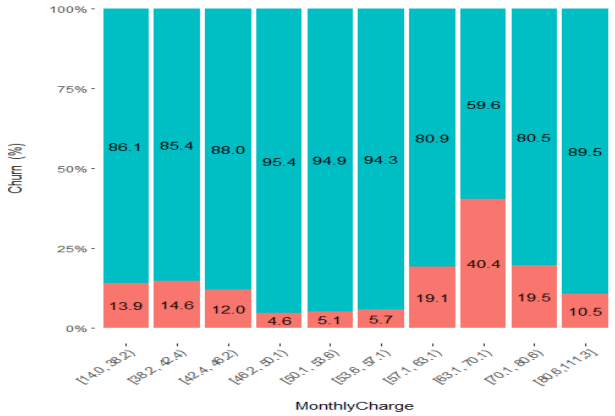

Churn (count)
900
600
300
0
605
92
1059
122
672
87
385
44
90
76
26
40
8
14
4
5
1
1
0
2
0
1
2
3
4
5
6
7
8
9
CustServCalls
0
1

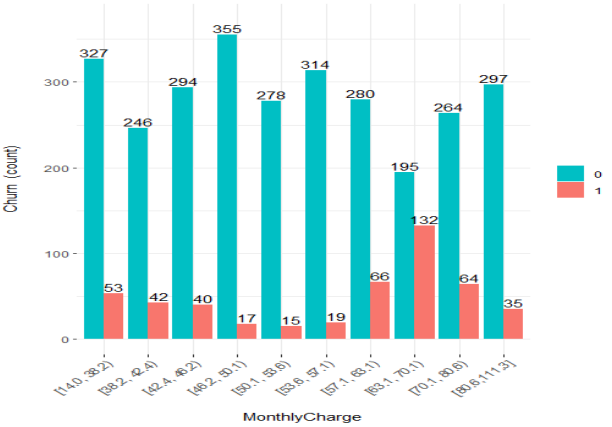

Churn (%)
100%
75%
50%
25%
0%
88.9
84.7
91.0
88.3
89.8
94.6
94.3
91.5
79.3
52.4
11.1
15.3
9.0
11.7
10.2
5.4
5.7
8.5
20.7
47.6
[0,110)
[110,135)
[135,151)
[151,166)
[166,180)
[180,194)
[194,209)
[209,225)
[225,250)
[250,351]
DayMins

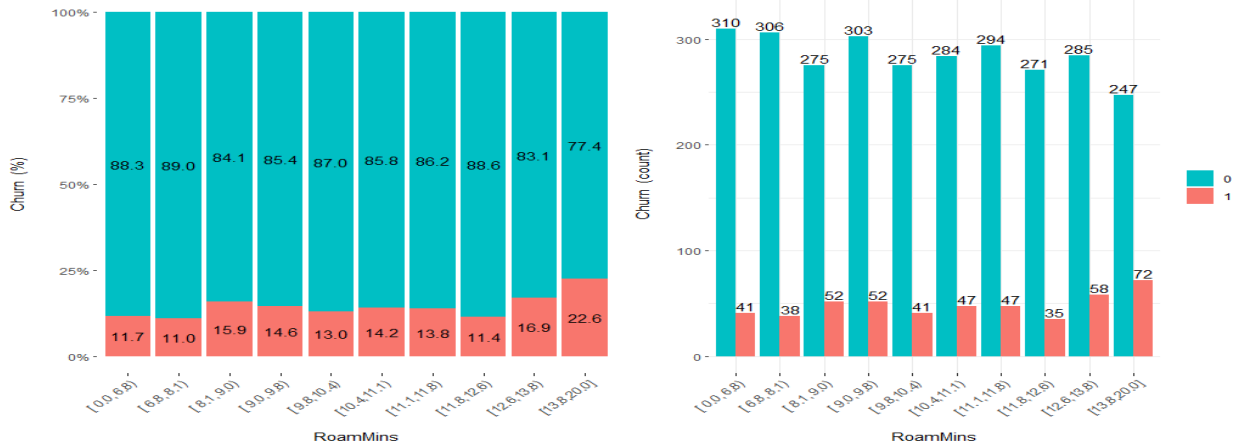

Churn (count)
300
200
100
0
297
282
304
295
300
314
317
302
265
174
37
51
30
39
34
18
19
28
69
158
0
1
DayMins


Churn (%)
100%
75%
50%
25%
0%
83.8
87.4
86.1
87.6
87.6
84.4
87.3
85.7
80.8
84.2
16.2
12.6
13.9
12.4
12.4
15.6
12.7
14.3
19.2
15.8
[0, 75)
[75, 85)
[85, 91)
[91, 97)
[97,102)
[102,107)
[107,112)
[112,118)
[118,127)
[127,165]
DayCalls


Churn (count)
300
200
100
0
280
319
273
331
282
287
261
282
274
261
54
46
44
47
40
53
38
47
65
49
0
1
DayCalls


Churn (%)
100%
75%
50%
25%
0%
86.1
85.4
88.0
95.4
94.9
94.3
80.9
59.6
80.5
89.5
13.9
14.6
12.0
4.6
5.1
5.7
19.1
40.4
19.5
10.5
[14.0,38.2)
[38.2,42.4)
[42.4,46.2)
[46.2,50.1)
[50.1,53.6)
[53.6,57.1)
[57.1,63.1)
[63.1,70.1)
[70.1,80.6)
[80.6,111.3]
MonthlyCharge


Churn (count)
300
200
100
0
327
246
294
355
278
314
280
195
264
297
53
42
40
17
15
19
66
132
64
35
0
1
MonthlyCharge



INFERENCES

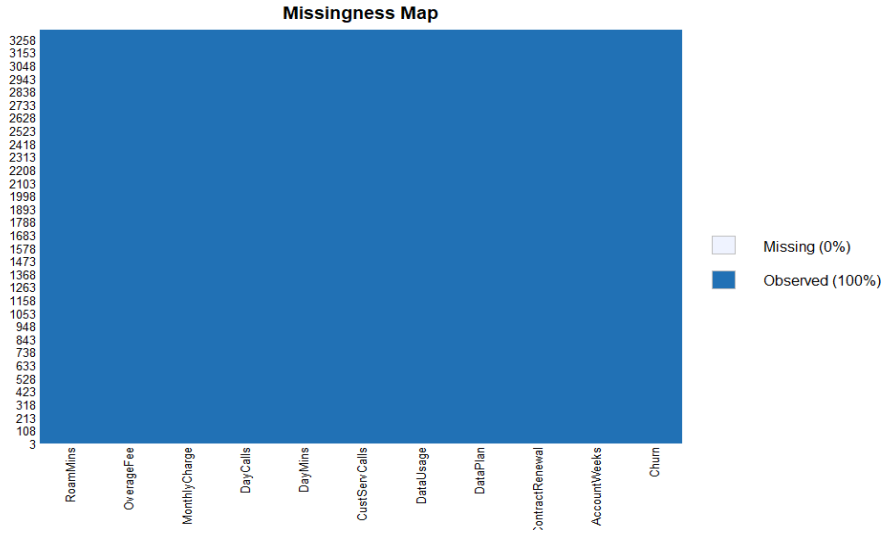
The above graphs shows the churn rate with respect to every single predictor in a detailed manner. Inferences are drawn for highest churn rate with each one of the predictors.

- When the customer's account of active connection was between 92 – 102 weeks, churn rate was more.
- 42.4 % customers churned when they did not renew the contract and 11.5 % customers churned even after renewing the contract.
- 16.7 % customers churned who had no data plan and 8.7 % customers churned who had a data plan.

- When there was 0 GHZ data usage, 17.8 % customers churned and when the data usage was between 0.11 – 0.26 GHZ, 14.2 % customers churned.
- 63.6 % customers churned when there was 6 calls to the customer service.
- 47.6 % customers churned when the average day time call minutes of the customers was within 250 – 351 mins.
- 19.2 % customers churned when the average day time calls was between 118 – 127.
- 40.4 % customers churned when their average monthly charges were between 63.1 % - 70.1 %.
- When the overage fee was between the range 13.30 % - 18.19 %, 21.3 % customers churned.
- When the average roaming minutes was between 13.8 % - 20.0 %, 22.6 % customers had churned.

2.4 MISSING VALUE IDENTIFICATION

There are no missing values in the provided dataset. Hence no missing value treatments is carried out.

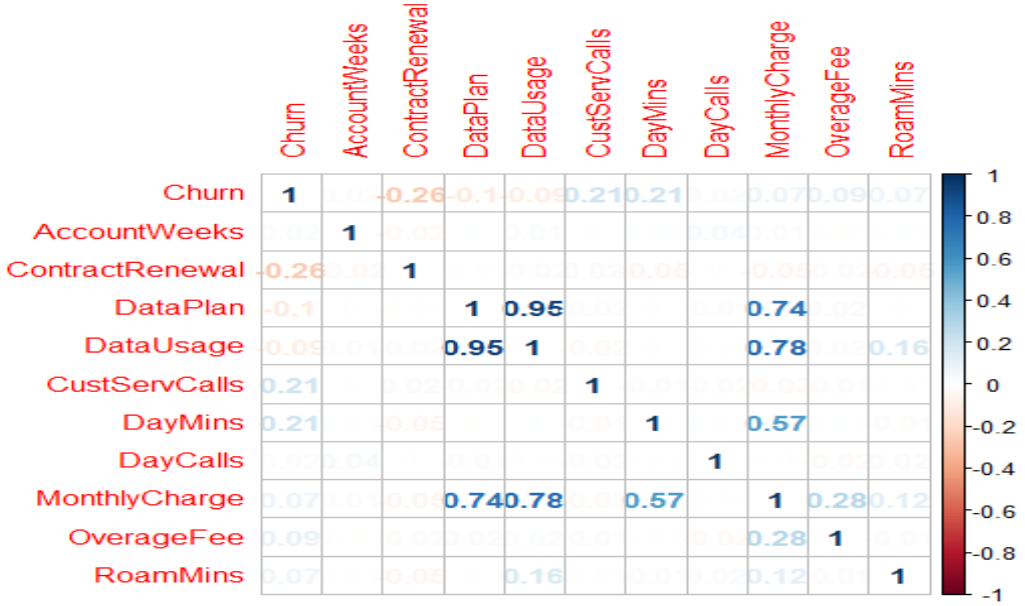


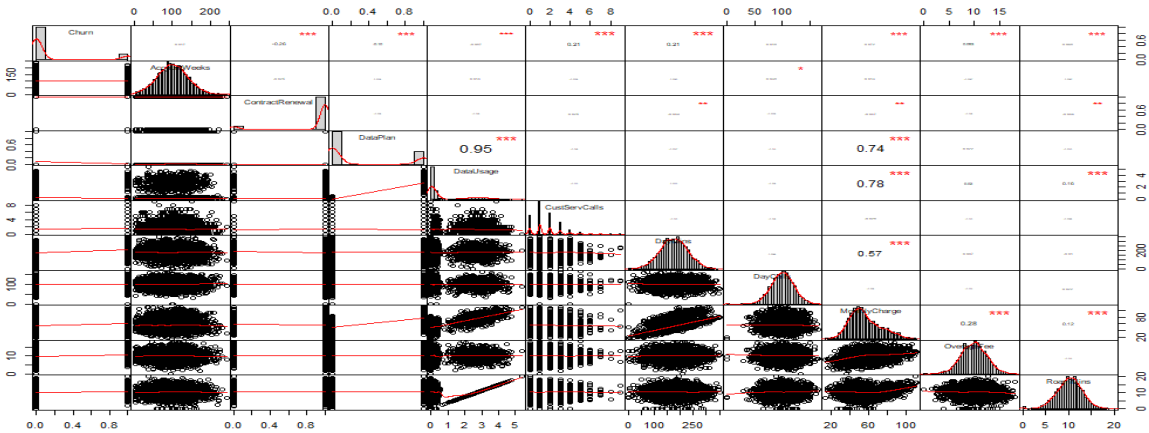
2.5 OUTLIER IDENTIFICATION

Except for the categorical variable ‘Data Plan’, the outliers are present among all the other categorical and continuous variables. Categorical variables have 2 levels at the minimum and 10 levels at the maximum (CustServCalls - 0 to 9). Not all outliers are extreme values, so we are not ignoring them on our further analysis. Refer source code.

2.6 MULTICOLLINEARITY CHECK

The following graph shows the correlation between all the variables in the dataset.





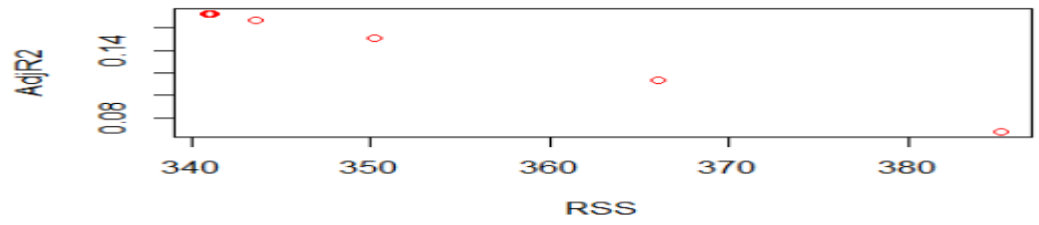
With reference to the graph above, we conclude that,

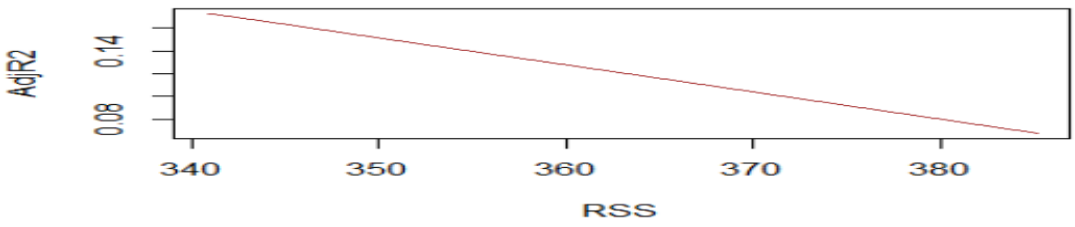
- The variables DataPlan and DataUsage are highly correlated.
- The variable MonthlyCharge is correlated with DataPlan and DataUsage on significant level.
- The variable DayMins is moderately correlated with MonthlyCharge.

Since the variables are of both categorical and continuous type, multicollinearity cannot be treated using dimensionality reduction techniques such as Principal Component Analysis. PCA would be highly feasible for continuous variables. Hence we can use Forward Selection , Backward Selection or Stepwise Selection of significant variables and dropping the insignificant ones using any linear models.

2.7 TREATMENT

Backward Selection of significant variables are done by building a Linear Probability Model and the same is been crosschecked with Subset Selection Algorithm. Refer source code.





MODELS

The only criterion of selecting the significant variables is increase in the Adjusted R2 value and decrease in the Residual Standard Error of the corresponding model. Other parameters such as CP and BIC are also taken into account when selecting the best model with the significant variables.

