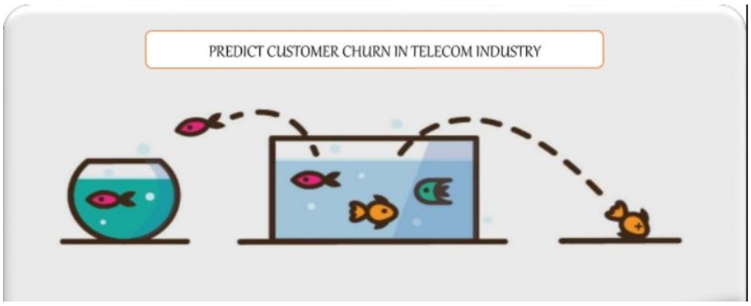


CHURN ANALYSIS

Predictive Modelling

Baijayanti Chakraborty
PGP-BABI

1. Project Objective

Customer Churn is a burning problem for Telecom companies. In this project, we simulate one such case of customer churn where we work on a data of postpaid customers with a contract. The data has information about the customer usage behavior, contract details and the payment details. The data also indicates which were the customers who canceled their service. Based on this past data, we need to build a model which can predict whether a customer will cancel their service in the future or not.

One is expected to do the following :

1. **EDA**
   - How does the data looks like, Univariate and bivariate analysis. Plots and charts which illustrate the relationships between variables
   - Look out for outliers and missing values
   - Check for multicollinearity & treat it
   - Summarize the insights you get from EDA
2. **Build Models and compare them to get to the best one**
   - Logistic Regression
   - KNN
   - Naive Bayes
   - Model Comparison using Model Performance metrics & Interpretation
3. **Actionable Insights**
   - Interpretation & Recommendations from the best model

2. Exploratory Data Analysis

2.1 Install and load the needed libraries.

```
#clean the global environment
rm(list = ls())
#set the working directory
#setwd("~/Desktop/PGP-BABI/Predictive Modelling/week3-frequency
based")

#load the libraries
library(readxl)

## Warning: package 'readxl' was built under R version 3.6.1

library(DataExplorer)

## Warning: package 'DataExplorer' was built under R version
3.6.1

library(caret)

## Warning: package 'caret' was built under R version 3.6.1

## Loading required package: lattice
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.1

library(psych)

## Warning: package 'psych' was built under R version 3.6.1

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

library(GGally)

## Warning: package 'GGally' was built under R version 3.6.1

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.1

## corrplot 0.84 loaded

library(earth)

## Warning: package 'earth' was built under R version 3.6.1

## Loading required package: Formula

## Loading required package: plotmo

## Warning: package 'plotmo' was built under R version 3.6.1

## Loading required package: plotrix

##
## Attaching package: 'plotrix'

## The following object is masked from 'package:psych':
##
##     rescale

## Loading required package: TeachingDemos

library(varImp)

## Warning: package 'varImp' was built under R version 3.6.1

## Loading required package: measures

## Warning: package 'measures' was built under R version 3.6.1
```

```
##
## Attaching package: 'measures'

## The following object is masked from 'package:psych':
##
##     AUC

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## Loading required package: party

## Warning: package 'party' was built under R version 3.6.1

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 3.6.1

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.1

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 3.6.1

##
## Attaching package: 'varImp'

## The following object is masked from 'package:caret':
##
##     varImp

library(IMTest)

## Warning: package 'IMTest' was built under R version 3.6.1

## Loading required package: ltm

## Warning: package 'ltm' was built under R version 3.6.1
```

```
## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.6.1

## Loading required package: msm

## Warning: package 'msm' was built under R version 3.6.1

## Loading required package: polycor

## Warning: package 'polycor' was built under R version 3.6.1

##
## Attaching package: 'polycor'

## The following object is masked from 'package:psych':
##
##     polyserial

##
## Attaching package: 'ltm'

## The following object is masked from 'package:psych':
##
##     factor.scores

library(pscl)

## Warning: package 'pscl' was built under R version 3.6.1

## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis

library(e1071)

## Warning: package 'e1071' was built under R version 3.6.1

library(caret)
library(pROC)

## Warning: package 'pROC' was built under R version 3.6.1

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(class)
```

```
## Warning: package 'class' was built under R version 3.6.1

library(gmodels)

## Warning: package 'gmodels' was built under R version 3.6.1

##
## Attaching package: 'gmodels'

## The following object is masked from 'package:pROC':
##
##     ci

library(car)

## Warning: package 'car' was built under R version 3.6.1

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:modeltools':
##
##     Predict

## The following object is masked from 'package:psych':
##
##     logit

library(ROCR)

## Warning: package 'ROCR' was built under R version 3.6.1

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.6.1

##
## Attaching package: 'gplots'

## The following object is masked from 'package:plotrix':
##
##     plotCI

## The following object is masked from 'package:stats':
##
##     lowess

library(blorr)

## Warning: package 'blorr' was built under R version 3.6.1

library(class)
library(car)
library(caret)
```

```r
library(class)
library(devtools)

## Warning: package 'devtools' was built under R version 3.6.1

## Loading required package: usethis

## Warning: package 'usethis' was built under R version 3.6.1

library(e1071)
library(ggplot2)
library(Hmisc)

## Warning: package 'Hmisc' was built under R version 3.6.1

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:e1071':
##
##     impute

## The following objects are masked from 'package:TeachingDemos':
##
##     cnvrt.coords, subplot

## The following object is masked from 'package:psych':
##
##     describe

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(klaR)

## Warning: package 'klaR' was built under R version 3.6.1

## This version of Shiny is designed to work with 'htmlwidgets'
>= 1.5.
##     Please upgrade via install.packages('htmlwidgets').

##
## Attaching package: 'klaR'
```

```
## The following object is masked from 'package:TeachingDemos':
##
##     triplot
```

```r
library(klaR)
library(MASS)
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.6.1

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize

## The following object is masked from 'package:modeltools':
##
##     empty
```

```r
library(scatterplot3d)
library(SDMTools)
```

```
## Warning: package 'SDMTools' was built under R version 3.6.1

## Registered S3 method overwritten by 'R.oo':
##   method        from
##   throw.default R.methodsS3

##
## Attaching package: 'SDMTools'

## The following object is masked from 'package:pROC':
##
##     auc

## The following objects are masked from 'package:caret':
##
##     sensitivity, specificity
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename,
summarise,
##     summarize
```

```
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize

## The following object is masked from 'package:car':
##
##     recode

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:GGally':
##
##     nasa

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ElemStatLearn)
```

```
## Warning: package 'ElemStatLearn' was built under R version
3.6.1

##
## Attaching package: 'ElemStatLearn'

## The following object is masked from 'package:plyr':
##
##     ozone
```

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.6.1
```

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 3.6.1

##
## Attaching package: 'boot'

## The following object is masked from 'package:survival':
##
##     aml

## The following object is masked from 'package:car':
##
##     logit
```

```
## The following object is masked from 'package:msm':
##
##     cav
## The following object is masked from 'package:psych':
##
##     logit
## The following object is masked from 'package:lattice':
##
##     melanoma
```

2.2 Set the working directory and check the basic statistics of the dataset.

```
#read the dataset
cell = read_excel("Cellphone.xlsx",sheet = 2)
#view the data set
head(cell,7)

## # A tibble: 7 x 11
##   Churn AccountWeeks ContractRenewal DataPlan DataUsage
CustServCalls
##   <dbl>        <dbl>           <dbl>    <dbl>     <dbl>
<dbl>
## 1     0          128               1        1      2.7
1
## 2     0          107               1        1      3.7
1
## 3     0          137               1        0      0
0
## 4     0           84               0        0      0
2
## 5     0           75               0        0      0
3
## 6     0          118               0        0      0
0
## 7     0          121               1        1      2.03
3
## # ... with 5 more variables: DayMins <dbl>, DayCalls <dbl>,
## #   MonthlyCharge <dbl>, OverageFee <dbl>, RoamMins <dbl>

#changing the column names to increase the readability
colnames(cell)

##  [1] "Churn"           "AccountWeeks"    "ContractRenewal"
##  [4] "DataPlan"        "DataUsage"       "CustServCalls"
##  [7] "DayMins"         "DayCalls"        "MonthlyCharge"
## [10] "OverageFee"      "RoamMins"

names(cell)[2] = "AccWeeks"
names(cell)[3] = "ContRenew"
names(cell)[4] = "Plan"
```

```r
names(cell)[5] = "Usage"
names(cell)[6] = "CScalls"
names(cell)[7] = "Min/Day"
names(cell)[8] = "Call/Day"
names(cell)[9] = "Charge/Month"
names(cell)[10] = "Over.Fee"
names(cell)[11] = "RoamingMins"
#Basic data summary, Univariate, Bivariate analysis, graphs
summary(cell)
```

```
##      Churn            AccWeeks       ContRenew           Plan
##  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
##      Usage            CScalls         Min/Day         Call/Day
##  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
##   Charge/Month       Over.Fee      RoamingMins
##  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
```

```r
str(cell)
```

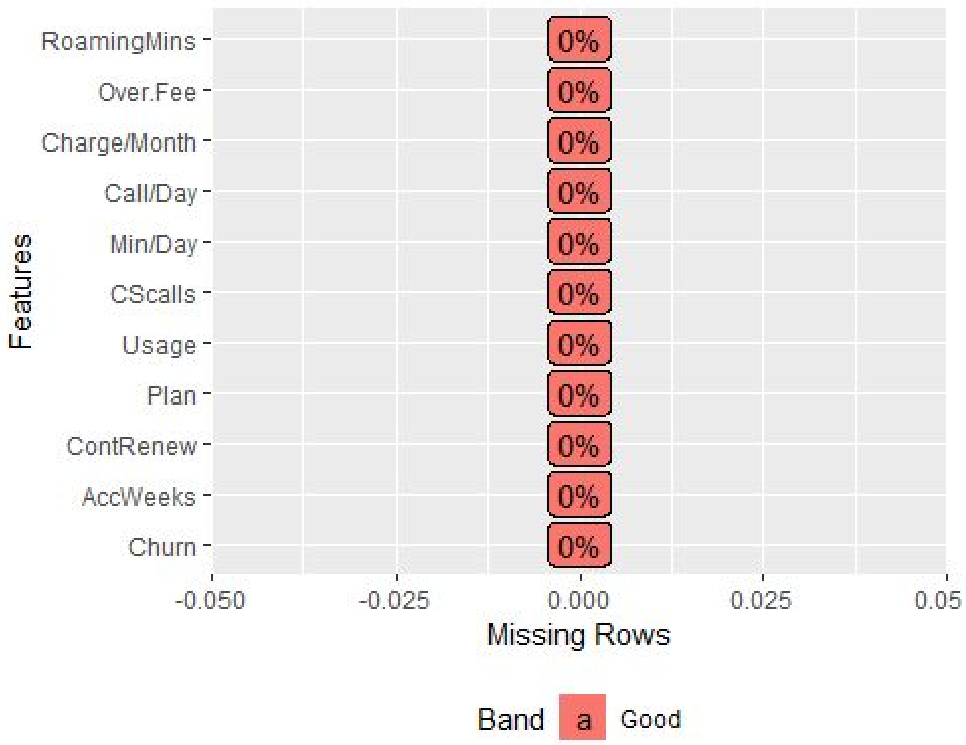
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11
variables:
##  $ Churn       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AccWeeks    : num  128 107 137 84 75 118 121 147 117 141
...
##  $ ContRenew   : num  1 1 1 0 0 0 1 0 1 0 ...
```

```
## $ Plan        : num  1 1 0 0 0 0 1 0 0 1 ...
## $ Usage       : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
## $ CScalls     : num  1 1 0 2 3 0 3 0 1 0 ...
## $ Min/Day     : num  265 162 243 299 167 ...
## $ Call/Day    : num  110 123 114 71 113 98 88 79 97 84 ...
## $ Charge/Month: num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
## $ Over.Fee    : num  9.87 9.78 6.06 3.1 7.42 ...
## $ RoamingMins : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7
11.2 ...
```

```
sum(is.na(cell)) #No null values in the whole dataset
```

```
## [1] 0
```

```
#graph for null value cross check
plot_missing(cell)
```



2.3 The descriptive analytics of the dataset.