Now the significant explanatory variables chosen for predicting Churn are:

- ContractRenewal
- DataUsage
- CustServCalls
- DayMins
- OverageFee
- RoamMins

  Their corresponding 'vif' values are also checked which shows no sign of correlation between them.

Vif values

ContractRenewal	DataUsage	CustServCalls	DayMins	OverageFee	RoamMins
1.005646	1.028250	1.001378	1.002803	1.001151	1.029580

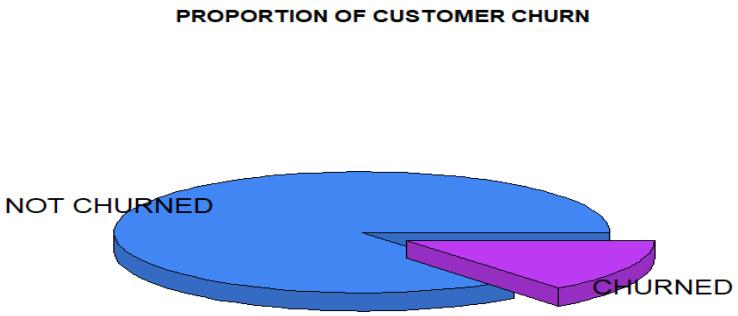
INSIGHTS

All the observed insights and inferences are discussed under each topic shown above. The model will be built upon only on the significant variables for our further analysis. The selected predictors are not correlated with one another. Refer source code.

2.8 BALANCING THE DATASET

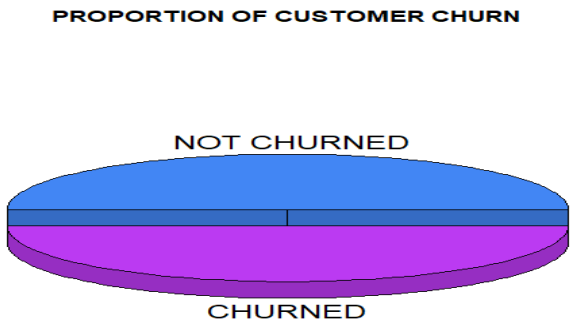
The provided dataset is highly imbalanced with the proportion of customers not churned being more than the customers who have churned. Hence building predictive models on an imbalanced data will reduce the reliability and performance of the models.

In order to build a robust model, the given dataset is balanced using SMOTE technique by under sampling the majority class and over sampling the minority class.



PROPORTION OF TARGET BEFORE BALANCING THE DATASET

PROPRTION OF TARGET AFTER BALANCING THE DATASET



3. LOGISTIC REGRESSION

Now the dataset is highly balanced with equal proportions of customer churned and not churned. The balanced dataset is further split into Train and Test sets in the ratio 70:30, where logistic model is built on Train set and is been validated using the Test set. We use one hot

encoding for the variable 'CustServCalls' which has 10 levels in it. We also choose only the significant levels among them for our analysis as shown in the source code.

3.1 INTERPRETATION

- Unlike Linear Regression, Logistic Regression does not depend on its coefficients for the model performance.
- The coefficients simply explains the predictor variables and its linear relation with the logs odd ratio of the predictand. The formula used to predict the probabilities of the target variable is,

$$\frac{\exp(\text{coefficients (log model)})}{1 + \exp(\text{coefficients (log model)})}$$

Logistic Regression's model performance is only measured through the High values of Concordance or the Confusion Matrix.

```
Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.59502    0.42481  -6.109 1.00e-09 ***
## ContractRenewal1 -2.49037    0.15710 -15.852  < 2e-16 ***
## DataUsage        -0.26025    0.04821  -5.398 6.72e-08 ***
## DayMins           0.01270    0.00106  11.976  < 2e-16 ***
## OverageFee        0.15222    0.02441   6.235 4.51e-10 ***
## RoamMins          0.11210    0.02183   5.135 2.82e-07 ***
## CALL_01          -0.58117    0.17712  -3.281  0.00103 **
## CALL_11          -0.92070    0.17064  -5.395 6.84e-08 ***
## CALL_21          -0.56376    0.17566  -3.209  0.00133 **
## CALL_31          -0.96765    0.21134  -4.579 4.68e-06 ***
## CALL_41           1.85016    0.25770   7.180 6.99e-13 ***
## CALL_51           3.51849    0.54192   6.493 8.44e-11 ***
## CALL_61           3.59288    0.76207   4.715 2.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

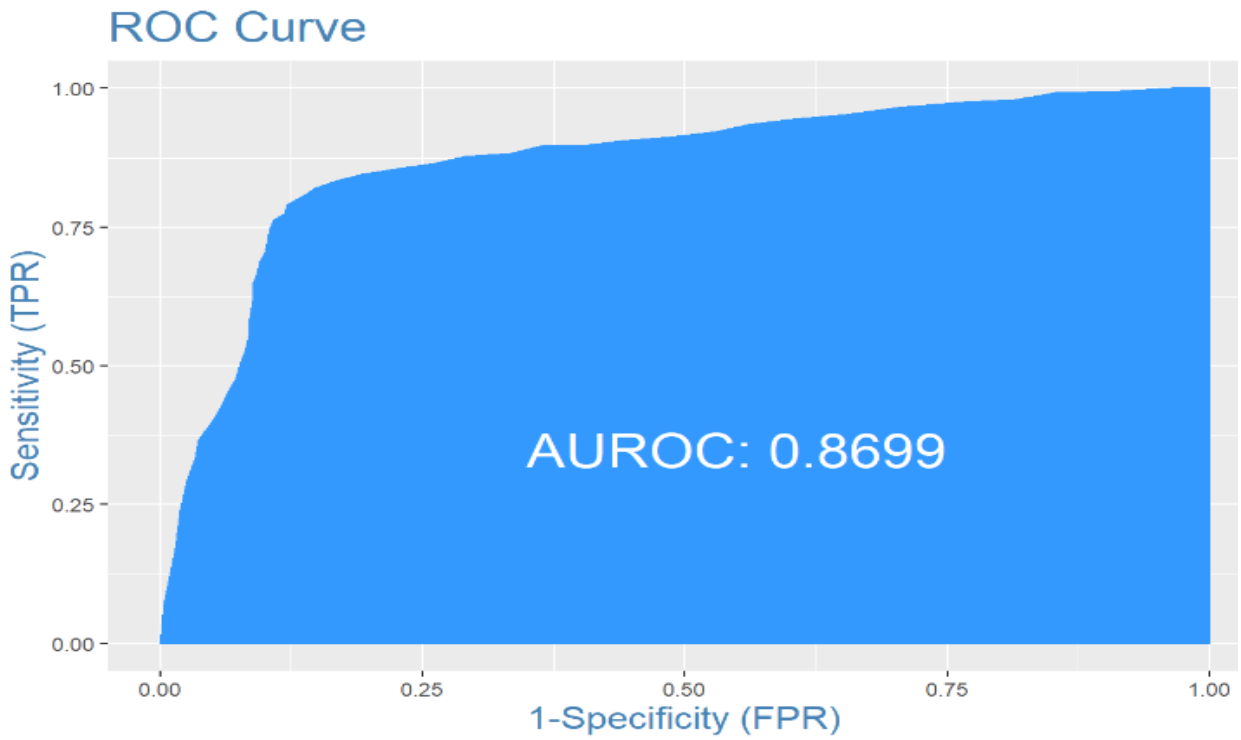
```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2810.4  on 2028  degrees of freedom
## Residual deviance: 1895.0  on 2016  degrees of freedom
## AIC: 1921
##
## Number of Fisher Scoring iterations: 6
```

- The above shows the summary of the Logistic regression model built for the Train set. All the variables are significant.
- The above coefficients are predicting whether or not customers have a '1' recorded (i.e., that they churned).
- The estimate of the intercept is -2.595, depicts that higher its value, lesser the chances of the customers to churn.
- Customer Renewal has a negative number -2.490, which means all else being equal, customers who recently renewed their contract have least chances to churn when compared to the customers who have not renewed their contract.
- Data Usage's estimate is -0.260, which means customers with good data usage have less chances to churn than those who are not.
- The estimate of DayMins is 0.012, which says customers who have high average minutes of day time calls are more likely to churn.
- The estimate of OverageFee is 0.152, says that customers having large overage fee are more likely to churn.
- The estimate of RoamMins is 0.112, meaning higher the value of the average roaming minutes of the customer, higher their chances to churn.

- The estimate of Call ‘0’ is – 0.581, which means when there is no calls to the customer service center, there are least chances of the customers to churn.
- The estimate of Call ‘1’ is -0.920, which says when there is ‘1’ call to the customer service center, there are least chances of the customers to churn.
- The estimate of Call ‘2’ is -0.563, which says when there are ‘2’ calls to the customer service center, there are less chances of the customers to churn.
- The estimate of Call ‘3’ is -0.967, which says when there are ‘3’ calls to the customer service center, there are less chances of the customers to churn.
- The estimate of Call ‘4’ is 1.850, which says when there are ‘4’ calls to the customer service center, there are high chances of the customers to churn.
- The estimate of Call ‘5’ is 3.518, which says when there are ‘5’ calls to the customer service center, there are high chances of the customers to churn.
- The estimate of Call ‘6’ is 3.592, which says when there are ‘6’ calls to the customer service center, there are high chances of the customers to churn.

CONFUSION MATRIX

```
##          Predicted
## Actual    0   1
##       0 396  73
##       1  68 332
```



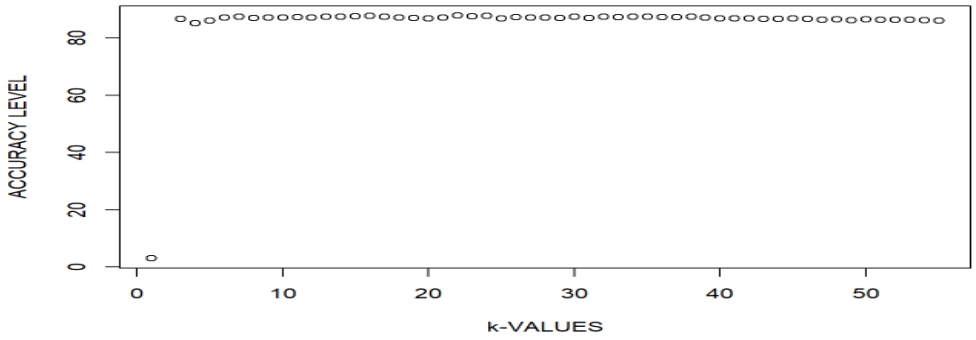
ROC CURVE FOR LOGISTIC REGRESSION MODEL

4. K – NEAREST NEIGHBOUR

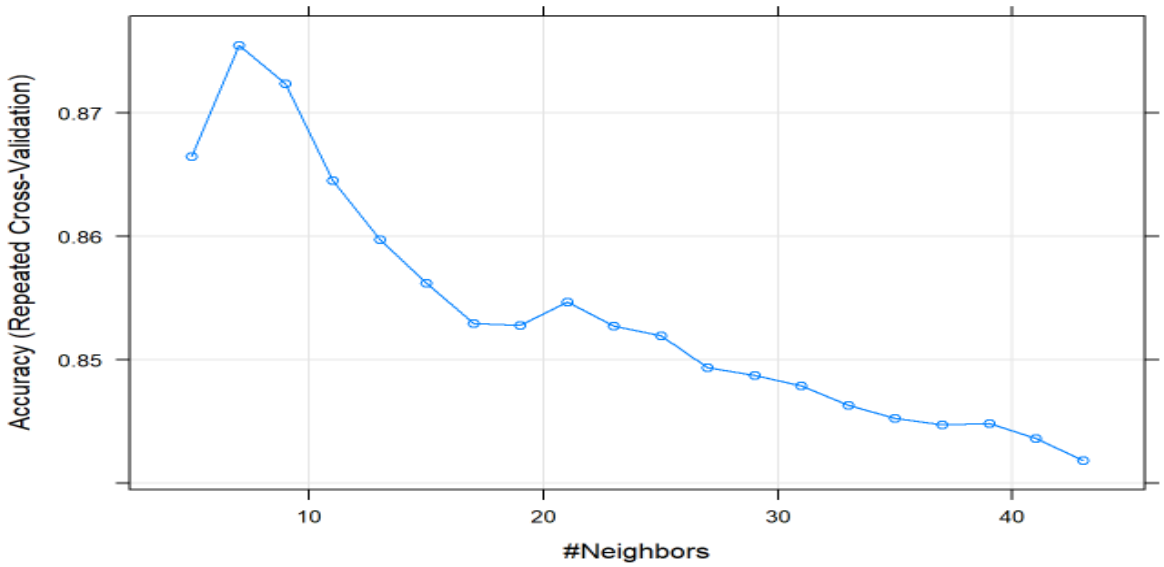
- KNN is a distance based algorithm (Euclidean), different variables with different scales are normalized before applying the algorithm.
- Though KNN works with both continuous and categorical variables, it demands little bit different approach for categorical predictors.
- KNN does not work with ordered factors, hence we perform one hot encoding for the categorical variables with different levels. Refer source code.

4.1 INTERPRETATION

- Using 10 repeats cross – validation the optimum value of ‘k’ is chosen as ‘7’ which is believed to provide a good accuracy level.
- The optimum value of ‘k’ is crucial for building a high performance model.



USING ‘FOR’ LOOP FOR DIFFERENT K VALUES AND ITS ACCURACY

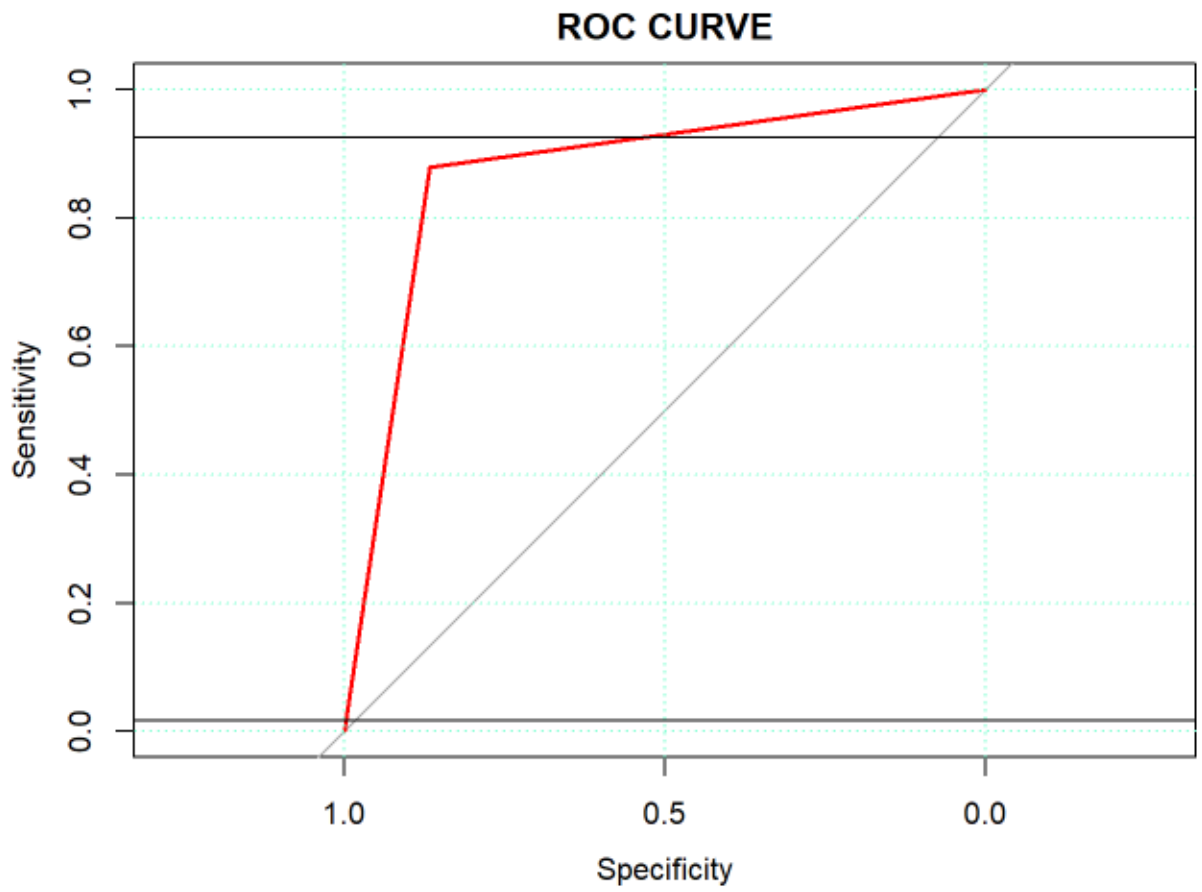


K VALUES AND ITS ACCURACY USING 10 REPEAT CROSS-VALIDATION

- Both the graphs more or less converge on optimum value of k as 7.

CONFUSION MATRIX

```
##         predicted
## actual    0   1
##      0  384  59
##      1   52 374
```



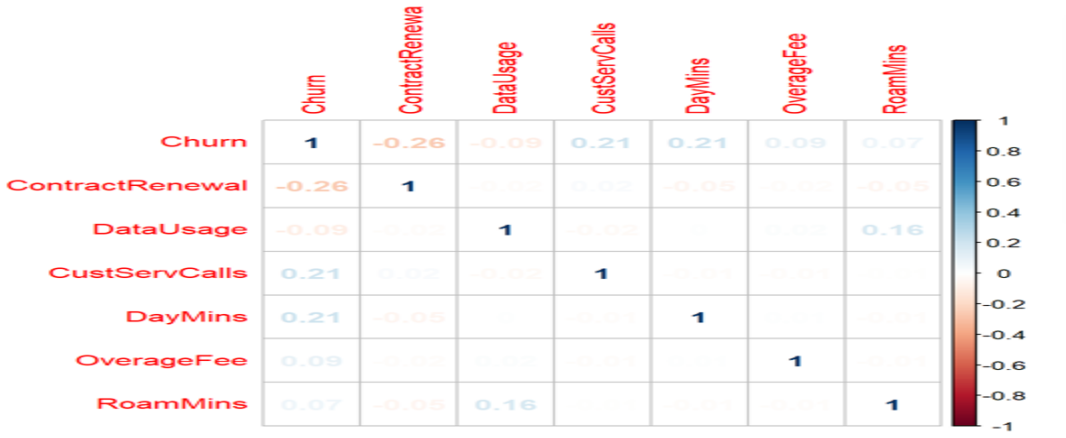
ROC CURVE FOR K – NEAREST NEIGHBOUR

5. NAÏVE BAYES

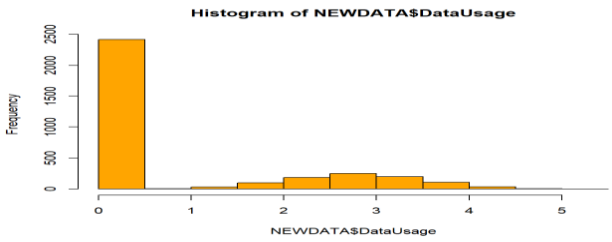
Naïve Bayes algorithm works based on Bayes Theorem for calculating probabilities and conditional probabilities.

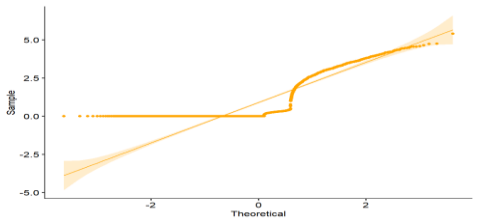
5.1 INTERPRETATION

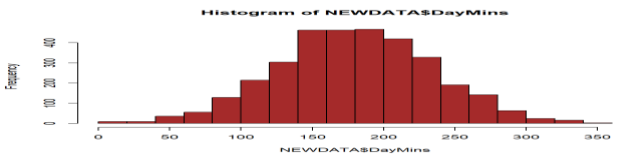
- It has a strong assumption of independence among the predictor variables. Multicollinearity shouldn't be a problem between the predictors.

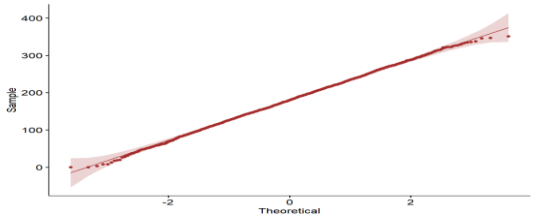


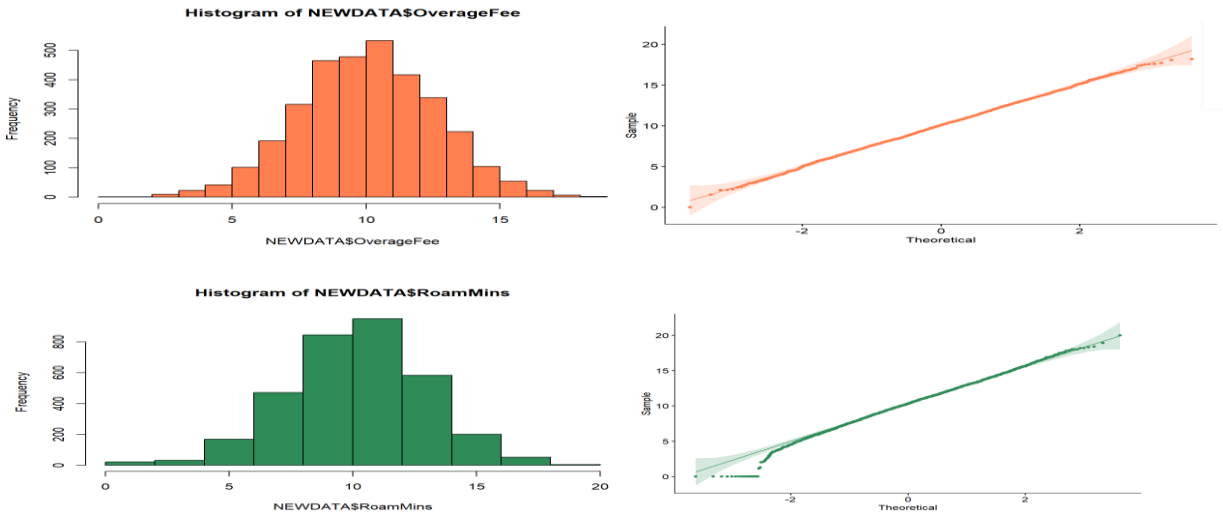
- The above correlation plot shows that all the variables used in the Naïve Bayes algorithm remains uncorrelated which satisfies its main assumption.
- The other important assumption of Naïve Bayes is that, all the continuous predictors should be normally distributed.
- The following graphs shows the normality of the continuous variables.



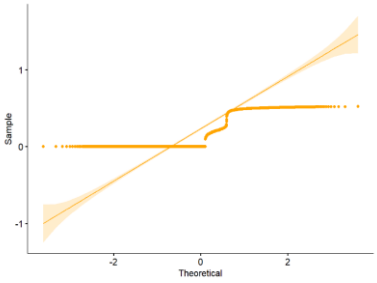


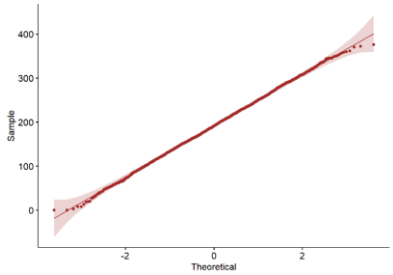


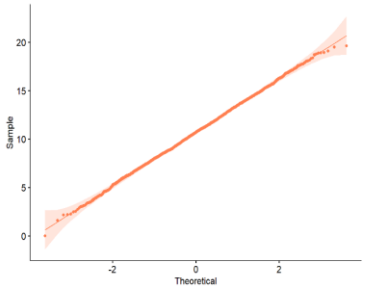


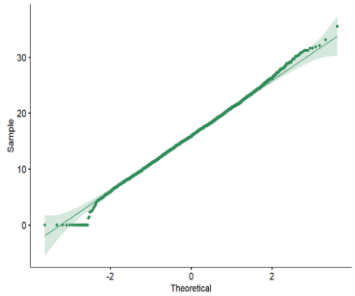


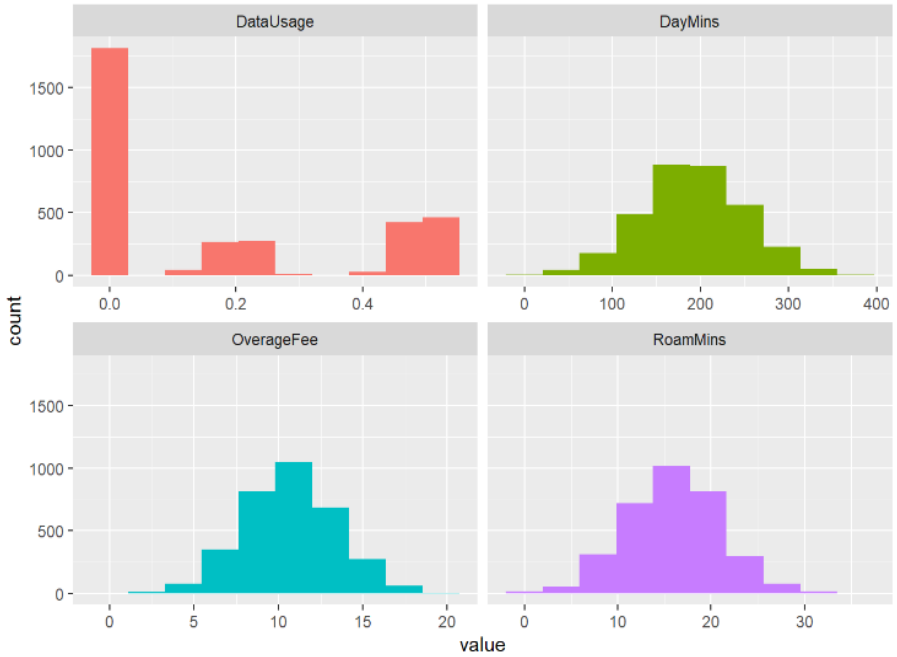
- From the above graphs we could clearly figure out that the predictors DataUsage and RoamMins are not normally distributed whereas DayMins and OverageFee are more or less nearing to the bell curve.
- In addition to the visual analysis, we also perform Statistical test methods such as Shapiro - Wilk's Normality test and Kolmogorov – Smirnov test on continuous predictors to check its normality. Refer source code.
- All the visuals and test proves to be identical depicting the variables DataUsage and RoamMins as not bell curves.
- Hence we try to transform the variables using BoxCox transformation or Yeo Johnson's method.
- Though the variables are not highly normal after the transformation, may be to some extent it can now comply with Naïve Bayes assumption.
- Thus using Naïve Bayes algorithm on TRAIN, we calculate probabilities for the TEST dataset and predict the chances of customer churning and not churning and thereby validate the same.







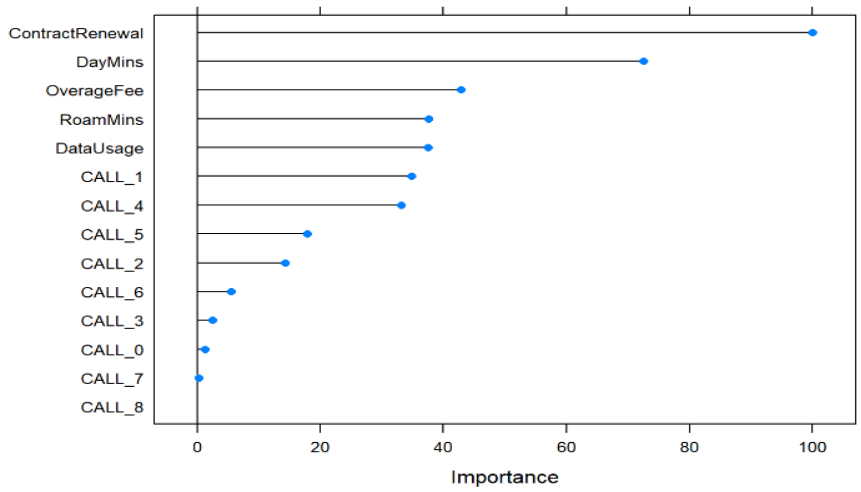


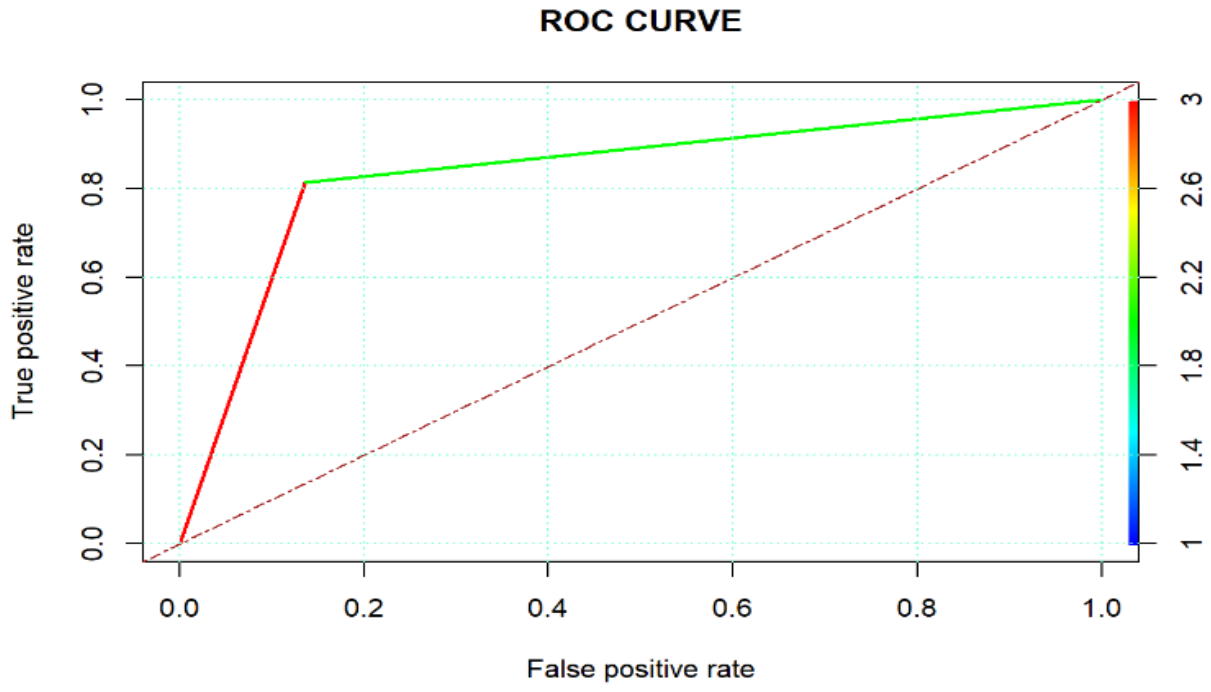


CONTINUOUS PREDICTORS AFTER TRANSFORMATION

CONFUSION MATRIX