```
#descriptive analytics
describe(cell)
```

```
## cell
##
##  11  Variables      3333  Observations
##
------------------------------------------------------------------------
-
## Churn
##          n  missing distinct     Info      Sum     Mean      Gmd
##       3333        0        2    0.372      483   0.1449   0.2479
```

```
##
##
--------------------------------------------------------------------------
-
## AccWeeks
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0      212        1    101.1    45.01       35       50
##      .25      .50      .75      .90      .95
##       74      101      127      152      167
##
## lowest :   1   2   3   4   5, highest: 221 224 225 232 243
##
--------------------------------------------------------------------------
-
## ContRenew
##        n  missing distinct     Info      Sum     Mean      Gmd
##     3333        0        2    0.263     3010   0.9031   0.1751
##
##
--------------------------------------------------------------------------
-
## Plan
##        n  missing distinct     Info      Sum     Mean      Gmd
##     3333        0        2      0.6      922   0.2766   0.4003
##
##
--------------------------------------------------------------------------
-
## Usage
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0      174    0.839   0.8165    1.202     0.00     0.00
##      .25      .50      .75      .90      .95
##     0.00     0.00     1.78     3.05     3.46
##
## lowest : 0.00 0.11 0.12 0.13 0.14, highest: 4.59 4.64 4.73 4.75 5.40
##
--------------------------------------------------------------------------
-
## CScalls
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0       10    0.932    1.563    1.392        0        0
##      .25      .50      .75      .90      .95
##        1        1        2        3        4
##
## Value          0     1     2     3     4     5     6     7     8     9
## Frequency    697  1181   759   429   166    66    22     9     2     2
## Proportion 0.209 0.354 0.228 0.129 0.050 0.020 0.007 0.003 0.001 0.001
##
--------------------------------------------------------------------------
-
## Min/Day
```

```
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0     1667        1    179.8    61.46    89.92   110.32
##      .25      .50      .75      .90      .95
##   143.70   179.40   216.40   249.58   270.74
##
## lowest :   0.0   2.6   7.8   7.9  12.5, highest: 335.5 337.4 345.3
346.8 350.8
##
--------------------------------------------------------------------------
-
## Call/Day
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0      119        1    100.4    22.59     67.0     74.2
##      .25      .50      .75      .90      .95
##     87.0    101.0    114.0    126.0    133.0
##
## lowest :   0  30  35  36  40, highest: 157 158 160 163 165
##
--------------------------------------------------------------------------
-
## Charge/Month
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0      656        1    56.31    18.35    33.26    38.00
##      .25      .50      .75      .90      .95
##    45.00    53.50    66.20    80.50    87.80
##
## lowest :  14.0  15.7  16.0  17.0  19.0, highest: 108.3 108.6 108.7
110.0 111.3
##
--------------------------------------------------------------------------
-
## Over.Fee
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0     1024        1    10.05     2.86     5.94     6.84
##      .25      .50      .75      .90      .95
##     8.33    10.07    11.77    13.29    14.22
##
## lowest :  0.00  1.56  2.11  2.13  2.20, highest: 17.55 17.58 17.71
18.09 18.19
##
--------------------------------------------------------------------------
-
## RoamingMins
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3333        0      162        1    10.24    3.114      5.7      6.7
##      .25      .50      .75      .90      .95
##      8.5     10.3     12.1     13.7     14.7
##
## lowest :  0.0  1.1  1.3  2.0  2.1, highest: 18.2 18.3 18.4 18.9 20.0
##
```

```
------------------------------------------------------------------------
-
```

2.3.1. Univariate and Bivariate analysis

1. **Univariate data –**This type of data consists of only one variable. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it
2. **Bivariate data –**This type of data involves two different variables. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship among the two variables.

Keeping this definition we see that our dataset is a Bivariate one since we see that more than one variable is involved in getting the relationship for the customer churn analysis. The "Churn" factor depends on other variables like "Plan","Uage","Cost of the renewal" etc.

```r
#Correlation Matrix
corrplot(corr = cor(cell), method = "number" , type = "upper")
```

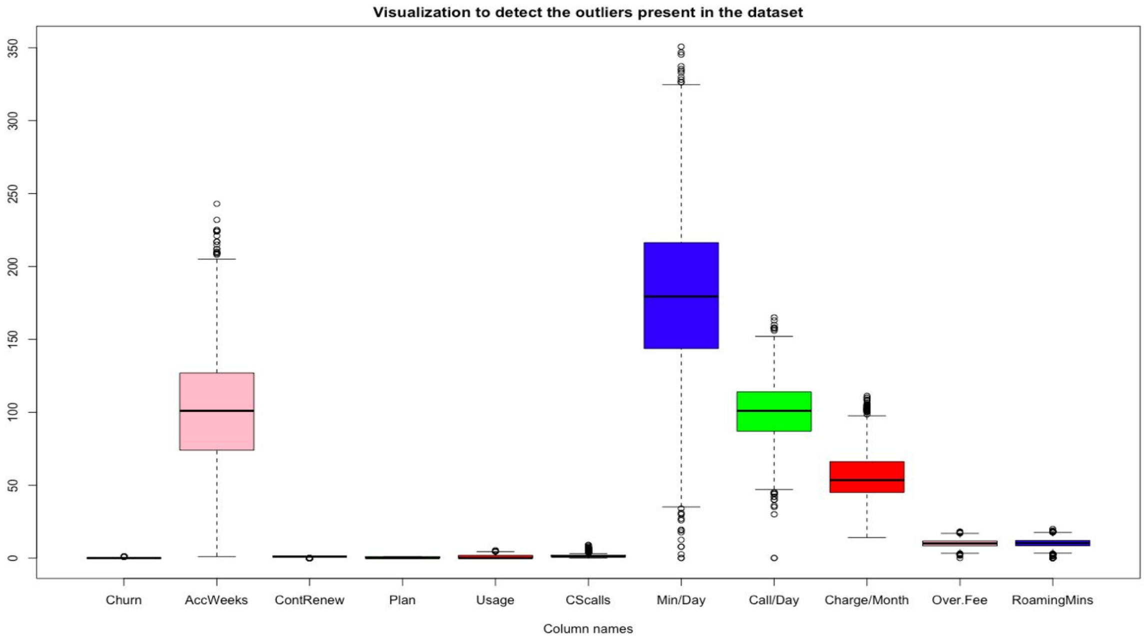

```r
mat = cor(cell)
print(mat)
```

```
##                    Churn      AccWeeks     ContRenew          Plan
## Churn         1.00000000  0.016540742 -0.259851847 -0.102148141
## AccWeeks      0.01654074  1.000000000 -0.024734655  0.002918409
## ContRenew    -0.25985185 -0.024734655  1.000000000 -0.006006371
## Plan         -0.10214814  0.002918409 -0.006006371  1.000000000
## Usage        -0.08719451  0.014390757 -0.019222913  0.945981734
## CScalls       0.20875000 -0.003795939  0.024521956 -0.017823944
## Min/Day       0.20515083  0.006216021 -0.049395824 -0.001684069
## Call/Day      0.01845931  0.038469882 -0.003754626 -0.011085902
## Charge/Month  0.07231271  0.012580670 -0.047291399  0.737489653
## Over.Fee      0.09281243 -0.006749462 -0.019104644  0.021525559
## RoamingMins   0.06823878  0.009513902 -0.045870743 -0.001317871
##                     Usage       CScalls       Min/Day      Call/Day
```

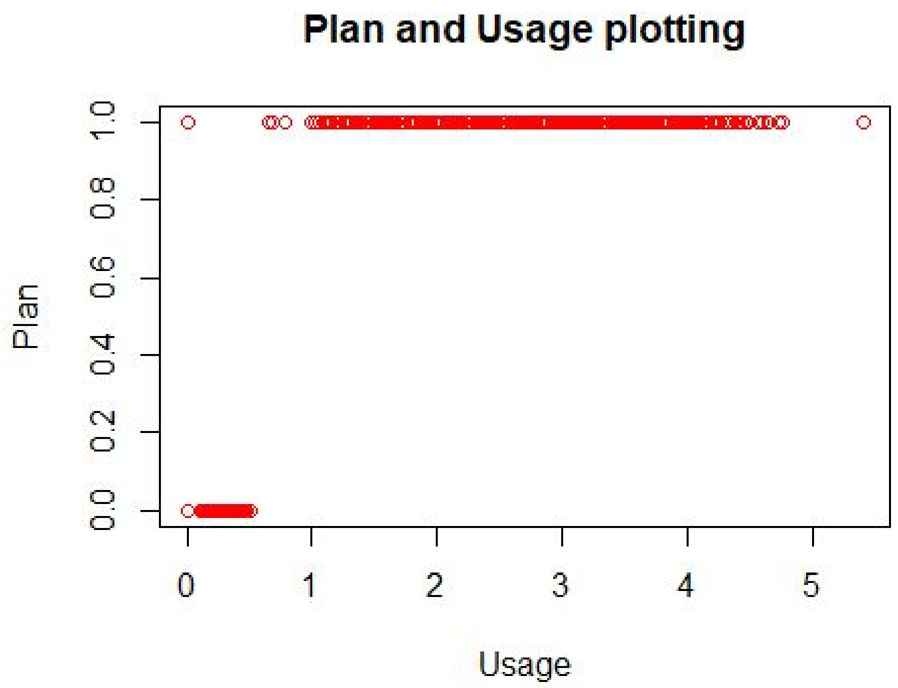
```
## Churn        -0.087194509  0.208749999  0.205150829  0.018459312
## AccWeeks      0.014390757 -0.003795939  0.006216021  0.038469882
## ContRenew    -0.019222913  0.024521956 -0.049395824 -0.003754626
## Plan          0.945981734 -0.017823944 -0.001684069 -0.011085902
## Usage         1.000000000 -0.021722518  0.003175951 -0.007962079
## CScalls      -0.021722518  1.000000000 -0.013423186 -0.018941930
## Min/Day       0.003175951 -0.013423186  1.000000000  0.006750414
## Call/Day     -0.007962079 -0.018941930  0.006750414  1.000000000
## Charge/Month  0.781660429 -0.028016853  0.567967924 -0.007963218
## Over.Fee      0.019637372 -0.012964219  0.007038214 -0.021448602
## RoamingMins   0.162745576 -0.009639680 -0.010154586  0.021564794
##              Charge/Month     Over.Fee  RoamingMins
## Churn         0.072312711  0.092812426  0.068238776
## AccWeeks      0.012580670 -0.006749462  0.009513902
## ContRenew    -0.047291399 -0.019104644 -0.045870743
## Plan          0.737489653  0.021525559 -0.001317871
## Usage         0.781660429  0.019637372  0.162745576
## CScalls      -0.028016853 -0.012964219 -0.009639680
## Min/Day       0.567967924  0.007038214 -0.010154586
## Call/Day     -0.007963218 -0.021448602  0.021564794
## Charge/Month  1.000000000  0.281766048  0.117432607
## Over.Fee      0.281766048  1.000000000 -0.011023336
## RoamingMins   0.117432607 -0.011023336  1.000000000
```

2.3.2. Data Visualizations

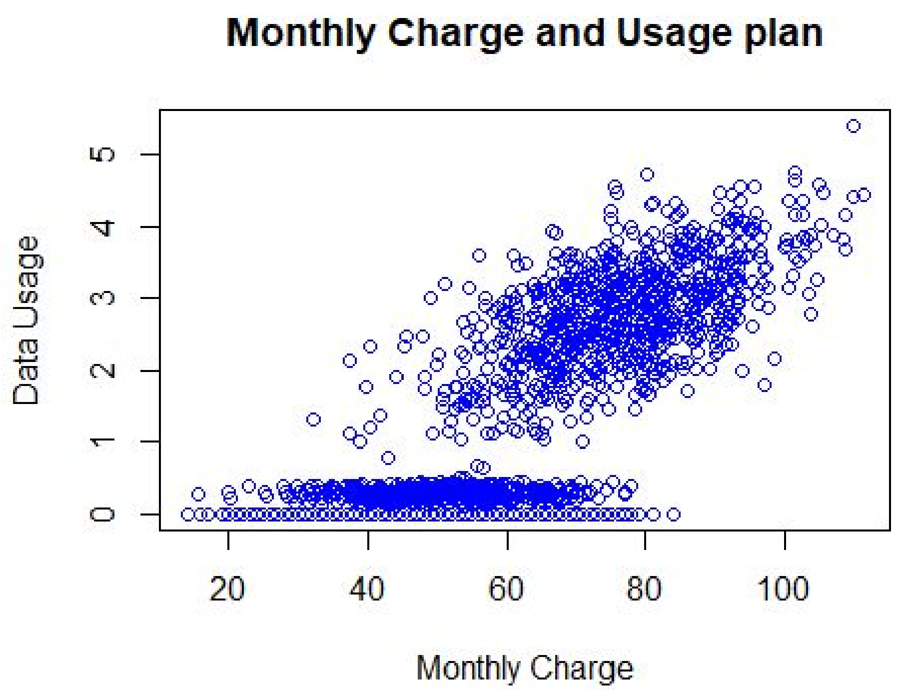
```r
boxplot(cell,main = "Visualization to detect the outliers present in
the dataset",xlab = "Column names",col =
c("red","pink","blue","green"))
```



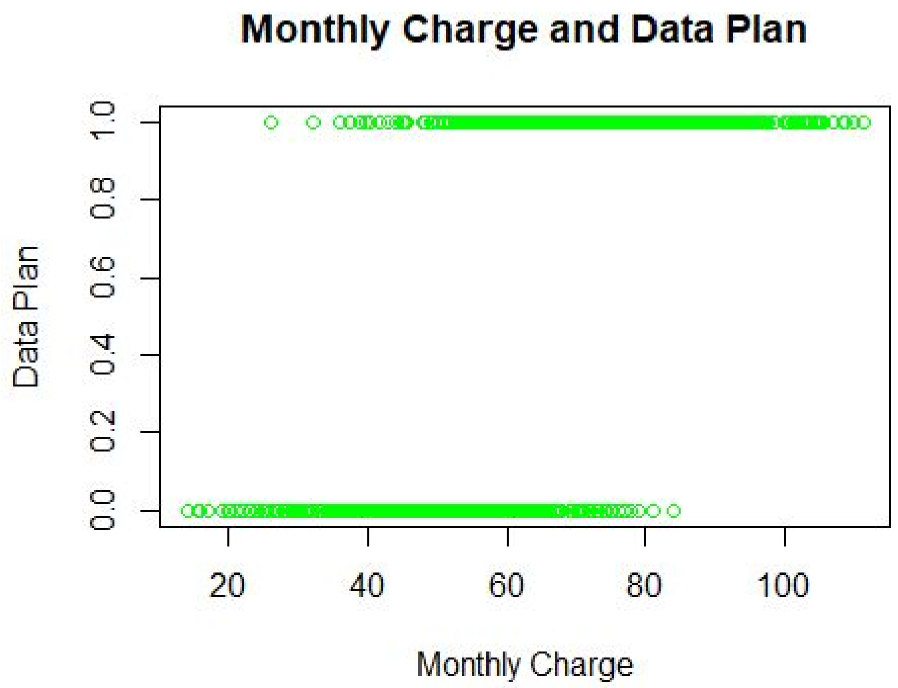
```r
plot(cell$Usage,cell$Plan,main = "Plan and Usage plotting" , xlab =
"Usage" , ylab = "Plan" , col = "red")
```



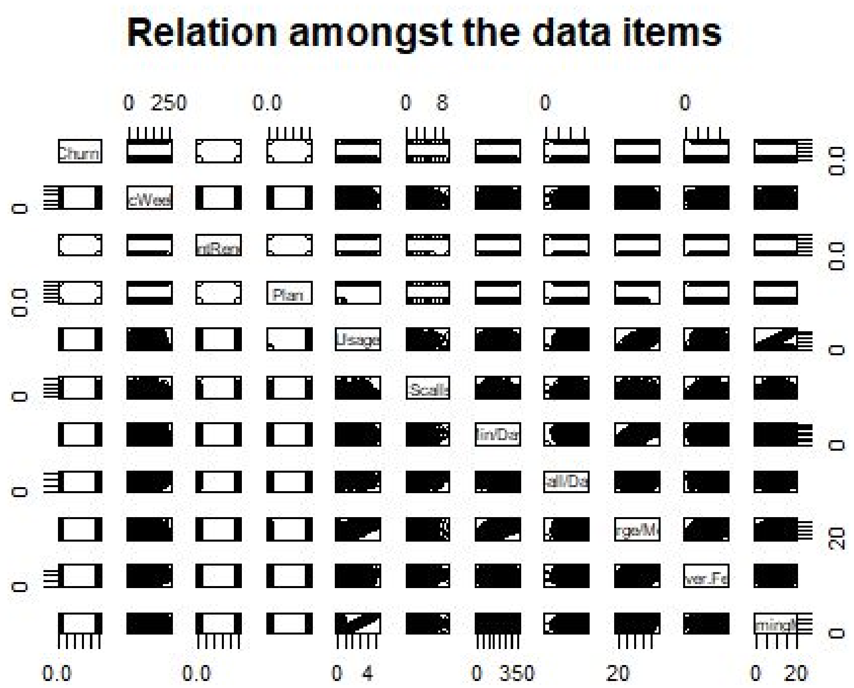

```r
plot(cell$`Charge/Month`,cell$Usage , main = "Monthly Charge and Usage
plan" , xlab = "Monthly Charge",ylab = "Data Usage",col = "blue")
```



```r
plot(cell$`Charge/Month`,cell$Plan , main = "Monthly Charge and Data Plan"
, xlab = "Monthly Charge",ylab = "Data Plan",col = "green")
```

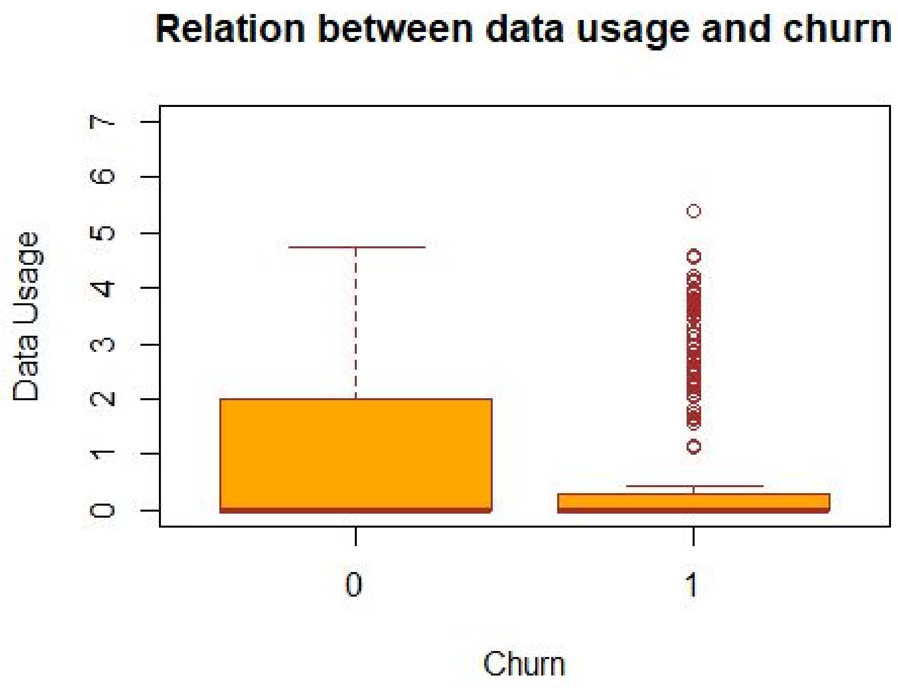



```r
pairs(cell , main = "Relation amongst the data items")
```

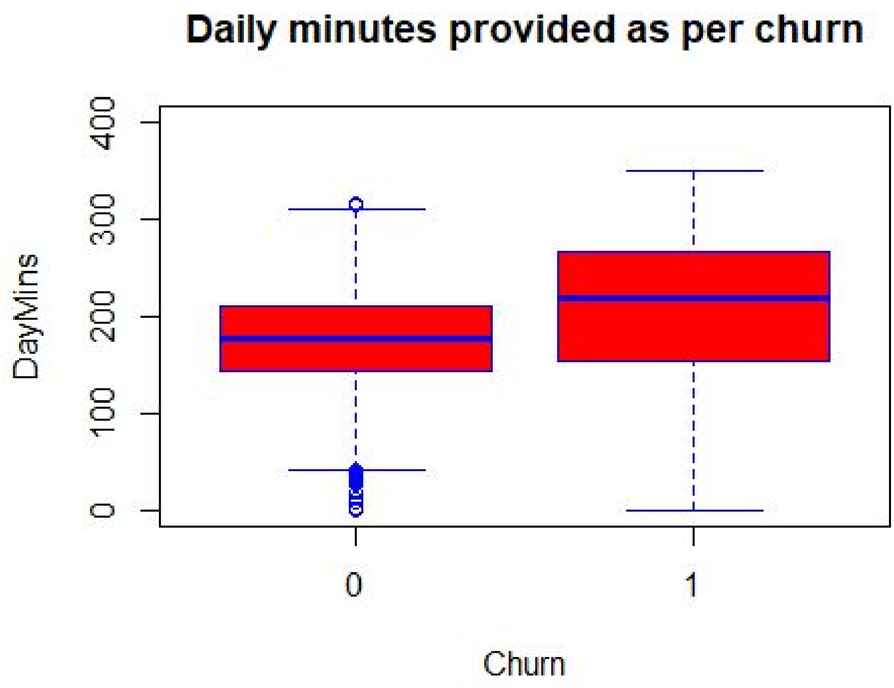



```r
boxplot(Usage~Churn,
data = cell,
```

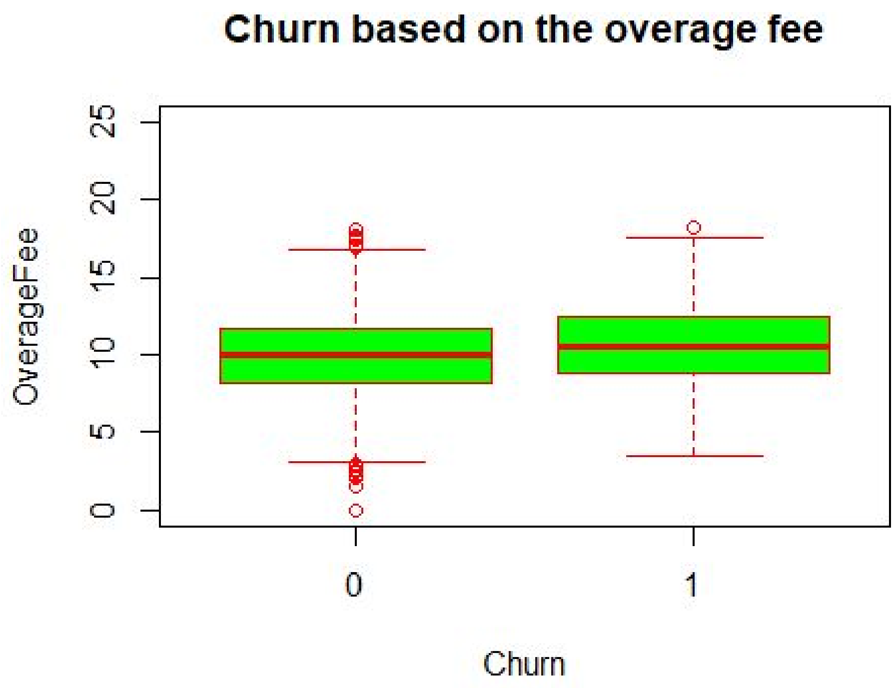
```
main = "Relation between data usage and churn",
xlab = "Churn",
ylab = "Data Usage",
col = "orange",
border = "brown",
ylim = c(0, 7)
)
```



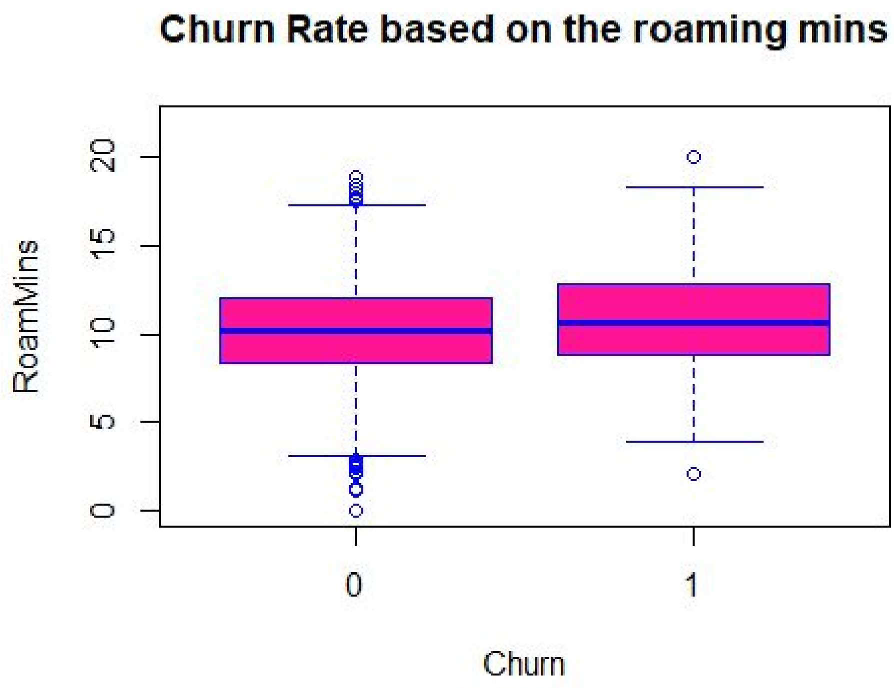

```
boxplot(`Min/Day`~Churn,
 data = cell,
 main = "Daily minutes provided as per churn",
 xlab = "Churn",
 ylab = "DayMins",
 col = "Red",
 border = "Blue",
 ylim = c(0, 400)
)
```



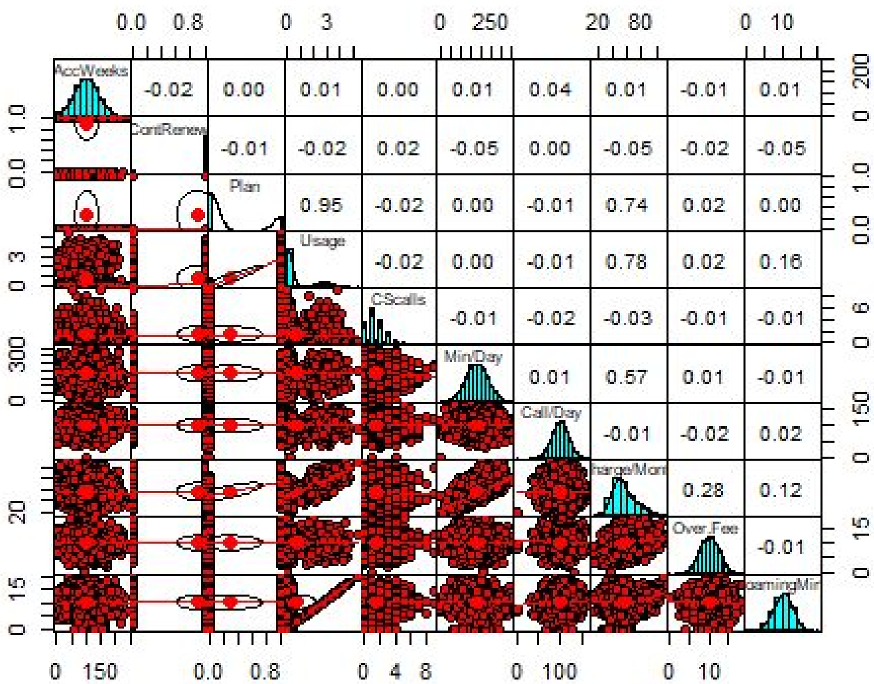

```r
boxplot(Over.Fee~Churn,
 data = cell,
 main = "Churn based on the overage fee",
 xlab = "Churn",
 ylab = "OverageFee",
 col = "Green",
 border = "red",
 ylim = c(0, 25)
)
```



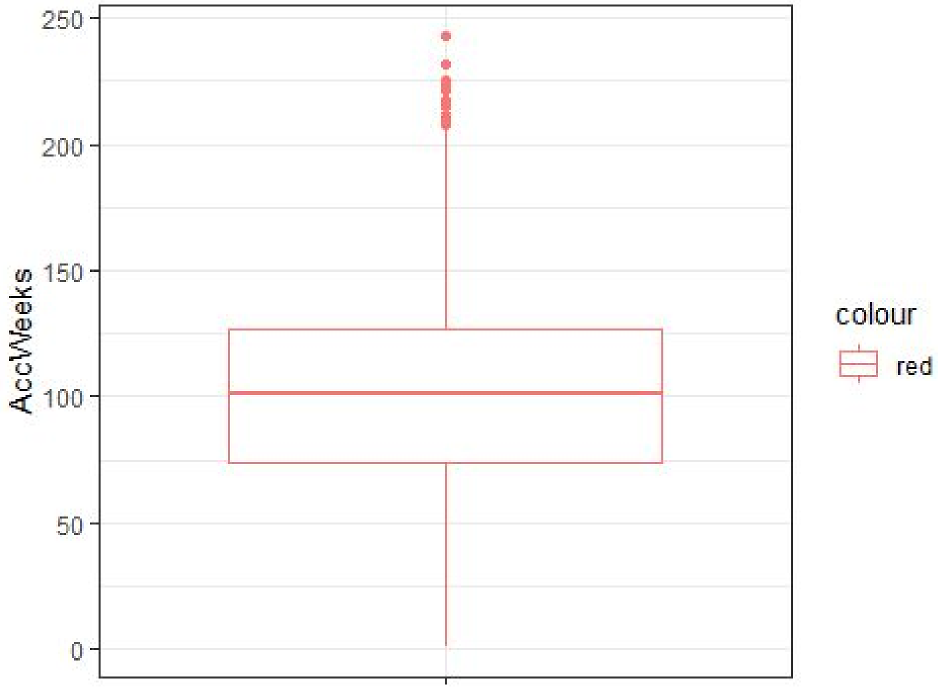