```
##         predicted
## Actual    0   1
##      0  381  60
##      1   80 348
```





ROC CURVE FOR NAÏVE BAYES CLASSIFIER

6. CONFUSION MATRIX INTERPRETATION FOR THE MODELS

PERFORMANCE MEASURES	LOGISTIC REGRESSION	KNN	NAÏVE BAYES
ACCURACY	83.77 %	87.23 %	83.89 %
SENSITIVITY	83 %	87.79 %	81.31 %
SPECIFICITY	84.43 %	86.68 %	86.39 %
PRECISION	81.98 %	86.37 %	85.29 %
FI SCORE	82.48 %	87.08 %	83.25 %
DETECTION RATE	38 %	43 %	40 %
BALANCED ACCURACY	83.72 %	87.24 %	83.85

- The Accuracy of KNN is more when compared to the LOGIT model and Naïve Bayes classifier. The models have done a decent job of segregating the data points. Lesser the false predictions, higher the accuracy of the model.
- Sensitivity is the actual positive data points identified by the model as positive. KNN leads for the Sensitivity score, followed by LOGIT model and Naïve Bayes.
- The Specificity value of the KNN is the most. It is followed by Naïve Bayes in a small difference and then comes the LOGIT model. It is the actual negative data points identified by the model as negative.

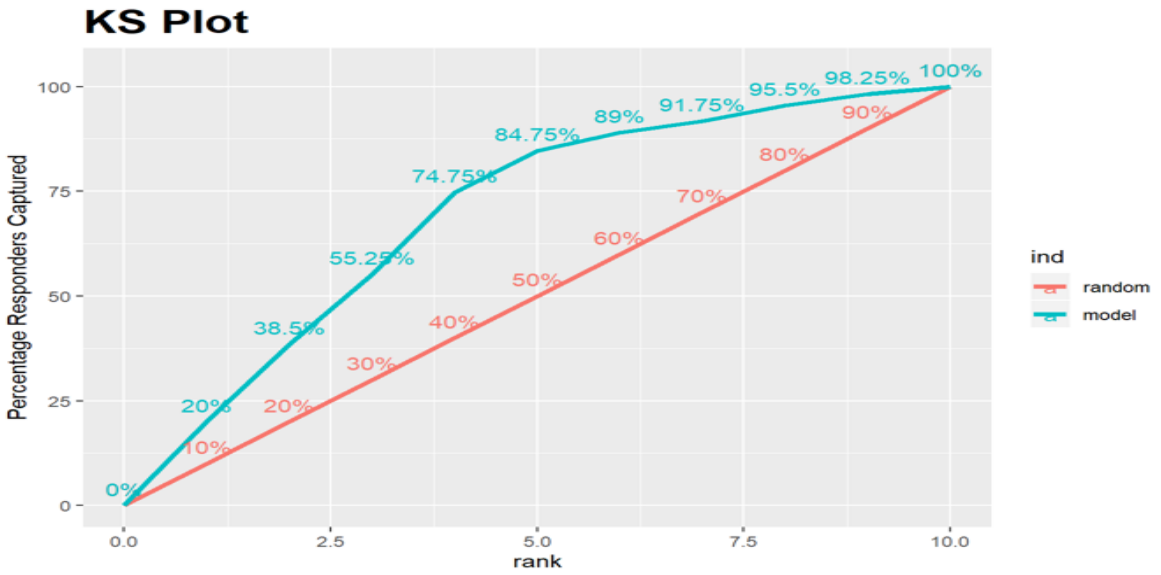
- Precision is the positive data points identified by the model which are really positive. Again KNN has best Precision value. Then comes the Naïve Bayes and finally the LOGIT model.
- The F1 score is also highest for KNN, followed by Naïve Bayes and then LOGIT model. It is the harmonic mean of Sensitivity and Precision.
- Type - 1 error of KNN is less when compared to the other two models. There is a very small degree of difference between the Type 1 errors of KNN and Naïve Bayes (only 1 data point).
- Type - 2 error of KNN is the least followed by LOGIT model and then Naïve Bayes.
- Even the Detection Rates and Balanced Accuracy of KNN is highest followed by Naïve Bayes and LOGIT model.
  Detection Rate = rate of accurately predicting the positive class as the proportion of total.
  Balanced Accuracy = $\frac{\text{Sensitivity + Specificity}}{2}$

7. OTHER PERFORMANCE MEASURES

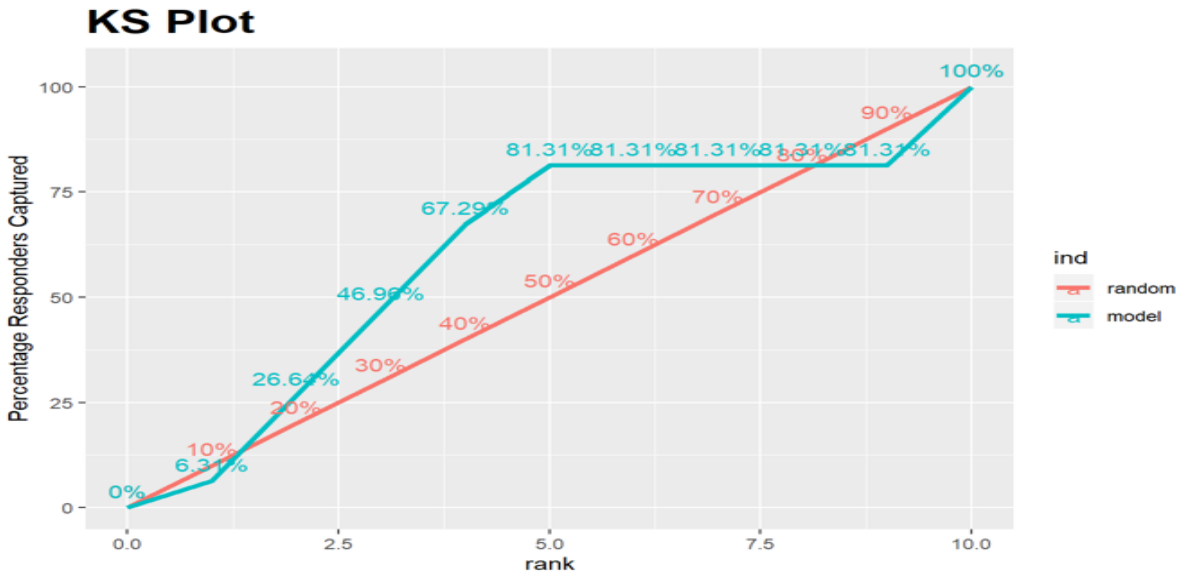
Few other performance measures are also calculated for all the three models such as Area under the Curve (AUC), Kolmogorov – Smirnov Goodness of Fit Test (K.S. statistic) and Gini Coefficient.

PERFORMANCE MEASURES	LOGISTIC REGRESSION	KNN	NAÏVE BAYES
AUC	86.99 %	93.83 %	83.85 %
K.S. STATISTIC	64.30 %	74.46 %	61.58 %
GINI COEFFICIENT	73.98 %	87.67 %	67.70 %

The following shows the K.S. statistic graph of Logistic Regression Model and Naïve Bayes Classifier.



LOGISTIC REGRESSION MODEL



NAÏVE BAYES MODEL

From the above table and graphs, we can strongly infer that KNN is performing best among the three models.

8. REMARKS ON MODEL VALIDATION

- With respect to all the performance metric measures discussed above, KNN proves to be the best model with high performance measure.
- Since our objective is to determine the Churn rate which is the positive class (1) for all the built models, our basic attention would be towards the metrics which measures the positive class such as Sensitivity, Precision, Detection rate and Balanced accuracy.
- Keeping that into account, both Logistic Regression and Naïve Bayes gives a relatively secondary performance when compared to KNN.
- With reference to other measures such as AUC, K.S. statistic and Gini coefficient, once again KNN has great values followed by Logistic Regression and Naïve Bayes Classifier.
- On summarizing all the above given performance metric values, we could strongly infer that KNN is the best optimum model. Both Naïve Bayes and Logistic Regression more or less gives a relatively secondary performance.

9. INFERENCES

INSIGHTS AND RECOMMENDATIONS

- The models are built on the balanced dataset, hence the results are not biased.
- With reference to the predictor variables we can infer that Customer who renew their contract, having a good data plan, best Customer service, low Overage fee, higher outgoing calls, an un-interrupted network and high roaming minutes at the minimum billing is less likely to churn.

- The above predictive modelling techniques can be used to identify customers who are more likely to churn and approaching them with suitable offers.
- We can further persuade them to renew their contract by providing additional data packs, extension of fixed minimum calls, valuable services like giving them a family plan on their current bill, improvised signal strength etc.,
- The biggest challenge of the Telecom Industry is to provide a good stable connection to the customer without incurring losses for themselves. But at the same time we should admit that it is less expensive to retain existing customers than to acquire new ones.

10. CONCLUSION

Thus three types of predictive modelling techniques such as LOGIT model, KNN and Naïve Bayes are modelled to predict the churn rate of the customer in a Telecom sector. Basically it is classification based supervised learning technique to predict the customer churn rate. The built models have classified the positive and negative classes on a balanced dataset (SMOTE technique applied).Hence necessary actions and recommendations are also provided for retention of the customers.

11. APPENDIX – A : SOURCE CODE

11.1 EXPLORATORY DATA ANALYSIS

```
setwd("C:/Training") ##### ENVIRONMENTAL SETUP

pacman::p_load(summarytools,lmtest,pscl,caret,MVN,hmeasure,mlbench,moments,rc
ompanion,bestNormalize,readxl,ggpubr,Hmisc,car,plotrix,Amelia,tidyverse,dplyr
,qgraph,corrplot,rsample,class,devtools,e1071,ggplot2,MASS,plyr,pROC,psych,dp
lyr,ROCR,InformationValue,e1071)

###### DATA IMPORT

read_xlsx("Cellphone (2).xlsX")->DATA
```

```
###### EXPLORATORY DATA ANALYSIS ######
dim(DATA)
## [1] 3333   11
names(DATA)
##  [1] "Churn"           "AccountWeeks"    "ContractRenewal" "DataPlan"
##  [5] "DataUsage"       "CustServCalls"   "DayMins"         "DayCalls"
##  [9] "MonthlyCharge"   "OverageFee"      "RoamMins"
str(DATA)
## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11 variables:
##  $ Churn          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
##  $ ContractRenewal: num  1 1 1 0 0 0 1 0 1 0 ...
##  $ DataPlan       : num  1 1 0 0 0 0 1 0 0 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
##  $ CustServCalls  : num  1 1 0 2 3 0 3 0 1 0 ...
##  $ DayMins        : num  265 162 243 299 167 ...
##  $ DayCalls       : num  110 123 114 71 113 98 88 79 97 84 ...
##  $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
class(DATA)
## [1] "tbl_df"     "tbl"        "data.frame"
DATA=as.data.frame(DATA)
summary(DATA)
##      Churn         AccountWeeks   ContractRenewal     DataPlan
##  Min.   :0.0000   Min.   :  1.0   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.: 74.0   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.0000   Median :101.0   Median :1.0000   Median :0.0000
##  Mean   :0.1449   Mean   :101.1   Mean   :0.9031   Mean   :0.2766
##  3rd Qu.:0.0000   3rd Qu.:127.0   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :243.0   Max.   :1.0000   Max.   :1.0000
##    DataUsage      CustServCalls      DayMins         DayCalls
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.0   Min.   :  0.0
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:143.7   1st Qu.: 87.0
```

```
##  Median :0.0000   Median :1.000   Median :179.4   Median :101.0
##  Mean   :0.8165   Mean   :1.563   Mean   :179.8   Mean   :100.4
##  3rd Qu.:1.7800   3rd Qu.:2.000   3rd Qu.:216.4   3rd Qu.:114.0
##  Max.   :5.4000   Max.   :9.000   Max.   :350.8   Max.   :165.0
##  MonthlyCharge      OverageFee       RoamMins
##  Min.   : 14.00   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 45.00   1st Qu.: 8.33   1st Qu.: 8.50
##  Median : 53.50   Median :10.07   Median :10.30
##  Mean   : 56.31   Mean   :10.05   Mean   :10.24
##  3rd Qu.: 66.20   3rd Qu.:11.77   3rd Qu.:12.10
##  Max.   :111.30   Max.   :18.19   Max.   :20.00
```

```
describe(DATA)
```

```
##                 vars    n   mean    sd median trimmed   mad min    max  ra
nge
## Churn              1 3333   0.14  0.35   0.00    0.06  0.00   0   1.00   1
.00
## AccountWeeks       2 3333 101.06 39.82 101.00  100.77 40.03   1 243.00 242
.00
## ContractRenewal    3 3333   0.90  0.30   1.00    1.00  0.00   0   1.00   1
.00
## DataPlan           4 3333   0.28  0.45   0.00    0.22  0.00   0   1.00   1
.00
## DataUsage          5 3333   0.82  1.27   0.00    0.58  0.00   0   5.40   5
.40
## CustServCalls      6 3333   1.56  1.32   1.00    1.42  1.48   0   9.00   9
.00
## DayMins            7 3333 179.78 54.47 179.40  179.85 53.82   0 350.80 350
.80
## DayCalls           8 3333 100.44 20.07 101.00  100.57 19.27   0 165.00 165
.00
## MonthlyCharge      9 3333  56.31 16.43  53.50   55.22 15.57  14 111.30  97
.30
## OverageFee        10 3333  10.05  2.54  10.07   10.05  2.55   0  18.19  18
.19
## RoamMins          11 3333  10.24  2.79  10.30   10.28  2.67   0  20.00  20
.00
##                  skew kurtosis   se
## Churn            2.02     2.07 0.01
## AccountWeeks     0.10    -0.11 0.69
```