```
boxplot(RoamingMins~Churn,
 data = cell,
 main = "Churn Rate based on the roaming mins",
 xlab = "Churn",
 ylab = "RoamMins",
 col = "Deeppink",
 border = "blue",
 ylim = c(0, 22)
)
```



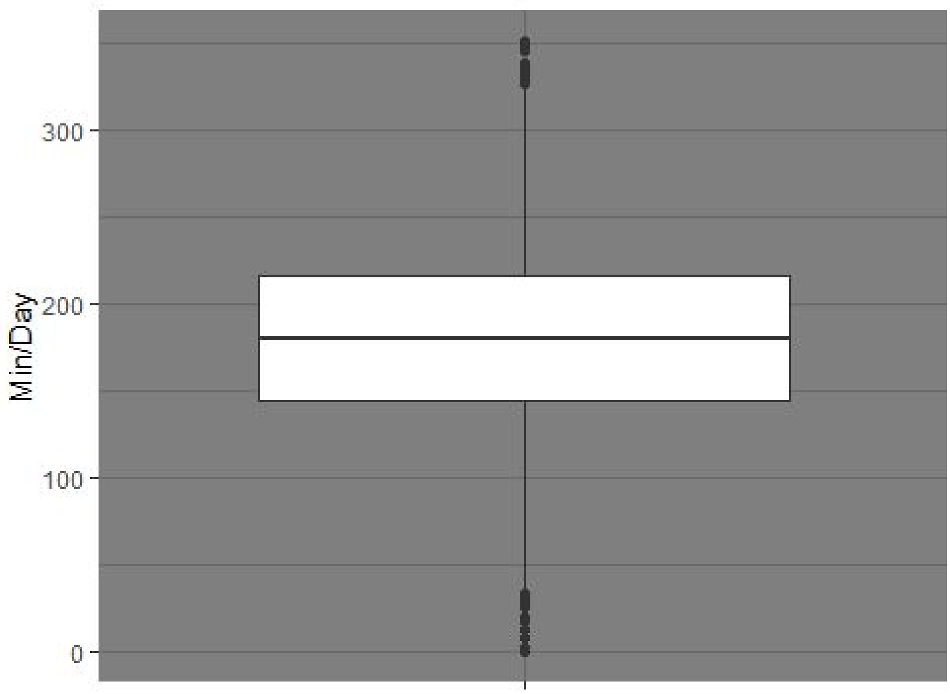

```r
pairs.panels(cell[,c(2:11)],gap = 0, bg = c("red", "green","blue" , main =
"Data Variations")[cell
$Churn], pch = 21)
```



```r
ggplot(cell, aes(y= AccWeeks, x = "", fill = Churn , col = "red")) +
 geom_boxplot()+
 theme_bw()+
 xlab(" ")
```




```r
ggplot(cell, aes(y=`Min/Day`, x = "", fill = Churn )) +
 geom_boxplot()+
 theme_dark()+
 xlab(" ")
```



2.3.3. Insights gained from the Exploratory Data Analysis can be concluded as :-

The variables in the dataset can be initially categorised into two types : categorical and Continuous ones.

The categorical variables are :-

1.Customer Service Calls

2.Contract Renewal

3.Data Plan

The Continuous variables are :-

1.AccountWeeks

2.Data Usage

3.Days/Min

4.Days/calls

5.Monthly Charges

6.Roaming Mins

There are other insights as well which are listed below that has been created from the EDA.

1.	Churn is the target variable for our analysis and we need to find the most optimized method of predicting the churn rate
2.	We do have outliers in the dataset in columns like : a)Churn, b)AccWeek, c)ContRenew , d)Usage , e)CScalls, f)Min/Day g)Call/Day h)Charge/Month i)over.fee j)RoamingMins.
3.	Multicolinearity does not exist in a huge amount amongst the variables.Most of them are either negatively correlated or has very minute correlation.

3.Model Creation

3.1.Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

```
#We have selected churn as the target variable so detection of logistic
regression be on that.
cell$Churn = as.factor(cell$Churn)
cell$ContRenew = as.factor(cell$ContRenew)
cell$Plan = as.factor(cell$Plan)
one<- cell[which(cell$Churn =="1"),]
zero<- cell[which(cell$Churn =="0"),]
training1 <- sample(1:nrow(one),0.7*nrow(one))
training0 <- sample(1:nrow(zero),0.7*nrow(zero))
Final_training1 <- one[training1,]
Final_training0 <- zero[training0,]
trainingData <- rbind(Final_training1, Final_training0)
test1 <- one[-training1,]
test0 <- zero[-training0,]
testData <- rbind(test1, test0)
prop.table(table(testData$Churn))
```

```
## 
##         0         1 
## 0.8551449 0.1448551
```

```r
Model1 <- glm(Churn ~ ., data = trainingData,family =
binomial(link="logit"))
summary(Model1)
```

```
## 
## Call:
## glm(formula = Churn ~ ., family = binomial(link = "logit"), data =
trainingData)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.9959  -0.5080  -0.3419  -0.2015   3.0076  
## 
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)    
## (Intercept)    -5.942045   0.661544  -8.982  < 2e-16 ***
## AccWeeks       -0.000276   0.001665  -0.166  0.86836    
## ContRenew1     -1.994126   0.173088 -11.521  < 2e-16 ***
## Plan1          -1.097446   0.640829  -1.713  0.08680 .  
## Usage           3.713872   2.315801   1.604  0.10878    
## CScalls         0.494678   0.046809  10.568  < 2e-16 ***
## `Min/Day`       0.076251   0.039138   1.948  0.05138 .  
## `Call/Day`      0.003849   0.003307   1.164  0.24451    
## `Charge/Month` -0.365452   0.229898  -1.590  0.11192    
## Over.Fee        0.759696   0.392729   1.934  0.05306 .  
## RoamingMins     0.077428   0.026205   2.955  0.00313 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1508.1  on 2321  degrees of freedom
## AIC: 1530.1
## 
## Number of Fisher Scoring iterations: 6
```

```r
anova(Model1,test = "Chisq")
```

```
## Analysis of Deviance Table
## 
## Model: binomial, link: logit
## 
## Response: Churn
## 
## Terms added sequentially (first to last)
## 
## 
```

```
##                 Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                             2331     1930.1
## AccWeeks         1    0.006      2330     1930.1  0.936623
## ContRenew        1  126.748      2329     1803.3 < 2.2e-16 ***
## Plan             1   31.150      2328     1772.2 2.389e-08 ***
## Usage            1    1.029      2327     1771.2  0.310382
## CScalls          1   98.258      2326     1672.9 < 2.2e-16 ***
## `Min/Day`        1  126.739      2325     1546.2 < 2.2e-16 ***
## `Call/Day`       1    1.218      2324     1545.0  0.269851
## `Charge/Month`   1   23.972      2323     1521.0 9.776e-07 ***
## Over.Fee         1    3.982      2322     1517.0  0.045998 *
## RoamingMins      1    8.905      2321     1508.1  0.002844 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pR2(Model1)
```

```
##          llh       llhNull           G2      McFadden         r2ML
## -754.0494864 -965.0529427  422.0069125    0.2186444    0.1655342
##         r2CU
##    0.2940584
```

```
#checking the multicollinearity
car::vif(Model1)
```

```
##       AccWeeks      ContRenew           Plan          Usage
CScalls
##       1.006944       1.056386      13.858458    1598.838326
1.083021
##      `Min/Day`     `Call/Day` `Charge/Month`       Over.Fee
RoamingMins
##     915.488946       1.002070    2768.195614     214.073261
1.192887
```

```
#the model is affected by :Usage , `Min/Day` , `Charge/Month` ,  Over.Fee

#making the models by removing the above the list variables
mod1 = glm(Churn ~ . -Usage ,data = trainingData,family =
binomial(link="logit"))
summary(mod1)
```

```
##
## Call:
## glm(formula = Churn ~ . - Usage, family = binomial(link = "logit"),
##     data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9995  -0.5030  -0.3412  -0.2035   3.0112
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -6.134660   0.651435  -9.417  < 2e-16 ***
```

```
## AccWeeks       -0.000144   0.001664  -0.087 0.931038
## ContRenew1     -1.987455   0.172900 -11.495  < 2e-16 ***
## Plan1          -0.973956   0.633689  -1.537 0.124303
## CScalls         0.494057   0.046695  10.581  < 2e-16 ***
## `Min/Day`       0.013820   0.003884   3.558 0.000374 ***
## `Call/Day`      0.003828   0.003300   1.160 0.246037
## `Charge/Month`  0.001591   0.021616   0.074 0.941328
## Over.Fee        0.134307   0.045424   2.957 0.003109 **
## RoamingMins     0.079861   0.026181   3.050 0.002286 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1510.7  on 2322  degrees of freedom
## AIC: 1530.7
##
## Number of Fisher Scoring iterations: 6
```

```
anova(mod1,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
##
##
##                Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                            2331     1930.1
## AccWeeks        1    0.006      2330     1930.1   0.93662
## ContRenew       1  126.748      2329     1803.3 < 2.2e-16 ***
## Plan            1   31.150      2328     1772.2 2.389e-08 ***
## CScalls         1   97.703      2327     1674.5 < 2.2e-16 ***
## `Min/Day`       1  126.469      2326     1548.0 < 2.2e-16 ***
## `Call/Day`      1    1.200      2325     1546.8   0.27323
## `Charge/Month`  1   22.439      2324     1524.4 2.169e-06 ***
## Over.Fee        1    4.207      2323     1520.2   0.04025 *
## RoamingMins     1    9.505      2322     1510.7   0.00205 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod2 = glm(Churn ~ . -`Min/Day` ,data = trainingData,family =
binomial(link="logit"))
summary(mod2)
```

```
##
## Call:
## glm(formula = Churn ~ . - `Min/Day`, family = binomial(link = "logit"),
```

```
##     data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9967  -0.5029  -0.3413  -0.2028   3.0058
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -6.1470698  0.6538640  -9.401  < 2e-16 ***
## AccWeeks       -0.0001181  0.0016630  -0.071 0.943377
## ContRenew1     -1.9845677  0.1727693 -11.487  < 2e-16 ***
## Plan1          -1.0593968  0.6392541  -1.657 0.097471 .
## Usage          -0.7775758  0.2298553  -3.383 0.000717 ***
## CScalls         0.4941528  0.0466706  10.588  < 2e-16 ***
## `Call/Day`      0.0038332  0.0032973   1.163 0.245019
## `Charge/Month`  0.0823911  0.0077398  10.645  < 2e-16 ***
## Over.Fee       -0.0033558  0.0293955  -0.114 0.909111
## RoamingMins     0.0784603  0.0261994   2.995 0.002747 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1511.9  on 2322  degrees of freedom
## AIC: 1531.9
##
## Number of Fisher Scoring iterations: 6
```

```
anova(mod2,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
##
##
##                Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                            2331     1930.1
## AccWeeks        1    0.006      2330     1930.1  0.936623
## ContRenew       1  126.748      2329     1803.3 < 2.2e-16 ***
## Plan            1   31.150      2328     1772.2 2.389e-08 ***
## Usage           1    1.029      2327     1771.2  0.310382
## CScalls         1   98.258      2326     1672.9 < 2.2e-16 ***
## `Call/Day`      1    1.406      2325     1671.5  0.235760
## `Charge/Month`  1  150.397      2324     1521.1 < 2.2e-16 ***
## Over.Fee        1    0.042      2323     1521.1  0.838191
## RoamingMins     1    9.158      2322     1511.9  0.002476 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod3 = glm(Churn ~ . -`Charge/Month` ,data = trainingData,family =
binomial(link="logit"))
summary(mod3)
```

```
##
## Call:
## glm(formula = Churn ~ . - `Charge/Month`, family = binomial(link =
"logit"),
##     data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9975  -0.5035  -0.3408  -0.2028   3.0102
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.1234718  0.6524488  -9.385  < 2e-16 ***
## AccWeeks    -0.0001468  0.0016632  -0.088  0.92965
## ContRenew1  -1.9862388  0.1728542 -11.491  < 2e-16 ***
## Plan1       -1.0671811  0.6396459  -1.668  0.09524 .
## Usage        0.0489381  0.2177992   0.225  0.82222
## CScalls      0.4945112  0.0467088  10.587  < 2e-16 ***
## `Min/Day`    0.0140941  0.0013181  10.692  < 2e-16 ***
## `Call/Day`   0.0038354  0.0032997   1.162  0.24508
## Over.Fee     0.1370751  0.0271191   5.055 4.31e-07 ***
## RoamingMins  0.0782991  0.0262027   2.988  0.00281 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1510.6  on 2322  degrees of freedom
## AIC: 1530.6
##
## Number of Fisher Scoring iterations: 6
```

```r
anova(mod3,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
##
##
##             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
```

```
## NULL                          2331     1930.1
## AccWeeks     1    0.006       2330     1930.1  0.936623
## ContRenew    1  126.748       2329     1803.3 < 2.2e-16 ***
## Plan         1   31.150       2328     1772.2 2.389e-08 ***
## Usage        1    1.029       2327     1771.2  0.310382
## CScalls      1   98.258       2326     1672.9 < 2.2e-16 ***
## `Min/Day`    1  126.739       2325     1546.2 < 2.2e-16 ***
## `Call/Day`   1    1.218       2324     1545.0  0.269851
## Over.Fee     1   25.207       2323     1519.8 5.149e-07 ***
## RoamingMins  1    9.116       2322     1510.6  0.002533 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod4 = glm(Churn ~ . -Over.Fee ,data = trainingData,family =
binomial(link="logit"))
summary(mod4)
```

```
##
## Call:
## glm(formula = Churn ~ . - Over.Fee, family = binomial(link = "logit"),
##     data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9954  -0.5037  -0.3410  -0.2031   3.0075
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -6.1211380  0.6555746  -9.337  < 2e-16 ***
## AccWeeks       -0.0001282  0.0016631  -0.077  0.93858
## ContRenew1     -1.9838356  0.1727524 -11.484  < 2e-16 ***
## Plan1          -1.0582243  0.6392634  -1.655  0.09785 .
## Usage          -0.7372190  0.2677779  -2.753  0.00590 **
## CScalls         0.4940484  0.0466715  10.586  < 2e-16 ***
## `Min/Day`       0.0007632  0.0029296   0.261  0.79447
## `Call/Day`      0.0038231  0.0032976   1.159  0.24631
## `Charge/Month`  0.0783494  0.0158639   4.939 7.86e-07 ***
## RoamingMins     0.0783426  0.0262022   2.990  0.00279 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1511.9  on 2322  degrees of freedom
## AIC: 1531.9
##
## Number of Fisher Scoring iterations: 6
```

```r
anova(mod4,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
##
##
##                Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                            2331     1930.1
## AccWeeks         1    0.006      2330     1930.1  0.936623
## ContRenew        1  126.748      2329     1803.3 < 2.2e-16 ***
## Plan             1   31.150      2328     1772.2 2.389e-08 ***
## Usage            1    1.029      2327     1771.2  0.310382
## CScalls          1   98.258      2326     1672.9 < 2.2e-16 ***
## `Min/Day`        1  126.739      2325     1546.2 < 2.2e-16 ***
## `Call/Day`       1    1.218      2324     1545.0  0.269851
## `Charge/Month`   1   23.972      2323     1521.0 9.776e-07 ***
## RoamingMins      1    9.129      2322     1511.9  0.002516 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#predicting the models on the training dataset
pred<-predict(Model1,newdata=trainingData,type="response")
prediction<- ifelse(pred>0.5,1,0)
prediction1 <- factor(prediction, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction1,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1942  269
##          1   52   69
##
##                Accuracy : 0.8623
##                  95% CI : (0.8477, 0.8761)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.166
##
##                   Kappa : 0.2428
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.20414
##             Specificity : 0.97392
##          Pos Pred Value : 0.57025
##          Neg Pred Value : 0.87834
##              Prevalence : 0.14494
##          Detection Rate : 0.02959
```

```
##    Detection Prevalence : 0.05189
##       Balanced Accuracy : 0.58903
##
##        'Positive' Class : 1
##
```

```r
pred1<-predict(mod1,newdata=trainingData,type="response")
prediction1<- ifelse(pred1>0.5,1,0)
prediction2 <- factor(prediction1, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction2,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1946  265
##          1   48   73
##
##                Accuracy : 0.8658
##                  95% CI : (0.8513, 0.8794)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.07367
##
##                   Kappa : 0.2617
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.21598
##             Specificity : 0.97593
##          Pos Pred Value : 0.60331
##          Neg Pred Value : 0.88014
##              Prevalence : 0.14494
##          Detection Rate : 0.03130
##    Detection Prevalence : 0.05189
##       Balanced Accuracy : 0.59595
##
##        'Positive' Class : 1
##
```

```r
pred2<-predict(mod2,newdata=trainingData,type="response")
prediction3<- ifelse(pred2>0.5,1,0)
prediction4 <- factor(prediction3, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction4,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1945  268
##          1   49   70
```

```
##
##                Accuracy : 0.8641
##                  95% CI : (0.8495, 0.8777)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.1133
##
##                   Kappa : 0.2497
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.20710
##             Specificity : 0.97543
##          Pos Pred Value : 0.58824
##          Neg Pred Value : 0.87890
##              Prevalence : 0.14494
##          Detection Rate : 0.03002
##    Detection Prevalence : 0.05103
##       Balanced Accuracy : 0.59126
##
##        'Positive' Class : 1
##
```

```
pred3<-predict(mod3,newdata=trainingData,type="response")
prediction5<- ifelse(pred3>0.5,1,0)
prediction6 <- factor(prediction5, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction6,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1947  266
##          1   47   72
##
##                Accuracy : 0.8658
##                  95% CI : (0.8513, 0.8794)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.07367
##
##                   Kappa : 0.2592
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.21302
##             Specificity : 0.97643
##          Pos Pred Value : 0.60504
##          Neg Pred Value : 0.87980
##              Prevalence : 0.14494
##          Detection Rate : 0.03087
##    Detection Prevalence : 0.05103
##       Balanced Accuracy : 0.59472
```

```
##
##        'Positive' Class : 1
##
```

```r
pred4<-predict(mod4,newdata=trainingData,type="response")
prediction7<- ifelse(pred4>0.5,1,0)
prediction8 <- factor(prediction7, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction8,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1946  268
##          1   48   70
##
##                Accuracy : 0.8645
##                  95% CI : (0.8499, 0.8781)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.1022
##
##                   Kappa : 0.2508
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.20710
##             Specificity : 0.97593
##          Pos Pred Value : 0.59322
##          Neg Pred Value : 0.87895
##              Prevalence : 0.14494
##          Detection Rate : 0.03002
##    Detection Prevalence : 0.05060
##       Balanced Accuracy : 0.59151
##
##        'Positive' Class : 1
##
```

```r
#After training data is tested we can go forward and start with the test data
pred_test<-predict(Model1,newdata=testData,type="response")
prediction_test<- ifelse(pred_test>0.5,1,0)
prediction1_test <- factor(prediction_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction1_test,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 837 127
##          1  19  18
```

```
##
##                  Accuracy : 0.8541
##                    95% CI : (0.8307, 0.8754)
##       No Information Rate : 0.8551
##       P-Value [Acc > NIR] : 0.5577
##
##                     Kappa : 0.1476
##
##   Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.12414
##              Specificity : 0.97780
##           Pos Pred Value : 0.48649
##           Neg Pred Value : 0.86826
##               Prevalence : 0.14486
##           Detection Rate : 0.01798
##     Detection Prevalence : 0.03696
##        Balanced Accuracy : 0.55097
##
##          'Positive' Class : 1
##
```

```
pred1_test<-predict(mod1,newdata=testData,type="response")
prediction1_test<- ifelse(pred1_test>0.5,1,0)
prediction2_test <- factor(prediction1_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction2_test,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 836 126
##          1  20  19
##
##                  Accuracy : 0.8541
##                    95% CI : (0.8307, 0.8754)
##       No Information Rate : 0.8551
##       P-Value [Acc > NIR] : 0.5577
##
##                     Kappa : 0.1546
##
##   Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.13103
##              Specificity : 0.97664
##           Pos Pred Value : 0.48718
##           Neg Pred Value : 0.86902
##               Prevalence : 0.14486
##           Detection Rate : 0.01898
##     Detection Prevalence : 0.03896
##        Balanced Accuracy : 0.55383
```

```
##
##        'Positive' Class : 1
##
```

```
pred2_test<-predict(mod2,newdata=testData,type="response")
prediction3_test<- ifelse(pred2_test>0.5,1,0)
prediction4_test <- factor(prediction3_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction4_test,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 837 126
##          1  19  19
##
##                Accuracy : 0.8551
##                  95% CI : (0.8318, 0.8764)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.5221
##
##                   Kappa : 0.1569
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.13103
##             Specificity : 0.97780
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.86916
##              Prevalence : 0.14486
##          Detection Rate : 0.01898
##    Detection Prevalence : 0.03796
##       Balanced Accuracy : 0.55442
##
##        'Positive' Class : 1
##
```

```
pred3_test<-predict(mod3,newdata=testData,type="response")
prediction5_test<- ifelse(pred3_test>0.5,1,0)
prediction6_test <- factor(prediction5_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction6_test,act,positive="1")
```

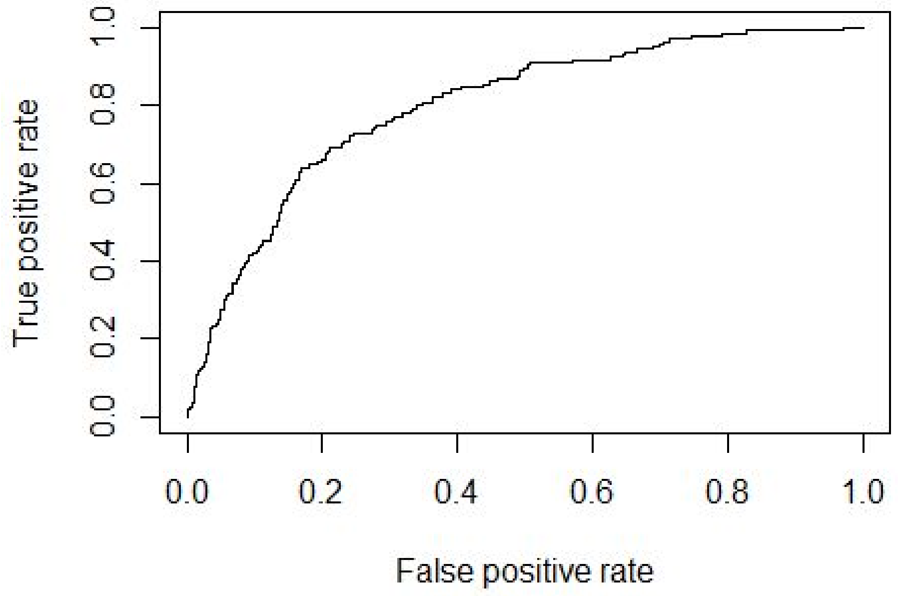
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 836 126
##          1  20  19
##
##                Accuracy : 0.8541
```

```
##                    95% CI : (0.8307, 0.8754)
##       No Information Rate : 0.8551
##       P-Value [Acc > NIR] : 0.5577
##
##                     Kappa : 0.1546
##
##    Mcnemar's Test P-Value : <2e-16
##
##               Sensitivity : 0.13103
##               Specificity : 0.97664
##            Pos Pred Value : 0.48718
##            Neg Pred Value : 0.86902
##                Prevalence : 0.14486
##            Detection Rate : 0.01898
##      Detection Prevalence : 0.03896
##         Balanced Accuracy : 0.55383
##
##          'Positive' Class : 1
##
```

```
pred4_test<-predict(mod4,newdata=testData,type="response")
prediction7_test<- ifelse(pred4_test>0.5,1,0)
prediction8_test <- factor(prediction7_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction8_test,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##            0 836 126
##            1  20  19
##
##                  Accuracy : 0.8541
##                    95% CI : (0.8307, 0.8754)
##       No Information Rate : 0.8551
##       P-Value [Acc > NIR] : 0.5577
##
##                     Kappa : 0.1546
##
##    Mcnemar's Test P-Value : <2e-16
##
##               Sensitivity : 0.13103
##               Specificity : 0.97664
##            Pos Pred Value : 0.48718
##            Neg Pred Value : 0.86902
##                Prevalence : 0.14486
##            Detection Rate : 0.01898
##      Detection Prevalence : 0.03896
##         Balanced Accuracy : 0.55383
##
```

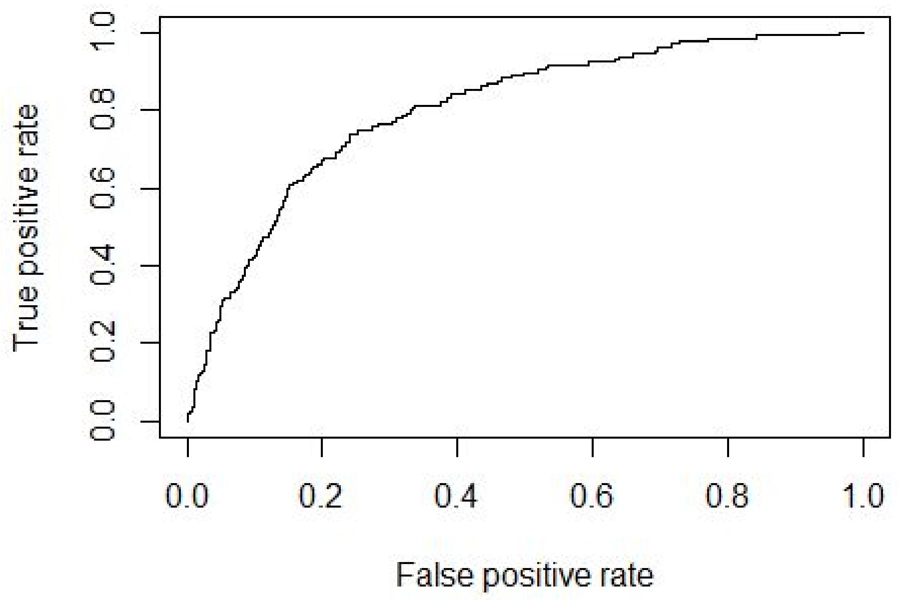
```
##        'Positive' Class : 1
##

#ROC plot and AUC of the different models we have created in Logistic
Regression
rocpred<-predict(Model1,testData,type = 'response')
rocpred<-prediction(pred_test, testData$Churn)
roc<-performance(rocpred,"tpr", "fpr")
plot(roc)
```




```
auc<-performance(rocpred,"auc")
auc

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.7995488
```

```
##
##
## Slot "alpha.values":
## list()
```

```
rocpred_mod1<-predict(mod1,testData,type = 'response')
rocpred_mod1<-prediction(pred1_test, testData$Churn)
roc_mod1<-performance(rocpred_mod1,"tpr", "fpr")
plot(roc_mod1)
```