```
## ContractRenewal -2.72     5.42 0.01
## DataPlan         1.00    -1.00 0.01
## DataUsage        1.27     0.04 0.02
## CustServCalls    1.09     1.72 0.02
## DayMins         -0.03    -0.02 0.94
## DayCalls        -0.11     0.24 0.35
## MonthlyCharge    0.59    -0.02 0.28
## OverageFee      -0.02     0.02 0.04
## RoamMins        -0.24     0.60 0.05
```

```
attach(DATA) ###### MISSING VALUES ######
anyNA(DATA)
```

```
## [1] FALSE
```

```
missmap(DATA)
###### OUTLIERS CHECK ######
boxplot.stats(DATA$Churn)$out
```

```
##   [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
##  [38] 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
##  [75] 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
## [112] 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
## [149] 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
## [186] 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
## [223] 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
## [260] 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
## [297] 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
## [334] 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
## [371] 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
## [408] 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
```

```
## [445] 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
## [482] 1 1
```

```
boxplot.stats(DATA$AccountWeeks)$out
```

```
##  [1] 208 215 209 224 243 217 210 212 232 225 225 224 212 210 217 209 221 2
09
```

```
boxplot.stats(DATA$ContractRenewal)$out
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
boxplot.stats(DATA$DataPlan)$out
```

```
## numeric(0)
```

```
boxplot.stats(DATA$DataUsage)$out
```

```
##  [1] 5.40 4.64 4.73 4.46 4.56 4.56 4.56 4.46 4.75 4.59 4.48
```

```
boxplot.stats(DATA$CustServCalls)$out
```

```
##   [1] 4 4 4 5 5 5 4 4 4 4 4 4 4 4 4 4 5 5 4 5 4 4 5 4 4 4 4 4 5 4 4 7 4 4
4 4 4
##  [38] 5 4 4 4 4 4 5 4 7 4 9 5 4 4 5 4 4 5 5 4 6 4 6 5 5 5 6 5 4 4 5 4 4 7
4 6 5
##  [75] 4 4 4 6 4 4 5 4 4 4 4 4 4 5 5 6 5 4 4 4 5 4 4 4 4 5 5 4 4 4 4 6 4 5
4 6 4
## [112] 4 4 4 4 4 4 4 4 6 4 4 4 4 8 4 4 5 4 4 4 6 5 5 7 4 4 5 4 4 5 4 4 5 7
4 4 5
## [149] 7 4 4 4 4 8 6 4 4 5 5 5 4 4 5 4 4 4 4 4 4 4 4 4 4 5 6 4 5 4 4 5 5 4
6 4 4
```

```
## [186] 4 9 6 4 5 5 4 6 4 4 5 4 4 4 5 5 6 4 5 4 4 4 4 5 4 4 4 5 4 5 6 4 4 5
4 4 4

## [223] 5 4 4 4 4 4 5 7 6 5 6 7 5 5 4 6 4 4 4 4 5 6 7 4 4 4 5 5 5 4 4 4 5 6
5 5 4

## [260] 4 4 4 4 4 4 4 5
```

```
boxplot.stats(DATA$DayMins)$out
```

```
##  [1] 332.9 337.4 326.5 350.8 335.5  30.9  34.0 334.3 346.8  12.5  25.9   0
.0

## [13]   0.0  19.5 329.8   7.9 328.1  27.0  17.6 326.3 345.3   2.6   7.8  18
.9

## [25]  29.9
```

```
boxplot.stats(DATA$DayCalls)$out
```

```
##  [1] 158 163  36  40 158 165  30  42   0  45   0  45 160 156  35  42 158 1
57  45

## [20]  44  44  44  40
```

```
boxplot.stats(DATA$MonthlyCharge)$out
```

```
##  [1] 110.0 104.3 102.9 101.4 101.8 100.3 102.6 108.3 105.6 101.6 110.0 104
.7

## [13] 100.5 101.2 102.5 102.1 103.9  98.6 108.7 103.5 100.3 108.6 111.3 101
.5

## [25] 102.1 103.8 101.6 103.1 104.9 105.2 106.9 102.6 100.6 100.0
```

```
boxplot.stats(DATA$OverageFee)$out
```

```
##  [1]  3.10 17.43 17.58  1.56 17.53  2.11 17.37  2.95  2.20  2.65  2.13  3.
04

## [13]  2.93  2.80  2.41  3.00 17.55  2.46 17.00 18.09 17.71 18.19  0.00 17.
07
```

```
boxplot.stats(DATA$RoamMins)$out
```

```
##  [1] 20.0  0.0 17.6  2.7 18.9  0.0 18.0  2.0  0.0 18.2  0.0  0.0  1.3  0.0
0.0

## [16]  0.0  2.2 18.0  0.0 17.9  0.0 18.4  2.0 17.8  2.9  3.1 17.6  2.6  0.0
0.0

## [31] 18.2  0.0 18.0  1.1  0.0 18.3  0.0  0.0  2.1  2.9  2.1  2.4  2.5  0.0
0.0

## [46] 17.8
```

```
###### UNIVARIATE ANALYSIS ######

funModeling::plot_num(DATA)

 ###### CUSTOMER CHURN RATE

 boxplot(Churn,col="blue")
```

```
plot(density(Churn,main="CHURN RATE"))
###### CUSTOMER'S ACTIVE WEEKS OF HAVING ACCOUNT
boxplot(AccountWeeks,col="viOlet")
plot(density(AccountWeeks,main="ACTIVE ACCOUNT WEEKS"))
###### CUSTOMER CONTRACT RENEWAL
 boxplot(ContractRenewal,col="coral")
 plot(density(ContractRenewal,main="CUSTOMER CONTRACT RENEWAL"))
###### DATA PLAN
boxplot(DataPlan,col="brown")
 plot(density(DataPlan,main="DATA PLAN"))
###### MONTHLY DATA USAGE
 boxplot(DataUsage,col="maroon")
plot(density(DataUsage,main="MONTHLY DATA USAGE"))
###### CUSTOMER SERVICE CALLS
 boxplot(CustServCalls,col="pink")
plot(density(CustServCalls,main="CUSTOMER SERVICE CALLS"))
###### AVERAGE DAY TIME MINUTES PER MONTH
boxplot(DayMins,col="Royal blue")
plot(density(DayMins,main="AVG DAY TIME MINS/MONTH"))
###### AVERAGE DAY TIME CALLS
boxplot(DayCalls,col="red")
plot(density(DayCalls,main="AVG DAY TIME CALLS"))
###### AVERAGE MONTHLY BILL
boxplot(MonthlyCharge,col="GREEN")
plot(density(MonthlyCharge,main="AVG MONTHLY BILL"))
###### LARGEST OVERAGE FEE
boxplot(OverageFee,col="YELLOW")
plot(density(OverageFee,main="LARGEST OVERAGE FEE IN LAST 12 MONTHS"))
###### AVERAGE ROAMING MINUTES
boxplot(RoamMins,col="ORANGE")
plot(density(RoamMins,main="AVERAGE ROAMING MINUTES"))
```

```
pie3D(prop.table((table(DATA$Churn))),labels=c(" NOT CHURNED","CHURNED"),heig
ht=0.05,explode = 0.1,main="PROPORTION OF CUSTOMER CHURN",col=c("#4286f4","#b
b3af2"))

pie3D(prop.table((table(DATA$ContractRenewal))),labels=c("NOT RENEWED","RENEW
ED"),explode = 0.1,main="PROPORTION OF CUSTOMER CONTRACT RENEWAL",col=c("turq
uoise"," medium sea green"))

pie3D(prop.table((table(DATA$DataPlan))),labels=c("NO PLAN","HAS PLAN"),explo
de = 0.1,main="CUSTOMER WITH OR WITHOUT DATAPLAN",col=c("coral","gray"))

 pie3D(prop.table((table(DATA$CustServCalls))),labels=c("0","1","2","3","4","
5","6","7","8","9"),explode = 0.1,main="PROPORTION OF CUSTOMER SERVICE CALLS"
,col=c("red","blue","green","yellow","sea green","maroon","pink","brown","gra
y","violet"))

###### BI-VARIATE ANALYSIS ######

funModeling::cross_plot(data = DATA,target = "Churn")

funModeling::correlation_table(DATA,"Churn")
```

```
##         Variable Churn
## 1          Churn  1.00
## 2 CustServCalls  0.21
## 3       DayMins  0.21
## 4    OverageFee  0.09
## 5 MonthlyCharge  0.07
## 6      RoamMins  0.07
## 7  AccountWeeks  0.02
## 8      DayCalls  0.02
## 9     DataUsage -0.09
```

```
# There is no significant correlation for the predictand with each predictors
.

# But there is evidence of multicollinearity within the predictors.

###### MULTICOLLINEARITY CHECK WITHIN THE PREDICTORS ######

###### LINEAR PROBABILITY MODEL ######

LPM<-lm(Churn~.,data = DATA)

summary(LPM)
```

```
##
## Call:
## lm(formula = Churn ~ ., data = DATA)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -0.66572 -0.16629 -0.08236  0.02060  1.08844 
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     -1.433e-01  5.363e-02  -2.672 0.007580 ** 
## AccountWeeks     8.888e-05  1.396e-04   0.637 0.524402    
## ContractRenewal -2.993e-01  1.882e-02 -15.904  < 2e-16 ***
## DataPlan        -4.175e-02  4.381e-02  -0.953 0.340650    
## DataUsage       -2.835e-02  1.933e-01  -0.147 0.883401    
## CustServCalls    5.829e-02  4.222e-03  13.804  < 2e-16 ***
## DayMins          1.021e-03  3.272e-03   0.312 0.754936    
## DayCalls         3.409e-04  2.769e-04   1.231 0.218433    
## MonthlyCharge    1.428e-03  1.924e-02   0.074 0.940838    
## OverageFee       1.046e-02  3.280e-02   0.319 0.749780    
## RoamMins         8.765e-03  2.307e-03   3.800 0.000147 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3203 on 3322 degrees of freedom
## Multiple R-squared:  0.1747, Adjusted R-squared:  0.1722 
## F-statistic: 70.31 on 10 and 3322 DF,  p-value: < 2.2e-16
```

```
vif(LPM)
```

```
##    AccountWeeks ContractRenewal        DataPlan       DataUsage   CustServ
Calls
##        1.003791        1.007216       12.473470     1964.800207        1.0
01945
##         DayMins        DayCalls   MonthlyCharge      OverageFee        Roa
mMins
##     1031.490608        1.002935     3243.300555      224.639750        1.3
46583
```

```
# The vif values shows existence of severe multicollinearity problems.Hence
we need to treat it inorder to build a robust model.
```

```r
# The highly correlated variables are DataPlan,DataUsage,DayMins,MonthlyCharge,OverageFee.

# Either we need to treat them or drop them from our analysis.


###### MULTICOLLINEARITY CHECK FOR THE PREDICTORS ######


###### HYPOTHESIS TESTING ######

# H0 = Multicollinearity does not exist within the predictors

# H1 = Multicollinearity exists within the predictors

(cor(DATA))->corr

qgraph(corr)

corrplot(corr,method = "number")

chart.Correlation(DATA,histogram = TRUE,pch="+",method = c("pearson","kendall","spearman"))

#### DROPPING INSIGNIFICANT VARIABLES

#ADJ.R2 SHOULD INCREASE OR STAY STABLE, WHEREAS RES.STD.ERROR SHOULD DECREASE IS THE CRITERION FOR INCLUDING/EXCLUDING EACH VARIABLE

#### MODEL WITH ALL THE VARIABLES

LPM1<-lm(Churn~ContractRenewal+DataPlan+DataUsage+CustServCalls+DayMins+DayCalls+MonthlyCharge+OverageFee+RoamMins,data = DATA)

summary(LPM1)
```

```
##
## Call:
## lm(formula = Churn ~ ContractRenewal + DataPlan + DataUsage +
##     CustServCalls + DayMins + DayCalls + MonthlyCharge + OverageFee +
##     RoamMins, data = DATA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66606 -0.16712 -0.08265  0.02047  1.08872
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.1344310  0.0517880  -2.596 0.009479 **
## ContractRenewal -0.2996236  0.0188161 -15.924  < 2e-16 ***
```

```
## DataPlan        -0.0426813  0.0437777  -0.975 0.329653
## DataUsage       -0.0258962  0.1932209  -0.134 0.893392
## CustServCalls    0.0582812  0.0042221  13.804  < 2e-16 ***
## DayMins          0.0010573  0.0032714   0.323 0.746557
## DayCalls         0.0003476  0.0002767   1.256 0.209094
## MonthlyCharge    0.0012184  0.0192351   0.063 0.949497
## OverageFee       0.0108092  0.0327929   0.330 0.741708
## RoamMins         0.0087476  0.0023062   3.793 0.000151 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3203 on 3323 degrees of freedom
## Multiple R-squared:  0.1746, Adjusted R-squared:  0.1724
## F-statistic:  78.1 on 9 and 3323 DF,  p-value: < 2.2e-16
```

```
vif(LPM1)
```

```
## ContractRenewal        DataPlan       DataUsage   CustServCalls         Da
yMins
##        1.006691       12.459502     1964.020063        1.001940     1031.1
84395
##        DayCalls   MonthlyCharge      OverageFee        RoamMins
##        1.001470     3242.351013      224.577593        1.346401
```

```
### VIF VALUES OF CERTAIN PREDICTORS ARE TOO HIGH FOR THE MODEL,WHICH HAS ALL
THE GIVEN VARIABLES,HENCE WE DROP THE CORRELATED VARIABLES AND RE-CHECK THE
MODEL WITH ONLY THE SIGNIFICANT PREDICTORS.
```

```
LPM2<-lm(Churn~ContractRenewal+DataUsage+CustServCalls+DayMins+OverageFee+Roa
mMins,data = DATA)
summary(LPM2)
```

```
##
## Call:
## lm(formula = Churn ~ ContractRenewal + DataUsage + CustServCalls +
##     DayMins + OverageFee + RoamMins, data = DATA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66425 -0.16807 -0.08194  0.02045  1.08471
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.109891   0.041161  -2.670  0.00763 **
## ContractRenewal -0.299977   0.018805 -15.952  < 2e-16 ***
## DataUsage       -0.028362   0.004421  -6.416 1.60e-10 ***
## CustServCalls    0.058154   0.004221  13.778  < 2e-16 ***
## DayMins          0.001267   0.000102  12.425  < 2e-16 ***
## OverageFee       0.012817   0.002189   5.854 5.26e-09 ***
## RoamMins         0.009895   0.002017   4.907 9.70e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3203 on 3326 degrees of freedom
## Multiple R-squared:  0.174,  Adjusted R-squared:  0.1725
## F-statistic: 116.7 on 6 and 3326 DF,  p-value: < 2.2e-16
```