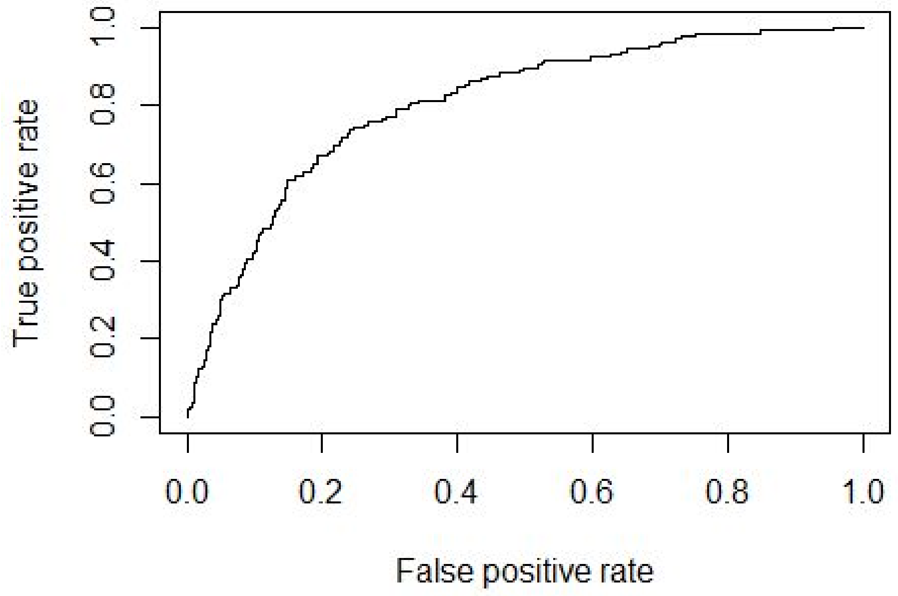



```
auc_mod1<-performance(rocpred_mod1,"auc")
auc_mod1
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8038269
```

```
##
##
## Slot "alpha.values":
## list()
```

```
rocpred_mod2<-predict(mod2,testData,type = 'response')
rocpred_mod2<-prediction(pred2_test, testData$Churn)
roc_mod2<-performance(rocpred_mod2,"tpr", "fpr")
plot(roc_mod2)
```

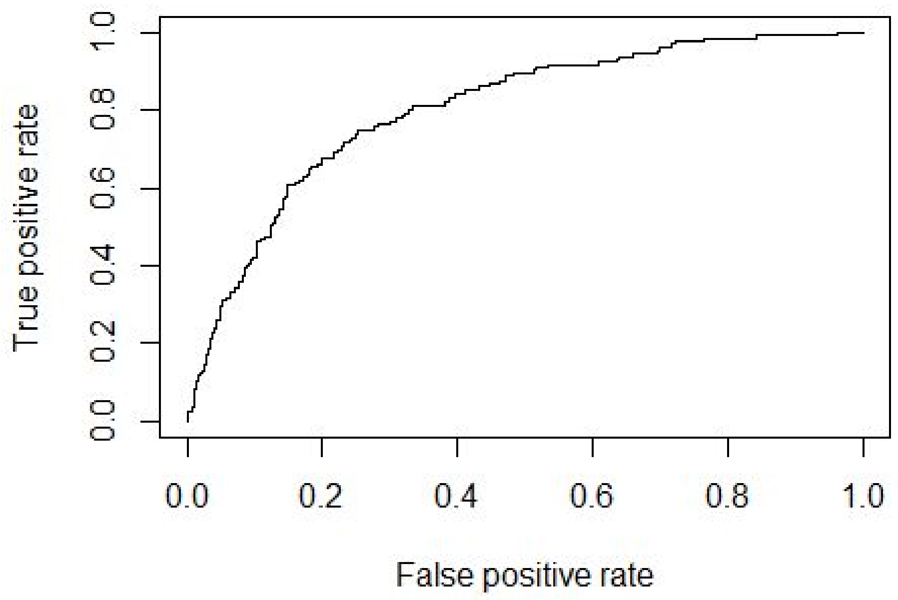



```
auc_mod2<-performance(rocpred_mod2,"auc")
auc_mod2
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8048501
```

```
##
##
## Slot "alpha.values":
## list()
```

```
rocpred_mod3<-predict(mod3,testData,type = 'response')
rocpred_mod3<-prediction(pred3_test, testData$Churn)
roc_mod3<-performance(rocpred_mod3,"tpr", "fpr")
plot(roc_mod3)
```

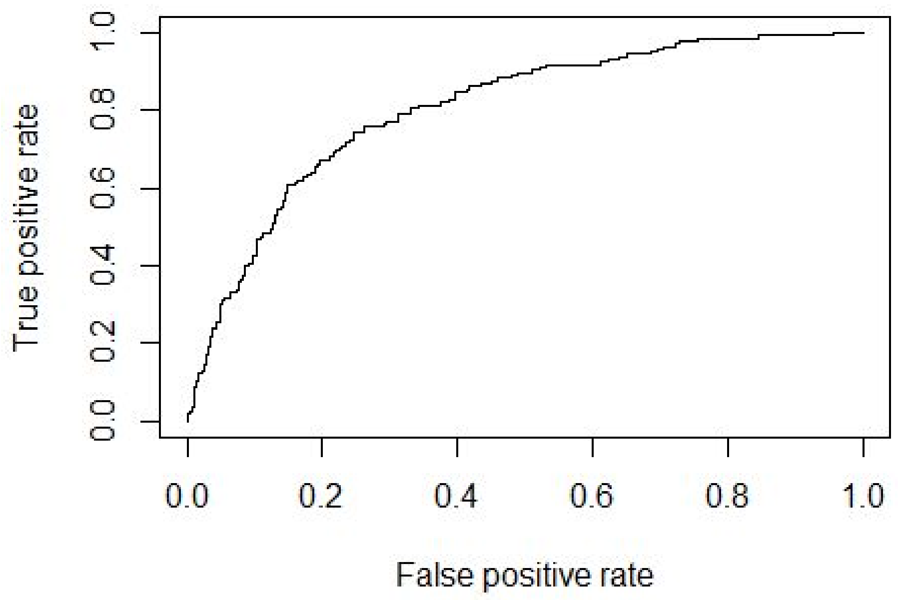



```
auc_mod3<-performance(rocpred_mod3,"auc")
auc_mod3
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.803972
```

```
##
##
## Slot "alpha.values":
## list()
```

```r
rocpred_mod4<-predict(mod4,testData,type = 'response')
rocpred_mod4<-prediction(pred4_test, testData$Churn)
roc_mod4<-performance(rocpred_mod4,"tpr", "fpr")
plot(roc_mod4)
```




```r
auc_mod4<-performance(rocpred_mod4,"auc")
auc_mod4
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8046407
```

```
##
##
## Slot "alpha.values":
## list()
```

So we can see that almost all the graphs are nearly the same for these models. The model1 is having all the elements while the other 4 models doesnot have the elements which were having high collinearity.
#The next model will not be having the highly collinear elements and we would examine the model.

```
Model2 = glm(Churn ~ AccWeeks +  ContRenew + CScalls + `Call/Day` +
RoamingMins ,data = trainingData , family = binomial(link = logit))
summary(Model2)
```

```
##
## Call:
## glm(formula = Churn ~ AccWeeks + ContRenew + CScalls + `Call/Day` +
##     RoamingMins, family = binomial(link = logit), data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5896  -0.5225  -0.4228  -0.3414   2.5721
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.9338635  0.4542735  -4.257 2.07e-05 ***
## AccWeeks    -0.0004651  0.0015919  -0.292  0.77015
## ContRenew1  -1.9234541  0.1593943 -12.067  < 2e-16 ***
## CScalls      0.4277921  0.0429859   9.952  < 2e-16 ***
## `Call/Day`   0.0035614  0.0031207   1.141  0.25378
## RoamingMins  0.0667935  0.0229942   2.905  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.1  on 2331  degrees of freedom
## Residual deviance: 1694.2  on 2326  degrees of freedom
## AIC: 1706.2
##
## Number of Fisher Scoring iterations: 5
```

```
anova(Model2,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Churn
##
## Terms added sequentially (first to last)
```

```
##
##
##             Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                         2331     1930.1
## AccWeeks     1    0.006      2330     1930.1  0.93662
## ContRenew    1  126.748      2329     1803.3  < 2e-16 ***
## CScalls      1   99.065      2328     1704.3  < 2e-16 ***
## `Call/Day`   1    1.471      2327     1702.8  0.22513
## RoamingMins  1    8.568      2326     1694.2  0.00342 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#Checking the multi collinearity
car::vif(Model2)
```

```
##    AccWeeks   ContRenew     CScalls  `Call/Day` RoamingMins
##    1.001538    1.033876    1.034264    1.001396    1.002103
```

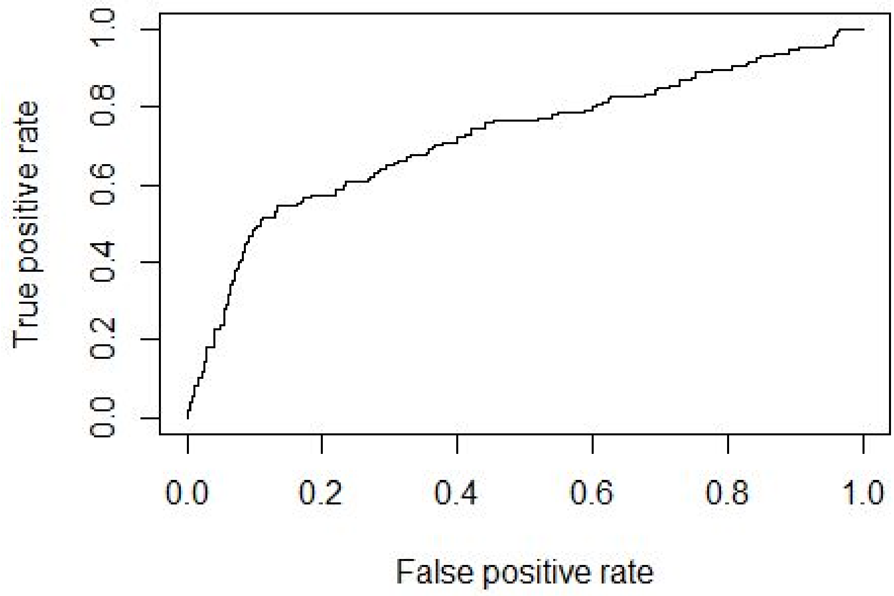
```r
#Prediction using the model for the training dataset
pred_Model2<-predict(Model2,newdata=trainingData,type="response")
prediction_Model2<- ifelse(pred_Model2>0.5,1,0)
prediction1_Model2 <- factor(prediction_Model2, levels=c(0,1))
act <- trainingData$Churn
confusionMatrix(prediction1_Model2,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1962  295
##          1   32   43
##
##                Accuracy : 0.8598
##                  95% CI : (0.845, 0.8736)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.2698
##
##                   Kappa : 0.1642
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.12722
##             Specificity : 0.98395
##          Pos Pred Value : 0.57333
##          Neg Pred Value : 0.86930
##              Prevalence : 0.14494
##          Detection Rate : 0.01844
##    Detection Prevalence : 0.03216
##       Balanced Accuracy : 0.55559
##
##        'Positive' Class : 1
##
```

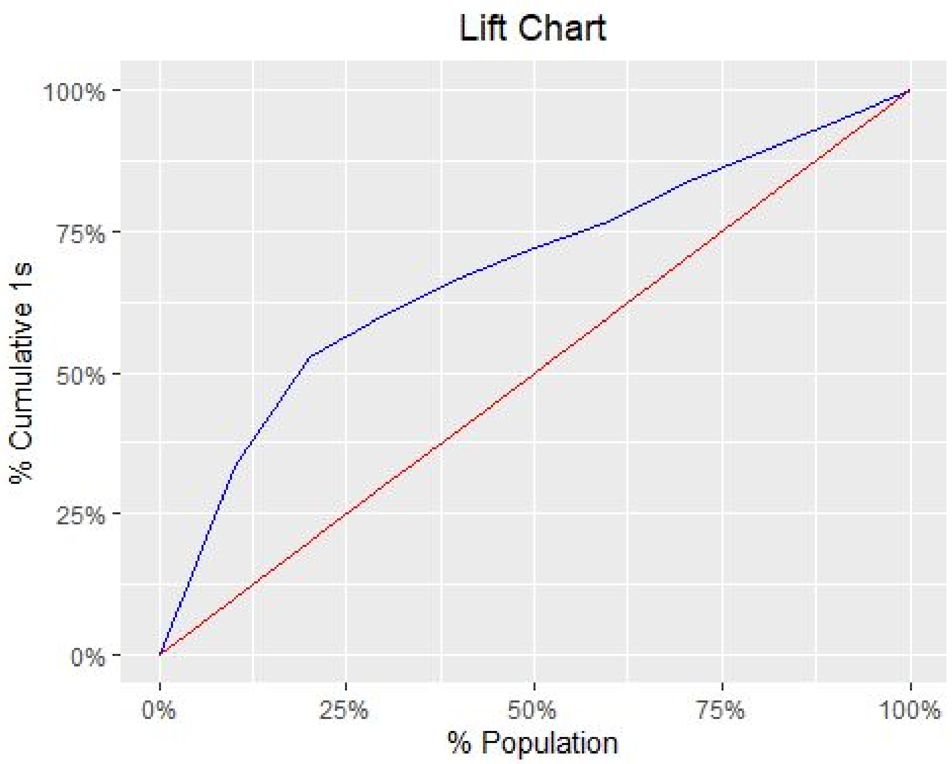
```r
#Prediction using the test dataset
pred_Model2_test<-predict(Model2,newdata=testData,type="response")
prediction_Model2_test<- ifelse(pred_Model2_test>0.5,1,0)
prediction1_Model2_test <- factor(prediction_Model2_test, levels=c(0,1))
act <- testData$Churn
confusionMatrix(prediction1_Model2_test,act,positive="1")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 843 132
##          1  13  13
##
##                Accuracy : 0.8551
##                  95% CI : (0.8318, 0.8764)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.5221
##
##                   Kappa : 0.113
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.08966
##             Specificity : 0.98481
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.86462
##              Prevalence : 0.14486
##          Detection Rate : 0.01299
##    Detection Prevalence : 0.02597
##       Balanced Accuracy : 0.53723
##
##        'Positive' Class : 1
##
```

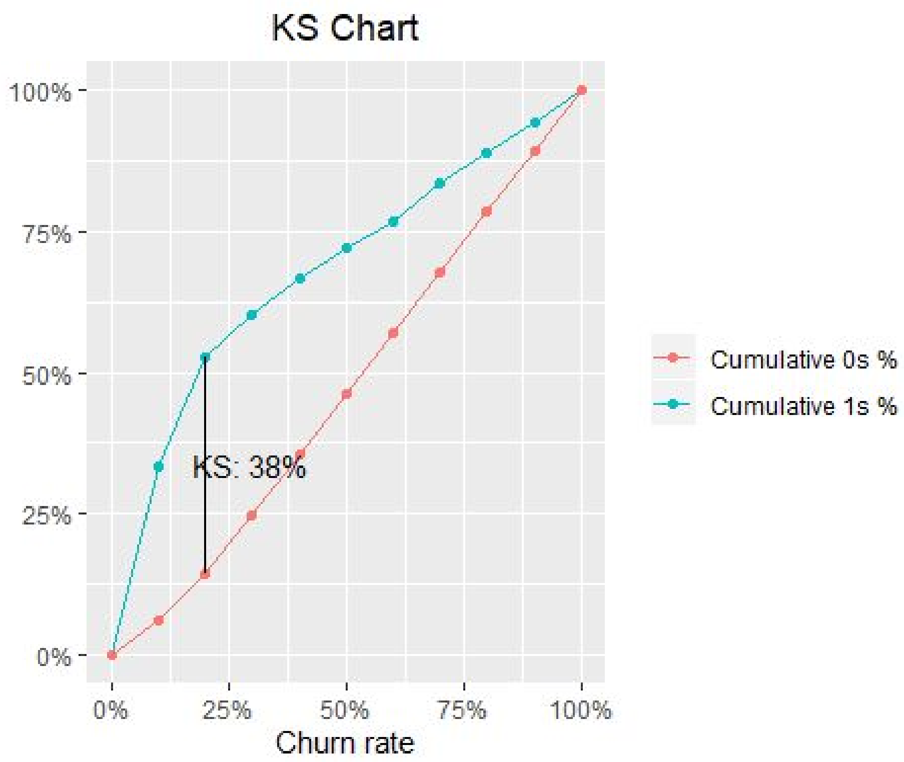
```r
#ROC Curve and AUC
rocpred_Model2<-predict(Model2,testData,type = 'response')
rocpred_Model2<-prediction(pred_Model2_test, testData$Churn)
roc_Model2<-performance(rocpred_Model2,"tpr", "fpr")
plot(roc_Model2)
```



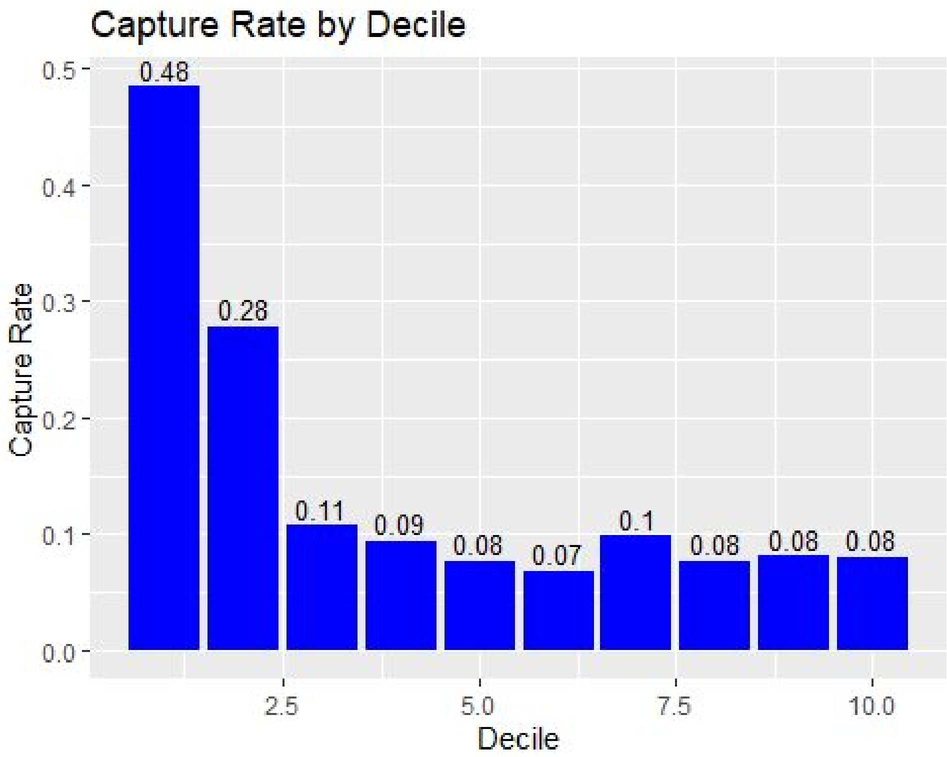

```
auc<-performance(rocpred_Model2,"auc")
auc

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.7214953
##
##
## Slot "alpha.values":
## list()

k = blr_gains_table(Model2)
plot(k)
```



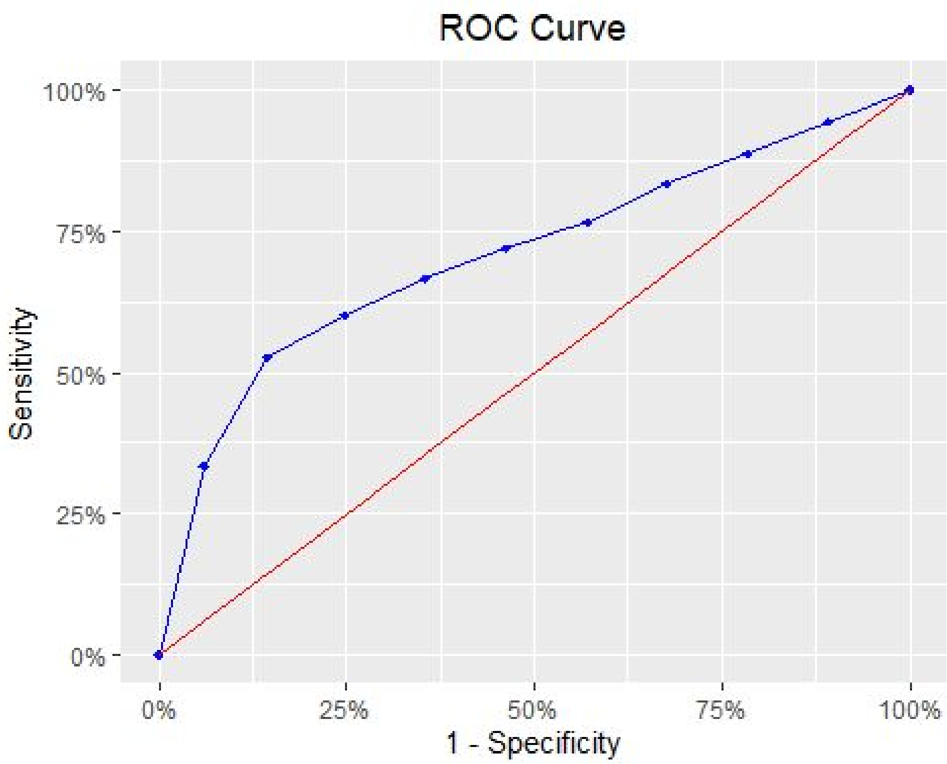

```r
blr_ks_chart(k, title = "KS Chart",
          yaxis_title = " ",xaxis_title = "Churn rate",
          ks_line_color = "black")
```



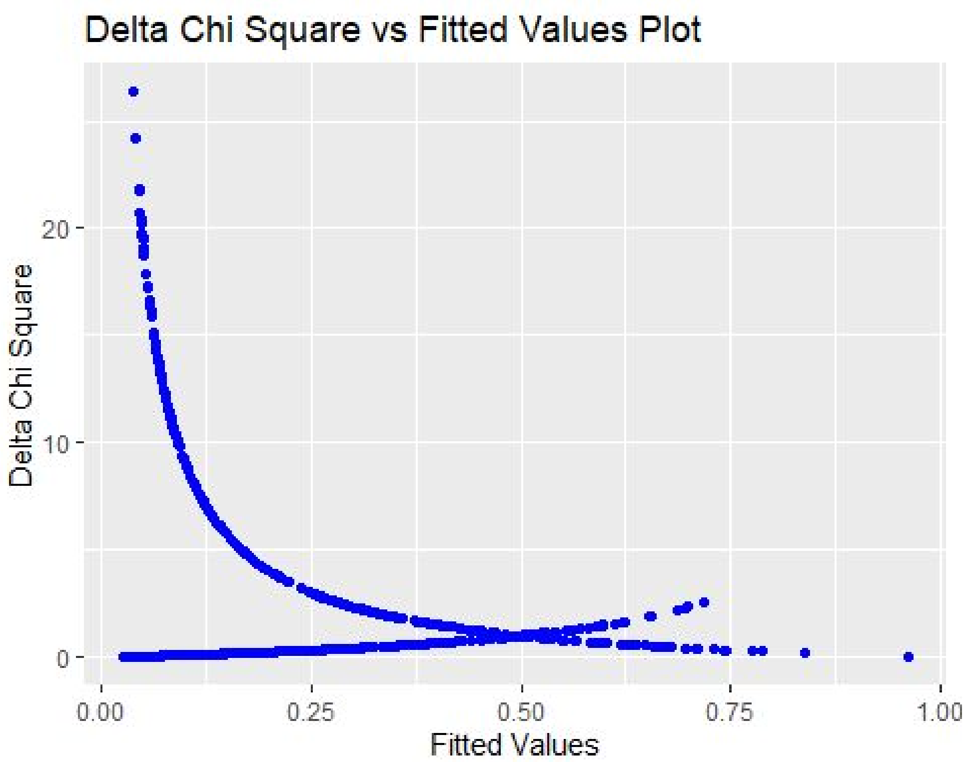
```r
blr_decile_lift_chart(k, xaxis_title = "Decile",
                      yaxis_title = "Decile Mean / Global Mean",
                      title = "Decile Lift Chart",
                      bar_color = "blue", text_size = 3.5,
                      text_vjust = -0.3)
```




```r
blr_roc_curve(k, title = "ROC Curve",
              xaxis_title = "1 - Specificity",
              yaxis_title = "Sensitivity",roc_curve_col = "blue",
              diag_line_col = "red", point_shape = 18,
              point_fill = "blue", point_color = "blue",
              plot_title_justify = 0.5)
```




```
blr_plot_difchisq_fitted(Model2, point_color = "blue",
                         title = "Delta Chi Square vs Fitted Values Plot",
                         xaxis_title = "Fitted Values",
                         yaxis_title = "Delta Chi Square")
```



3.2.K-Nearest Neighbour(KNN) :-

The k-nearest neighbors (**KNN**) algorithm is a simple, easy-to-implement supervised **machine learning** algorithm that can be used to solve both classification and regression problems.