```
vif(LPM5)
```

```
## ContractRenewal       DataUsage   CustServCalls         DayMins       Overa
geFee
##        1.005646        1.028250        1.001378        1.002803        1.0
01151
##        RoamMins
##        1.029580
### NOW THE CHOSEN PREDICTORS SEEM TO BE SIGNIFICANT WITH OPTIMAL VIF VALUES
WHICH SHOWS NO CORRELATION BETWEEN THEM.
```

```
###### CROSS- CHECKING VARIABLE SELECTION WITH SUBSET SELECTION ALGORITHM
###### VARIABLE SELECTION : SUBSET COMPARISON
###### BEST SUBSET SELECTION
library(leaps)
regsubsets(Churn~.,data = DATA)->MODEL
summary(MODEL)->fit
fit
```

```
## Subset selection object
## Call: regsubsets.formula(Churn ~ ., data = DATA)
## 10 Variables  (and intercept)
```

```
##                 Forced in Forced out
## AccountWeeks        FALSE      FALSE
## ContractRenewal     FALSE      FALSE
## DataPlan            FALSE      FALSE
## DataUsage           FALSE      FALSE
## CustServCalls       FALSE      FALSE
## DayMins             FALSE      FALSE
## DayCalls            FALSE      FALSE
## MonthlyCharge       FALSE      FALSE
## OverageFee          FALSE      FALSE
## RoamMins            FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls Day
Mins
## 1  ( 1 ) " "          "*"             " "      " "       " "           " "
## 2  ( 1 ) " "          "*"             " "      " "       "*"           " "
## 3  ( 1 ) " "          "*"             " "      " "       "*"           "*"
## 4  ( 1 ) " "          "*"             " "      "*"       "*"           " "
## 5  ( 1 ) " "          "*"             " "      "*"       "*"           " "
## 6  ( 1 ) " "          "*"             " "      "*"       "*"           " "
## 7  ( 1 ) " "          "*"             "*"      "*"       "*"           " "
## 8  ( 1 ) "*"          "*"             "*"      "*"       "*"           " "
##          DayCalls MonthlyCharge OverageFee RoamMins
## 1  ( 1 ) " "      " "           " "        " "
## 2  ( 1 ) " "      " "           " "        " "
## 3  ( 1 ) " "      " "           " "        " "
## 4  ( 1 ) " "      "*"           " "        " "
## 5  ( 1 ) " "      "*"           " "        "*"
## 6  ( 1 ) "*"      "*"           " "        "*"
## 7  ( 1 ) "*"      "*"           " "        "*"
## 8  ( 1 ) "*"      "*"           " "        "*"
```

```
fit$rss->RSS
fit$adjr2->AdjR2
```

```
cbind(RSS,AdjR2)->TABLE
TABLE
```

```
##           RSS      AdjR2
## [1,] 385.1189 0.06724304
## [2,] 365.9945 0.11329610
## [3,] 350.2582 0.15116584
## [4,] 343.6394 0.16695606
## [5,] 341.1699 0.17269386
## [6,] 341.0087 0.17283613
## [7,] 340.9125 0.17282091
## [8,] 340.8703 0.17267450
```

```
# PLOTTING
par(mfrow=c(1,1))
plot(TABLE,col="red")
plot(TABLE,col="brown",type = "line")
# From the selection algorithm,the best set of variables are selected.A model is best only if its adjusted R2 is maximised and the prediction error is minimised(rss,cp,bic).
data.frame(Adj.R2=which.max(fit$adjr2),CP=which.min(fit$cp),BIC=which.min(fit$bic))
```

```
##   Adj.R2 CP BIC
## 1      6  5   5
```

```
# FITTING THE MODEL WITH BEST VARIABLES(FROM 5TH MODEL IN SELECTION ALGO)
lm(Churn~ContractRenewal+DataUsage+CustServCalls+MonthlyCharge+RoamMins,data=DATA)->best_model
summary(best_model)
```

```
##
## Call:
## lm(formula = Churn ~ ContractRenewal + DataUsage + CustServCalls +
##     MonthlyCharge + RoamMins, data = DATA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66451 -0.16706 -0.08237  0.01922  1.08093
```

```
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.1124513  0.0394709  -2.849  0.00441 **
## ContractRenewal -0.3001751  0.0188016 -15.965  < 2e-16 ***
## DataUsage       -0.1030067  0.0070392 -14.633  < 2e-16 ***
## CustServCalls    0.0581732  0.0042200  13.785  < 2e-16 ***
## MonthlyCharge    0.0074656  0.0005424  13.764  < 2e-16 ***
## RoamMins         0.0098940  0.0020162   4.907 9.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3202 on 3327 degrees of freedom
## Multiple R-squared:  0.1739, Adjusted R-squared:  0.1727
## F-statistic: 140.1 on 5 and 3327 DF,  p-value: < 2.2e-16
```

```
vif(best_model)
```

```
## ContractRenewal       DataUsage    CustServCalls    MonthlyCharge         Roa
mMins
##        1.005550        2.607736        1.001356        2.579109        1.0
29509
```

```
# MORE OR LESS THE SUBSET ALGORITHM SHOWS THE SAME PREDICTORS WE CHOSE AS THE
SIGNIFICANT ONES IN PREDICTING CHURNING RATE OF THE CUSTOMERS.

# CREATING DATAFRAME WITH ONLY THE SIGNIFICANT PREDICTORS

NEWDATA<-DATA[,-c(2,4,8,9)]

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

NEWDATA$ContractRenewal=as.factor(NEWDATA$ContractRenewal)

## CREATING DUMMIES FOR CUSTOMER SERVICE CALLS

NEWDATA$CALL_0<-ifelse(NEWDATA$CustServCalls=="0",1,0)

NEWDATA$CALL_1<-ifelse(NEWDATA$CustServCalls=="1",1,0)

NEWDATA$CALL_2<-ifelse(NEWDATA$CustServCalls=="2",1,0)

NEWDATA$CALL_3<-ifelse(NEWDATA$CustServCalls=="3",1,0)

NEWDATA$CALL_4<-ifelse(NEWDATA$CustServCalls=="4",1,0)

NEWDATA$CALL_5<-ifelse(NEWDATA$CustServCalls=="5",1,0)

NEWDATA$CALL_6<-ifelse(NEWDATA$CustServCalls=="6",1,0)

NEWDATA$CALL_7<-ifelse(NEWDATA$CustServCalls=="7",1,0)
```

```
NEWDATA$CALL_8<-ifelse(NEWDATA$CustServCalls=="8",1,0)
NEWDATA<-NEWDATA[-4]
NEWDATA$CALL_0=as.factor(NEWDATA$CALL_0)
NEWDATA$CALL_1=as.factor(NEWDATA$CALL_1)
NEWDATA$CALL_2=as.factor(NEWDATA$CALL_2)
NEWDATA$CALL_3=as.factor(NEWDATA$CALL_3)
NEWDATA$CALL_4=as.factor(NEWDATA$CALL_4)
NEWDATA$CALL_5=as.factor(NEWDATA$CALL_5)
NEWDATA$CALL_6=as.factor(NEWDATA$CALL_6)
NEWDATA$CALL_7=as.factor(NEWDATA$CALL_7)
NEWDATA$CALL_8=as.factor(NEWDATA$CALL_8)
str(NEWDATA)
```

```
## 'data.frame':    3333 obs. of  15 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
##  $ DayMins        : num  265 162 243 299 167 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ CALL_0         : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 1 2 ...
##  $ CALL_1         : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 2 1 ...
##  $ CALL_2         : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ CALL_3         : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...
##  $ CALL_4         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_5         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_6         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_7         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_8         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
###### SPLITTING THE DATASET INTO TEST AND TRAIN DATA
table(NEWDATA$Churn)
```

```
##
##    0    1
## 2850  483
```

```
prop.table(table(NEWDATA$Churn))
```

```
##
##         0         1
## 0.8550855 0.1449145
```

```
#### THE DATASET IS HIGHLY IMBALANCED.HENCE PREDICTIONS WITH IMABALANCED DATA
WILL BE A BIASED ONE.

# BALANCING THE DATASET USING SMOTE

library(DMwR)

DATA.BAL<-SMOTE(Churn~.,NEWDATA,perc.over =250 ,perc.under = 150)

table(DATA.BAL$Churn)
```

```
##
##    0    1
## 1449 1449
```

```
prop.table(table(DATA.BAL$Churn))
```

```
##
##   0   1
## 0.5 0.5
```

```
pie3D(prop.table((table(DATA.BAL$Churn))),labels=c(" NOT CHURNED","CHURNED"),
height=0.05,explode = 0.1,main="PROPORTION OF CUSTOMER CHURN",col=c("#4286f4"
,"#bb3af2"))
```

11.2 LOGISTIC REGRESSION

```
set.seed(1234)


SPLIT_DATA=initial_split(DATA.BAL,prop = 0.70)

TRAIN=training(SPLIT_DATA)

TEST=testing(SPLIT_DATA)


table(TRAIN$Churn)
```

```
##
```

```
##    0    1
##  980 1049
```

```
table(TEST$Churn)
```

```
##
##   0   1
## 469 400
```

```
prop.table(table(TRAIN$Churn))
```

```
##
##         0         1
## 0.4829966 0.5170034
```

```
prop.table(table(TEST$Churn))
```

```
##
##         0         1
## 0.5397008 0.4602992
```

```
TRAIN=TRAIN[,-c(14,15)]


###### BUILDING LOGISTIC REGRESSION MODEL ON TRAIN SET
LOGMODEL<-glm(Churn~.,data = TRAIN,family = binomial())


# STEP:1 DATA VALIDITY OF LOGISTIC REGRESSION MODEL


lmtest::lrtest(LOGMODEL)
```

```
## Likelihood ratio test
##
## Model 1: Churn ~ ContractRenewal + DataUsage + DayMins + OverageFee +
##     RoamMins + CALL_0 + CALL_1 + CALL_2 + CALL_3 + CALL_4 + CALL_5 +
##     CALL_6
## Model 2: Churn ~ 1
##   #Df   LogLik  Df  Chisq Pr(>Chisq)
## 1  13  -947.51
## 2   1 -1405.22 -12 915.43  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# STEP:2 MC FADEN TEST FOR R2

pscl::pR2(LOGMODEL)
## fitting null model for pseudo-r2
##           llh       llhNull            G2      McFadden          r2ML
##  -947.5059799 -1405.2221651   915.4323704     0.3257251     0.3631201
##          r2CU
##     0.4843470

# STEP:3 TEST FOR INDIVIDUAL COEFFICIENTS(BETAS)

summary(LOGMODEL)
##
## Call:
## glm(formula = Churn ~ ., family = binomial(), data = TRAIN)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1974  -0.7581   0.1233   0.7290   2.5829
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.59502    0.42481  -6.109 1.00e-09 ***
## ContractRenewal1 -2.49037    0.15710 -15.852  < 2e-16 ***
## DataUsage        -0.26025    0.04821  -5.398 6.72e-08 ***
## DayMins           0.01270    0.00106  11.976  < 2e-16 ***
## OverageFee        0.15222    0.02441   6.235 4.51e-10 ***
## RoamMins          0.11210    0.02183   5.135 2.82e-07 ***
## CALL_01          -0.58117    0.17712  -3.281  0.00103 **
## CALL_11          -0.92070    0.17064  -5.395 6.84e-08 ***
## CALL_21          -0.56376    0.17566  -3.209  0.00133 **
## CALL_31          -0.96765    0.21134  -4.579 4.68e-06 ***
## CALL_41           1.85016    0.25770   7.180 6.99e-13 ***
## CALL_51           3.51849    0.54192   6.493 8.44e-11 ***
```

```
## CALL_61           3.59288    0.76207   4.715 2.42e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2810.4  on 2028  degrees of freedom
## Residual deviance: 1895.0  on 2016  degrees of freedom
## AIC: 1921
##
## Number of Fisher Scoring iterations: 6
```

```
# STEP:4 EXPLANATORY POWER OF ODDS AND INTERPRETATION OF IT


exp(coef(LOGMODEL))
```

```
##      (Intercept) ContractRenewal1         DataUsage          DayMins
##       0.07464466       0.08287951       0.77086177       1.01278195
##       OverageFee         RoamMins          CALL_01          CALL_11
##       1.16441929       1.11862844       0.55924172       0.39823860
##          CALL_21          CALL_31          CALL_41          CALL_51
##       0.56906698       0.37997633       6.36084194      33.73339386
##          CALL_61
##      36.33871184
```

```
# PROBABILITY


exp(coef(LOGMODEL))/1+exp(coef(LOGMODEL))
```

```
##      (Intercept) ContractRenewal1         DataUsage          DayMins
##        0.1492893        0.1657590        1.5417235        2.0255639
##       OverageFee         RoamMins          CALL_01          CALL_11
##        2.3288386        2.2372569        1.1184834        0.7964772
##          CALL_21          CALL_31          CALL_41          CALL_51
##        1.1381340        0.7599527       12.7216839       67.4667877
##          CALL_61
##       72.6774237
```

```
# ODDS -- ANY VALUE GREATER THAN 1 ,THEN THE VARIABLE IS IMPORTANT FOR THE MO
DEL


vif(LOGMODEL)
```

```
## ContractRenewal       DataUsage         DayMins      OverageFee         Roa
mMins
##        1.047273        1.042972        1.114453        1.031511        1.0
34417
##          CALL_0          CALL_1          CALL_2          CALL_3          C
ALL_4
##        1.746227        1.939477        1.698458        1.505777        1.2
46900
##          CALL_5          CALL_6
##        1.052541        1.026052
```

```
varImp(LOGMODEL)->IMP

IMP
```

```
##                     Overall
## ContractRenewal1 15.851915
## DataUsage         5.398422
## DayMins          11.976416
## OverageFee        6.235441
## RoamMins          5.135047
## CALL_01           3.281301
## CALL_11           5.395433
## CALL_21           3.209289
## CALL_31           4.578593
## CALL_41           7.179597
## CALL_51           6.492627
## CALL_61           4.714618
```

```
plot(IMP)
###### PREDICTIONS FOR TEST SET
PRED.LOGMODEL<- predict(LOGMODEL,newdata=TEST,type = "response")
TEST$PRED<-PRED.LOGMODEL
optcutoff<-optimalCutoff(TEST$Churn,PRED.LOGMODEL)
optcutoff
```

```
## [1] 0.4497596
```

```
cutoff=floor(PRED.LOGMODEL>=optcutoff)
TEST$PREDCLASS<-cutoff
TEST$PREDCLASS=as.factor(TEST$PREDCLASS)
str(TEST)
```

```
## 'data.frame':    869 obs. of  17 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...
##  $ DataUsage      : num  0 0 0 0 0 2.78 3.11 0 1.65 0 ...
##  $ DayMins        : num  210 197 140 182 161 ...
##  $ OverageFee     : num  10.23 13.47 6.21 9.93 11.03 ...
##  $ RoamMins       : num  8.5 10.8 15 9.3 5.1 10.3 11.5 9.4 6.1 11.6 ...
##  $ CALL_0         : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ CALL_1         : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 2 2 2 2 ...
##  $ CALL_2         : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 1 1 ...
##  $ CALL_3         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_4         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_5         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_6         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_7         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_8         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ PRED           : num  0.305 0.44 0.223 0.313 0.203 ...
##  $ PREDCLASS      : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 2 1 1 ...
```

```
###### PERFORMANCE MEASURE METRICS
###### CONFUSION MATRIX
caret::confusionMatrix(TEST$PREDCLASS,TEST$Churn,positive="1",mode="everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 396  68
##          1  73 332
##
##                Accuracy : 0.8377
```

```
##                  95% CI : (0.8115, 0.8617)
##     No Information Rate : 0.5397
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6737
##
##  Mcnemar's Test P-Value : 0.7362
##
##             Sensitivity : 0.8300
##             Specificity : 0.8443
##          Pos Pred Value : 0.8198
##          Neg Pred Value : 0.8534
##               Precision : 0.8198
##                  Recall : 0.8300
##                      F1 : 0.8248
##              Prevalence : 0.4603
##          Detection Rate : 0.3820
##    Detection Prevalence : 0.4661
##       Balanced Accuracy : 0.8372
##
##        'Positive' Class : 1
##
table(Actual=TEST$Churn,Predicted=cutoff)
##       Predicted
## Actual   0   1
##      0 396  73
##      1  68 332
###### MISCLASSIFICATION ERROR
misClassError(TEST$Churn,cutoff,threshold =optcutoff )
## [1] 0.1623



###### RECEIVER OPEARATOR CHARACTERISTIC CURVE (ROC)
ROCRpred<-prediction(PRED.LOGMODEL,TEST$Churn)
```

```
ROCRperf<-performance(ROCRpred,"tpr","fpr")
plot(ROCRperf,colorize=TRUE,lwd=2,main="ROC CURVE",text.adj=c(-0.2,1.7))
box()
abline(0,1,lty=300,col="brown")
grid(col = "aquamarine")
InformationValue::plotROC(TEST$Churn,PRED.LOGMODEL)
InformationValue::AUROC(TEST$Churn,PRED.LOGMODEL)->AUC
AUC
```

```
## [1] 0.869928
```

```
###### GINI COEFFICIENT
GINI<-(2*AUC)-1
GINI
```

```
## [1] 0.7398561
```

```
###### K STATISTIC
ks_stat(TEST$Churn,PRED.LOGMODEL)
```

```
## [1] 0.643
```

```
ks_stat(TEST$Churn,PRED.LOGMODEL,returnKSTable = TRUE)
```

```
##    rank total_pop non_responders responders expected_responders_by_random
## 1     1        87              7         80                       40.04603
## 2     2        87             13         74                       40.04603
## 3     3        87             20         67                       40.04603
## 4     4        87              9         78                       40.04603
## 5     5        87             47         40                       40.04603
## 6     6        87             70         17                       40.04603
## 7     7        87             76         11                       40.04603
## 8     8        87             72         15                       40.04603
## 9     9        87             76         11                       40.04603
## 10   10        86             79          7                       39.58573
##    perc_responders perc_non_responders cum_perc_responders
## 1           0.2000          0.01492537              0.2000
## 2           0.1850          0.02771855              0.3850
## 3           0.1675          0.04264392              0.5525
## 4           0.1950          0.01918977              0.7475
```

```
## 5           0.1000          0.10021322              0.8475
## 6           0.0425          0.14925373              0.8900
## 7           0.0275          0.16204691              0.9175
## 8           0.0375          0.15351812              0.9550
## 9           0.0275          0.16204691              0.9825
## 10          0.0175          0.16844350              1.0000
##    cum_perc_non_responders difference
## 1               0.01492537  0.1850746
## 2               0.04264392  0.3423561
## 3               0.08528785  0.4672122
## 4               0.10447761  0.6430224
## 5               0.20469083  0.6428092
## 6               0.35394456  0.5360554
## 7               0.51599147  0.4015085
## 8               0.66950959  0.2854904
## 9               0.83155650  0.1509435
## 10              1.00000000  0.0000000
```

```
ks_plot(TEST$Churn,PRED.LOGMODEL)
###### GAINS TABLE
###### TO CALCULATE GAINS THE ORIGINAL TARGET SHOULD BE IN NUMERIC FORMAT
TEST$Churn=as.numeric(TEST$Churn)
library(gains)
GAINS<-gains(actual = TEST$Churn,predicted = PRED.LOGMODEL,groups = 10)
print(GAINS)
```

```
## Depth                                Cume   Cume Pct                      Mean
##  of           Cume     Mean       Mean   of Total    Lift   Cume      Model
## File     N      N      Resp       Resp      Resp    Index   Lift      Score
## ------------------------------------------------------------------------------
##   10    86     86      1.92       1.92      13.0%     131    131       0.98
##   20    87    173      1.85       1.88      25.7%     127    129       0.92
##   30    87    260      1.78       1.85      37.9%     122    127       0.79
##   40    87    347      1.89       1.86      50.8%     129    127       0.64
##   50    87    434      1.47       1.78      60.9%     101    122       0.48
##   60    87    521      1.20       1.68      69.1%      82    115       0.37
```

```
##   70    88    609      1.12      1.60      76.9%      77    110      0.29
##   80    86    695      1.17      1.55      84.9%      80    106      0.21
##   90    87    782      1.11      1.50      92.5%      76    103      0.15
##  100    87    869      1.09      1.46     100.0%      75    100      0.08
```

11.3 KNN

```
set.seed(15)
SPLIT_DATA=initial_split(DATA.BAL,prop = 0.70)
TRAIN=training(SPLIT_DATA)
TEST=testing(SPLIT_DATA)


table(TRAIN$Churn)
##
##    0    1
## 1006 1023
table(TEST$Churn)
##
##   0   1
## 443 426
prop.table(table(TRAIN$Churn))
##
##         0         1
## 0.4958107 0.5041893
prop.table(table(TEST$Churn))
##
##         0         1
## 0.5097814 0.4902186
###### NORMALIZATION OF THE PREDICTANDS
normalize<-function(x){
  return((x-min(x))/(max(x)-min(x)))}


NORMALIZED_TRAIN<-as.data.frame(lapply(TRAIN[,2:16],normalize))
```

```
str(NORMALIZED_TRAIN)
```

```
## 'data.frame':    2029 obs. of  15 variables:
##  $ ContractRenewal: num  0 1 1 1 1 1 1 1 1 1 ...
##  $ DataUsage      : num  0 0 0.0421 0 0 ...
##  $ CustServCalls  : num  0.111 0.333 0 0.444 0.222 ...
##  $ DayMins        : num  0.401 0.369 0.457 0.645 0.574 ...
##  $ OverageFee     : num  0.628 0.393 0.609 0.613 0.625 ...
##  $ RoamMins       : num  0.59 0.666 0.606 0.704 0.455 ...
##  $ CALL_0         : num  0 0 1 0 0 1 1 0 1 0 ...
##  $ CALL_1         : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_2         : num  0 0 0 0 1 0 0 1 0 1 ...
##  $ CALL_3         : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ CALL_4         : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ CALL_5         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_6         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_7         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_8         : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
NORMALIZED_TEST<-as.data.frame(lapply(TEST[,2:16],normalize))
str(NORMALIZED_TEST)
```

```
## 'data.frame':    869 obs. of  15 variables:
##  $ ContractRenewal: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ DataUsage      : num  0.7296 0 0 0 0.0556 ...
##  $ CustServCalls  : num  0.222 0.667 0 0.333 0.111 ...
##  $ DayMins        : num  0.516 0.377 0.465 0.426 0.274 ...
##  $ OverageFee     : num  0.702 0.732 0.453 0.413 0.404 ...
##  $ RoamMins       : num  0.73 0.545 0.335 0.495 0.485 0.53 0.565 0.5 0.32
0.445 ...
##  $ CALL_0         : num  0 0 1 0 0 0 1 0 0 0 ...
##  $ CALL_1         : num  0 0 0 0 1 1 0 0 1 1 ...
##  $ CALL_2         : num  1 0 0 0 0 0 0 1 0 0 ...
##  $ CALL_3         : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ CALL_4         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_5         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_6         : num  0 1 0 0 0 0 0 0 0 0 ...
```

```
##  $ CALL_7          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALL_8          : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
###### CREATING SEPERATE DATAFRAME FOR OUR TARGET VARIABLE
TRAIN_CHURN<-TRAIN[,1]
TEST_CHURN<-TEST[,1]
###### PERFORMING K-NEAREST NEIGHBOUR ALGORITHM
###### CHOOSING OPTIMUM 'K' VALUE
###### METHOD :1 ;USING FOR LOOP FOR DIFFERENT K VALUES
set.seed(123)
i=3
k.optm=3
for(i in 3:55){
  knn.mod<-knn(train = NORMALIZED_TRAIN,test = NORMALIZED_TEST,cl=TRAIN_CHURN
,k=i)
  k.optm[i]<-100 * sum(TEST_CHURN==knn.mod)/NROW(TEST_CHURN)
  k=i
  cat(k,'=',k.optm[i], '',}
    plot(k.optm,type="b",xlab="k-VALUES",ylab="ACCURACY LEVEL")
# THIS METHOD SAYS K = 7 for good accuracy
###### METHOD:2 ; CHOOSING K - VALUE USING CROSS - VALIDATION
set.seed(400)
ctrl<-trainControl(method = "repeatedcv",repeats=10)
knnfit<-train(Churn~.,data = TRAIN,method="knn",trControl=ctrl,preProcess=c("
center","scale"),tuneLength=20)
knnfit
```

```
## k-Nearest Neighbors
##
## 2029 samples
##   15 predictor
##    2 classes: '0', '1'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 1826, 1826, 1826, 1826, 1827, 1826, ...
```

```
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8664864  0.7328571
##    7  0.8755027  0.7509413
##    9  0.8723507  0.7446388
##   11  0.8645190  0.7289787
##   13  0.8597385  0.7194229
##   15  0.8561916  0.7123325
##   17  0.8529336  0.7058137
##   19  0.8527853  0.7055180
##   21  0.8546597  0.7092692
##   23  0.8527373  0.7054342
##   25  0.8519522  0.7038777
##   27  0.8493404  0.6986630
##   29  0.8486995  0.6973897
##   31  0.8478613  0.6957201
##   33  0.8462820  0.6925603
##   35  0.8452475  0.6904957
##   37  0.8447059  0.6894117
##   39  0.8448044  0.6896144
##   41  0.8436226  0.6872492
##   43  0.8417995  0.6836075
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
plot(knnfit)
```



```
knnfit$bestTune
```

```
##   k
## 2 7
```

```
# 10 REPEAT CROSS-VALIDATION SAYS THAT K =7 IS THE OPTIMAL VALUE WITH GOOD AC
CURACY
```

```
####### SETTING UP THE KNN BASED CLASSIFIER
knn.best<-knn(train = NORMALIZED_TRAIN,test = NORMALIZED_TEST,cl=TRAIN_CHURN,
k=7,prob = TRUE,use.all = TRUE)



####### PERFORMANCE METRIC MEASURES
###### CONFUSION MATRIX
caret::confusionMatrix(table(knn.best,TEST_CHURN),positive="1",mode="everythi
ng")
```

```
## Confusion Matrix and Statistics
## TEST_CHURN
## knn.best   0   1
##        0 384  52
##        1  59 374
##
##                Accuracy : 0.8723
##                  95% CI : (0.8482, 0.8937)
##     No Information Rate : 0.5098
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7445
##
##  Mcnemar's Test P-Value : 0.569
##
##             Sensitivity : 0.8779
##             Specificity : 0.8668
##          Pos Pred Value : 0.8637
##          Neg Pred Value : 0.8807
##               Precision : 0.8637
##                  Recall : 0.8779
##                      F1 : 0.8708
##              Prevalence : 0.4902
##          Detection Rate : 0.4304
##    Detection Prevalence : 0.4983
```

```
##       Balanced Accuracy : 0.8724
##
##        'Positive' Class : 1
##
table(actual=TEST_CHURN,predicted=knn.best)
##       predicted
## actual   0   1
##      0 384  59
##      1  52 374
###### RECEIVER OPEARATOR CHARACTERISTIC CURVE (ROC)
scores.knn<-attr(knn.best,"prob")
scores.knn[knn.best=="0"]<-1-scores.knn[knn.best=="0"]
scores<-data.frame(kNN=scores.knn)
HMeasure(TEST_CHURN,scores)->result
summary(result,show.all=TRUE)
##         Length Class      Mode
## metrics 22     data.frame list
class(result)
## [1] "hmeasure"
knn.best=as.numeric(knn.best)
plot.roc(TEST_CHURN,knn.best,col="red",lwd=2,main="ROC CURVE",text.adj=c(-0.2
,1.7))
grid(col = "aquamarine")
box()
#### OTHER PERFORMANCE MEASURES
####### GINI COEFFICIENT AND KS STATISTIC
result$metrics
##             H      Gini       AUC      AUCH        KS       MER       MWL
## kNN 0.650898 0.8767261 0.9383631 0.9395527 0.7446613 0.127733 0.1276205
##     Spec.Sens95 Sens.Spec95       ER      Sens      Spec Precision    Reca
ll
## kNN   0.6320918   0.7225148 0.127733 0.8755869 0.8690745 0.8654292 0.87558
69
##           TPR       FPR         F    Youden  TP FP  TN FN
## kNN 0.8755869 0.1309255 0.8704784 0.7446613 373 58 385 53
```

11.4 NAÏVE BAYES

```
# CREATING DATAFRAME WITH ONLY THE SIGNIFICANT PREDICTORS
NEWDATA<-DATA[,-c(2,4,8,9)]


# ASSUMPTION :1 - UNCORRELATED PREDICTORS
(cor(NEWDATA))->corr
corrplot(corr,method = "number")
# THE PREDICTORS SATISFIES THE ASSUMPTION OF NAIVE BAYES BEING UNCORRELATED T
O EACH OTHER
## CREATING DUMMIES FOR CUSTOMER SERVICE CALLS
NEWDATA$CALL_0<-ifelse(NEWDATA$CustServCalls=="0",1,0)
NEWDATA$CALL_1<-ifelse(NEWDATA$CustServCalls=="1",1,0)
NEWDATA$CALL_2<-ifelse(NEWDATA$CustServCalls=="2",1,0)
NEWDATA$CALL_3<-ifelse(NEWDATA$CustServCalls=="3",1,0)
NEWDATA$CALL_4<-ifelse(NEWDATA$CustServCalls=="4",1,0)
NEWDATA$CALL_5<-ifelse(NEWDATA$CustServCalls=="5",1,0)
NEWDATA$CALL_6<-ifelse(NEWDATA$CustServCalls=="6",1,0)
NEWDATA$CALL_7<-ifelse(NEWDATA$CustServCalls=="7",1,0)
NEWDATA$CALL_8<-ifelse(NEWDATA$CustServCalls=="8",1,0)
NEWDATA<-NEWDATA[-4]


# SETTING THE DATATYPE FOR CATEGORICAL VARIABLES
NEWDATA$Churn=as.factor(NEWDATA$Churn)
NEWDATA$ContractRenewal=as.factor(NEWDATA$ContractRenewal)
NEWDATA$CALL_0=as.factor(NEWDATA$CALL_0)
NEWDATA$CALL_1=as.factor(NEWDATA$CALL_1)
NEWDATA$CALL_2=as.factor(NEWDATA$CALL_2)
NEWDATA$CALL_3=as.factor(NEWDATA$CALL_3)
NEWDATA$CALL_4=as.factor(NEWDATA$CALL_4)
NEWDATA$CALL_5=as.factor(NEWDATA$CALL_5)
```

```
NEWDATA$CALL_6=as.factor(NEWDATA$CALL_6)
NEWDATA$CALL_7=as.factor(NEWDATA$CALL_7)
NEWDATA$CALL_8=as.factor(NEWDATA$CALL_8)

str(NEWDATA)
```

```
## 'data.frame':    3333 obs. of  15 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
##  $ DayMins        : num  265 162 243 299 167 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ CALL_0         : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 1 2 ...
##  $ CALL_1         : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 2 1 ...
##  $ CALL_2         : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ CALL_3         : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...
##  $ CALL_4         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_5         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_6         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_7         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CALL_8         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
funModeling::plot_num(NEWDATA)
```



```
# ASSUMPTION : 2 - CONTINUOUS PREDICTORS ARE NORMALLY DISTRIBUTED
# NAIVE BAYES STRONGLY ASSUMES CONTINUOUS VARIABLES ARE NORMALLY DISTRIBUTED
###### NORMALITY TEST FOR THE CONTINUOUS PREDICTORS
# NULL HYPOTHESIS : SAMPLE DISTRIBUTION IS NORMAL
# ALTERNATE HYPOTHESIS : SAMPLE DISTRIBUTION IS NOT NORMAL

MVN::mvn(NEWDATA[,3:6])
```

```
## $multivariateNormality
##              Test         Statistic               p value Result
```

```
## 1 Mardia Skewness  1105.61252875762 1.12149920460133e-221     NO
## 2 Mardia Kurtosis 0.728098340783713     0.466553389463242    YES
## 3             MVN              <NA>                   <NA>     NO
##
## $univariateNormality
##           Test   Variable Statistic   p value Normality
## 1 Shapiro-Wilk DataUsage     0.6723  <0.001      NO
## 2 Shapiro-Wilk  DayMins      0.9995  0.6401      YES
## 3 Shapiro-Wilk OverageFee    0.9996   0.71       YES
## 4 Shapiro-Wilk  RoamMins     0.9937  <0.001      NO
##
## $Descriptives
##               n        Mean   Std.Dev Median Min    Max   25th   75th
## DataUsage  3333   0.8164746  1.272668   0.00   0   5.40   0.00   1.78
## DayMins    3333 179.7750975 54.467389 179.40   0 350.80 143.70 216.40
## OverageFee 3333  10.0514881  2.535712  10.07   0  18.19   8.33  11.77
## RoamMins   3333  10.2372937  2.791840  10.30   0  20.00   8.50  12.10
##                   Skew    Kurtosis
## DataUsage   1.27091258  0.03894223
## DayMins    -0.02905090 -0.02349700
## OverageFee -0.02382388  0.02204686
## RoamMins   -0.24491534  0.60430786
```

```
## TEST SAYS DATAUSAGE AND ROAMMINS ARE NOT BELLCURVES,WHEREAS DAYMINS,OVERAG
EFEE ARE NEARLY BELL CURVES
hist(NEWDATA$DataUsage,col="orange")
ggqqplot(NEWDATA$DataUsage,col="orange")
hist(NEWDATA$DayMins,col="brown")
ggqqplot(NEWDATA$DayMins,col="brown")
hist(NEWDATA$OverageFee,col="coral")
ggqqplot(NEWDATA$OverageFee,col="coral")
hist(NEWDATA$RoamMins,col="sea green")
ggqqplot(NEWDATA$RoamMins,col="sea green")
###### CROSS CHECK WITH SHAPIRO-WILK & KS METHOD
###### SHAPIRO - WILK'S METHOD
```

```
shapiro.test(NEWDATA$DataUsage)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  NEWDATA$DataUsage
## W = 0.67226, p-value < 2.2e-16
```

```
shapiro.test(NEWDATA$DayMins)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  NEWDATA$DayMins
## W = 0.99954, p-value = 0.6401
```

```
shapiro.test(NEWDATA$OverageFee)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  NEWDATA$OverageFee
## W = 0.99957, p-value = 0.71
```

```
shapiro.test(NEWDATA$RoamMins)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  NEWDATA$RoamMins
## W = 0.99373, p-value = 7.998e-11
```

```
###### KOLMOGOROV - SMIRNOV TEST
ks.test(NEWDATA$DataUsage,"pnorm",mean=mean(NEWDATA$DataUsage),sd=sd(NEWDATA$
DataUsage))
```

```
##  One-sample Kolmogorov-Smirnov test
## data:  NEWDATA$DataUsage
## D = 0.34207, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(NEWDATA$DayMins,"pnorm",mean=mean(NEWDATA$DayMins),sd=sd(NEWDATA$DayM
ins))
```

```
##  One-sample Kolmogorov-Smirnov test
## data:  NEWDATA$DayMins
```

```
## D = 0.0092545, p-value = 0.9377
## alternative hypothesis: two-sided
```

```
ks.test(NEWDATA$OverageFee,"pnorm",mean=mean(NEWDATA$OverageFee),sd=sd(NEWDATA$OverageFee))
```

```
##  One-sample Kolmogorov-Smirnov test
## data:  NEWDATA$OverageFee
## D = 0.012611, p-value = 0.6641
## alternative hypothesis: two-sided
```

```
ks.test(NEWDATA$RoamMins,"pnorm",mean=mean(NEWDATA$RoamMins),sd=sd(NEWDATA$RoamMins))
```

```
##  One-sample Kolmogorov-Smirnov test
## data:  NEWDATA$RoamMins
## D = 0.026219, p-value = 0.02046
## alternative hypothesis: two-sided
```

```
# ALL TYPES OF NORMALITY TEST SHOWS THAT THE VARIABLE DATAUSAGE AND ROAMMINS ARE NOT NORMALLY DISTRIBUTED

###### NORMALISING THE CONTINUOUS PREDICTORS

###### TRANSFORMING DATA USING YEOJOHNSON'S METHOD

summary(NEWDATA[,3:6])
```

```
##    DataUsage          DayMins        OverageFee       RoamMins
##  Min.   :0.0000   Min.   :  0.0   Min.   : 0.00   Min.   : 0.00
##  1st Qu.:0.0000   1st Qu.:143.7   1st Qu.: 8.33   1st Qu.: 8.50
##  Median :0.0000   Median :179.4   Median :10.07   Median :10.30
##  Mean   :0.8165   Mean   :179.8   Mean   :10.05   Mean   :10.24
##  3rd Qu.:1.7800   3rd Qu.:216.4   3rd Qu.:11.77   3rd Qu.:12.10
##  Max.   :5.4000   Max.   :350.8   Max.   :18.19   Max.   :20.00
```

```
PREPROCESS<-preProcess(NEWDATA[,3:6],method = c("YeoJohnson"))
print(PREPROCESS)
```

```
## Created from 3333 samples and 4 variables
##
## Pre-processing:
##   - ignored (0)
##   - Yeo-Johnson transformation (4)
##
## Lambda estimates for Yeo-Johnson transformation:
```

```
## -1.85, 1.01, 1.04, 1.25
```

```
TRANSFORMED<-predict(PREPROCESS,NEWDATA[,3:6])
summary(TRANSFORMED)
```

```
##    DataUsage          DayMins        OverageFee        RoamMins
##  Min.   :0.0000   Min.   :  0.0   Min.   : 0.000   Min.   : 0.00
##  1st Qu.:0.0000   1st Qu.:152.1   1st Qu.: 8.789   1st Qu.:12.62
##  Median :0.0000   Median :190.5   Median :10.678   Median :15.88
##  Mean   :0.1709   Mean   :191.0   Mean   :10.670   Mean   :15.93
##  3rd Qu.:0.4597   3rd Qu.:230.3   3rd Qu.:12.535   3rd Qu.:19.27
##  Max.   :0.5242   Max.   :375.9   Max.   :19.618   Max.   :35.47
```

```
ggqqplot(TRANSFORMED$DataUsage,col="orange")
ggqqplot(TRANSFORMED$DayMins,col="brown")
ggqqplot(TRANSFORMED$OverageFee,col="coral")
ggqqplot(TRANSFORMED$RoamMins,col="sea green")
funModeling::plot_num(TRANSFORMED)
NEWDATA<-NEWDATA[,-c(3,4,5,6)]
NEWDATA=cbind(TRANSFORMED,NEWDATA)


set.seed(300)
SPLIT_DATA=initial_split(DATA.BAL,prop = 0.70)
TRAIN=training(SPLIT_DATA)
TEST=testing(SPLIT_DATA)
table(TRAIN$Churn)
```

```
##
##    0    1
## 1008 1021
```

```
prop.table(table(TRAIN$Churn))
```

```
##
##         0         1
## 0.4967965 0.5032035
```

```
table(TEST$Churn)
```

```
##
##   0   1
## 441 428
```

```
prop.table(table(TEST$Churn))
```

```
##
##         0         1
## 0.5074799 0.4925201
```

```
###### CREATING SEPARATE OBJECTS FOR OUR PREDICTORS & PREDICTAND
x<-TRAIN[,-5]
y<-TRAIN[,5]
###### BULIDING NAIVE- BAYES CLASSIFIER MODEL
###### METHOD :1
MODEL<-naiveBayes(Churn~.,data=TRAIN)
PRED<-predict(MODEL,newdata = TEST)

###### METHOD :2
TRAIN_CONTROL<-trainControl(method = "cv",number = 10)
NBMODEL<-train(x=x,y=y,method = "nb",trControl = TRAIN_CONTROL,prob=TRUE,use.
all=TRUE,preProcess=c("center","scale"),tuneLength=20)
PREDICTION<-predict(NBMODEL,newdata=TEST,prob=TRUE)
print(NBMODEL)
```

```
## Naive Bayes
##
## 2029 samples
##   14 predictor
##    2 classes: '0', '1'
##
## Pre-processing: centered (4), scaled (4), ignore (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1827, 1827, 1826, 1826, 1825, 1826, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8274854  0.6550858
##    TRUE      0.8210839  0.6421126
##
```

```
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adj
ust
##  = 1.
```

```
varImp(NBMODEL)->IMP
plot(IMP)
###### PERFORMANCE MEASURE METRICS
###### CONFUSION MATRIX
caret::confusionMatrix(PREDICTION,TEST$Churn,positive="1",mode="everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 381  80
##          1  60 348
##
##                Accuracy : 0.8389
##                  95% CI : (0.8127, 0.8627)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6775
##
##  Mcnemar's Test P-Value : 0.1083
##
##             Sensitivity : 0.8131
##             Specificity : 0.8639
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.8265
##               Precision : 0.8529
##                  Recall : 0.8131
```

```
##                       F1 : 0.8325
##               Prevalence : 0.4925
##           Detection Rate : 0.4005
##     Detection Prevalence : 0.4695
##        Balanced Accuracy : 0.8385
##
##         'Positive' Class : 1
##
```

```
table(Actual=TEST$Churn,predicted=PREDICTION)
```

```
##        predicted
## Actual   0   1
##      0 381  60
##      1  80 348
```

```
###### RECEIVER OPEARATOR CHARACTERISTIC CURVE (ROC)
```

```
TEST$Churn=as.numeric(TEST$Churn)
PREDICTION=as.numeric(PREDICTION)
ROCRpred<-prediction(PREDICTION,TEST$Churn)
ROCRperf<-performance(ROCRpred,"tpr","fpr")
plot(ROCRperf,colorize=TRUE,lwd=2,main="ROC CURVE",text.adj=c(-0.2,1.7))
box()
abline(0,1,lty=300,col="brown")
grid(col = "aquamarine")
as.numeric(performance(ROCRpred,"auc")@y.values)->AUC


AUC
```

```
## [1] 0.8385148
```

```
##### MISCLASSIFICATION ERROR
```

```
misClassError(TEST$Churn,PREDICTION )
```

```
## [1] 0.4925
```

```
###### GINI COEFFICIENT
```

```
GINI<-(2*AUC)-1
GINI
```

```
## [1] 0.6770297
```

```
###### K STATISTIC
```

```
ks_stat(TEST$Churn,PREDICTION)
```

```
## [1] 0.6158
```

```
ks_stat(TEST$Churn,PREDICTION,returnKSTable = TRUE)
```

```
##    rank total_pop non_responders responders expected_responders_by_random
## 1     1        87             60         27                       42.84925
## 2     2        87              0         87                       42.84925
## 3     3        87              0         87                       42.84925
## 4     4        87              0         87                       42.84925
## 5     5        87             27         60                       42.84925
## 6     6        87             87          0                       42.84925
## 7     7        87             87          0                       42.84925
## 8     8        87             87          0                       42.84925
## 9     9        87             87          0                       42.84925
## 10   10        86              6         80                       42.35673
##    perc_responders perc_non_responders cum_perc_responders
## 1       0.06308411          0.13605442          0.06308411
## 2       0.20327103          0.00000000          0.26635514
## 3       0.20327103          0.00000000          0.46962617
## 4       0.20327103          0.00000000          0.67289720
## 5       0.14018692          0.06122449          0.81308411
## 6       0.00000000          0.19727891          0.81308411
## 7       0.00000000          0.19727891          0.81308411
## 8       0.00000000          0.19727891          0.81308411
## 9       0.00000000          0.19727891          0.81308411
## 10      0.18691589          0.01360544          1.00000000
##    cum_perc_non_responders  difference
## 1                0.1360544 -0.07297031
## 2                0.1360544  0.13030072
## 3                0.1360544  0.33357175
## 4                0.1360544  0.53684277
## 5                0.1972789  0.61580520
## 6                0.3945578  0.41852629
## 7                0.5918367  0.22124738
## 8                0.7891156  0.02396847
```

```
## 9                0.9863946 -0.17331045
## 10               1.0000000  0.00000000
```

```
ks_plot(TEST$Churn,PREDICTION)
###### GAINS TABLE
###### TO CALCULATE GAINS THE ORIGINAL TARGET SHOULD BE IN NUMERIC FORMAT
library(gains)
GAINS<-gains(actual = TEST$Churn,predicted = PREDICTION,groups = 10)
print(GAINS)
```

```
## Depth                                Cume    Cume Pct                      Mean
##  of             Cume      Mean      Mean    of Total     Lift    Cume      Model
## File      N       N      Resp      Resp       Resp     Index    Lift      Score
## ------------------------------------------------------------------------------
##   47    408     408      1.85      1.85      58.3%      124     124       2.00
##  100    461     869      1.17      1.49     100.0%       79     100       1.00
```