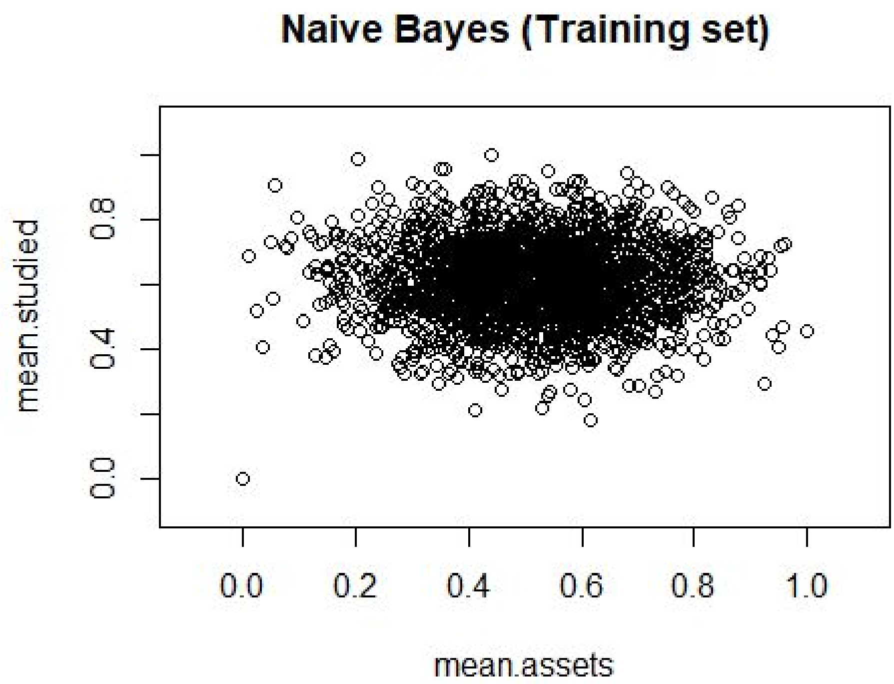
```
cell$ContRenew = as.numeric(cell$ContRenew)
cell$Plan = as.numeric(cell$Plan)

#Normalizing the data
normailize<-function(x){ return((x-min(x))/(max(x)-min(x)))
}

cell.norm<-as.data.frame(lapply(cell[,-1],normailize ))
#View(cell.norm)
usable.data = cbind(cell[,1], cell.norm)
str(usable.data)

## 'data.frame':    3333 obs. of  11 variables:
##  $ Churn        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AccWeeks     : num  0.525 0.438 0.562 0.343 0.306 ...
##  $ ContRenew    : num  1 1 1 0 0 0 1 0 1 0 ...
##  $ Plan         : num  1 1 0 0 0 0 1 0 0 1 ...
##  $ Usage        : num  0.5 0.685 0 0 0 ...
##  $ CScalls      : num  0.111 0.111 0 0.222 0.333 ...
##  $ Min.Day      : num  0.756 0.461 0.694 0.853 0.475 ...
##  $ Call.Day     : num  0.667 0.745 0.691 0.43 0.685 ...
##  $ Charge.Month: num  0.771 0.699 0.391 0.442 0.277 ...
##  $ Over.Fee     : num  0.543 0.538 0.333 0.17 0.408 ...
##  $ RoamingMins : num  0.5 0.685 0.61 0.33 0.505 0.315 0.375 0.355 0.435
0.56 ...

#View(usable.data)
# Data partitioning
spl = sample.split(usable.data$Churn, SplitRatio = 0.7)
train = subset(usable.data, spl == T)
test = subset(usable.data, spl == F)
dim(train)

## [1] 2333   11

dim(test)

## [1] 1000   11

pred_knn_5 = knn(train[-1], test[-1], train[,1], k = 5)
table.knn_5 = table(test[,1], pred_knn_5)
accuracy_knn_5 = sum(diag(table.knn_5)/sum(table.knn_5))
accuracy_knn_5

## [1] 0.904
```

```
loss.knn.5<-table.knn_5[2,1]/(table.knn_5[2,1]+table.knn_5[1,1])
loss.knn.5

## [1] 0.09061489

pred_knn_9 = knn(train[-1], test[-1], train[,1], k = 9)
table.knn_9 = table(test[,1], pred_knn_9)
accuracy_knn_9 = sum(diag(table.knn_9)/sum(table.knn_9))
accuracy_knn_9

## [1] 0.898

loss.knn.9<-table.knn_9[2,1]/(table.knn_9[2,1]+table.knn_9[1,1])
loss.knn.9

## [1] 0.0981857
```

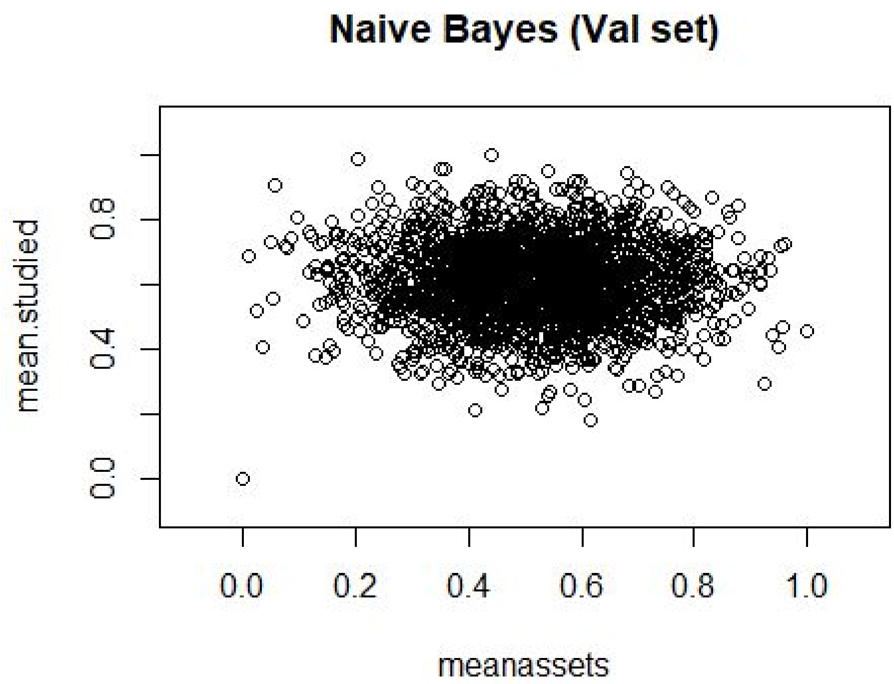
3.3.Naive Bayes

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

```
train.2fact = train[,c(7,8,9)]
val.2fact = train[,c(7,8,9)]
set = train.2fact
X1 = seq(min(set[, 1])-0.1 , max(set[, 1])+0.1 , by = 0.005)
X2 = seq(min(set[, 2]) -0.1, max(set[, 2])+0.1 , by = 0.005)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('mean.assets', 'mean.studied')
plot(set[, -3],
     main = 'Naive Bayes (Training set)',
     xlab = 'mean.assets', ylab = 'mean.studied',
     xlim = range(X1), ylim = range(X2))
```




```
set = val.2fact
X1 = seq(min(set[, 1])-0.1 , max(set[, 1])+0.1 , by = 0.005)
X2 = seq(min(set[, 2]) -0.1, max(set[, 2])+0.1 , by = 0.005)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('mean.assets', 'mean.studied')
plot(set[, -3],
     main = 'Naive Bayes (Val set)',
     xlab = 'meanassets', ylab = 'mean.studied',
     xlim = range(X1), ylim = range(X2))
```

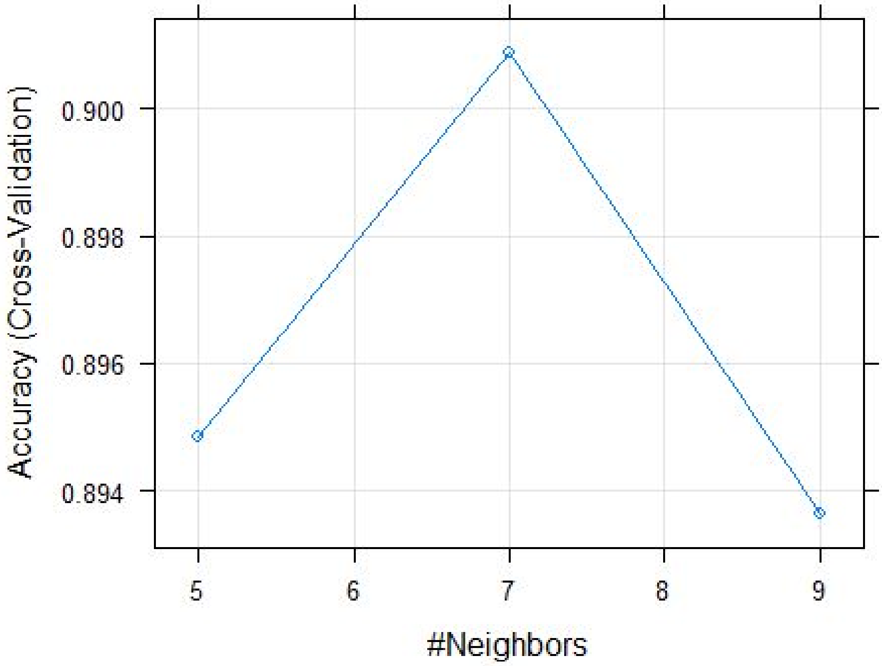



```
NB = naiveBayes(Churn ~., data = train)
predNB = predict(NB, test, type = "class")
tab.NB = table(test[,1], predNB)
tab.NB

##     predNB
##        0   1
##   0 801  54
##   1  84  61

accuracy_NB = sum(diag(tab.NB)/sum(tab.NB))
accuracy_NB

## [1] 0.862

#Confusion Matrix
confusion.matrix(predNB,test$Churn)

## Warning in Ops.factor(pred, threshold): '>=' not meaningful for factors

## Warning in Ops.factor(pred, threshold): '<' not meaningful for factors

##      obs
## pred   0  1
##    0 801 54
##    1  84 61
## attr(,"class")
## [1] "confusion.matrix"
```

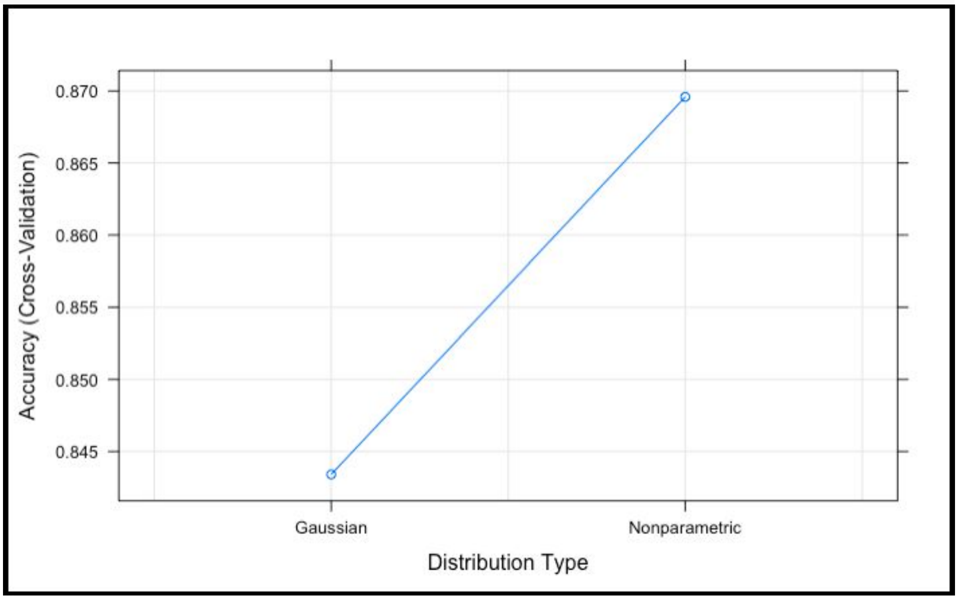
4.Model Performance

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.
Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models.
There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.

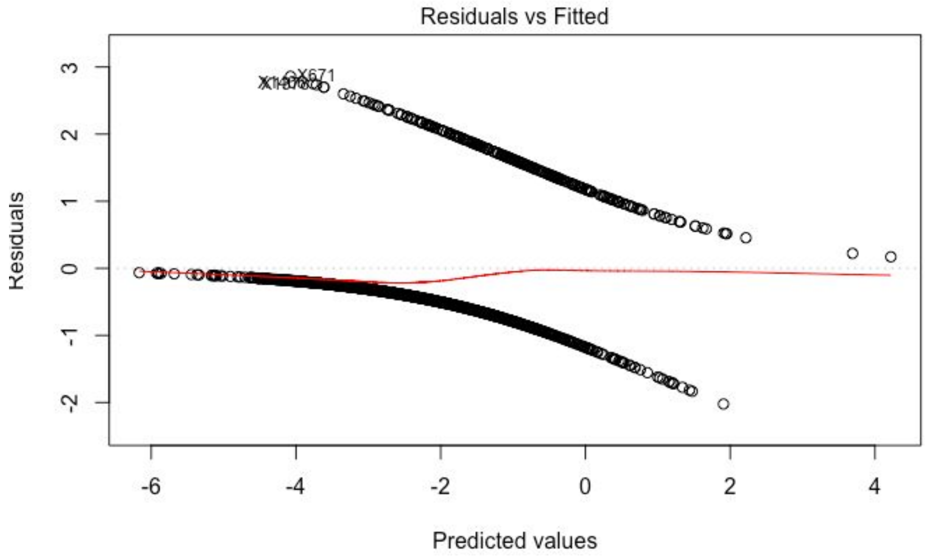
```
splitSample <- sample(1:2, size = nrow(cell), prob = c(0.7,
0.3), replace = T)

train_set <- cell[splitSample == 1, ]

intrain <- sample(1:2, size = nrow(train_set), prob =
c(0.7, 0.3), replace = T)

trainset <- train_set[intrain == 1, ]

validset <- train_set[intrain == 2, ]

testset <- cell[splitSample == 2, ]

#cross validation of the data
tcontrol <- trainControl(method = "cv", number = 10)
set.seed(1234)

# KNN
modelKNN <- train(Churn ~ ., data = trainset, method =
"knn", preProcess = c("center",
    "scale"), trControl = tcontrol)  # data is normalised
using Preprocess
par(mfrow = c(1,2))
plot(modelKNN)
```

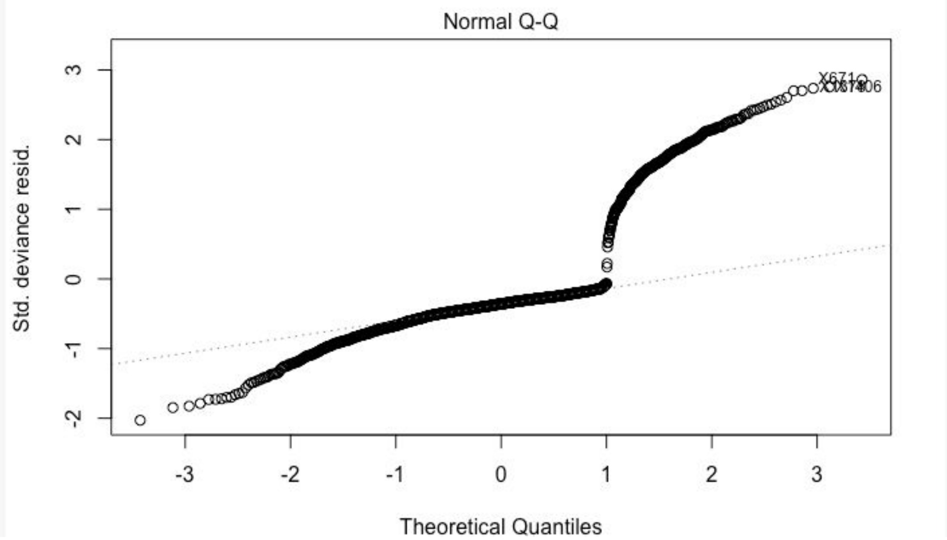


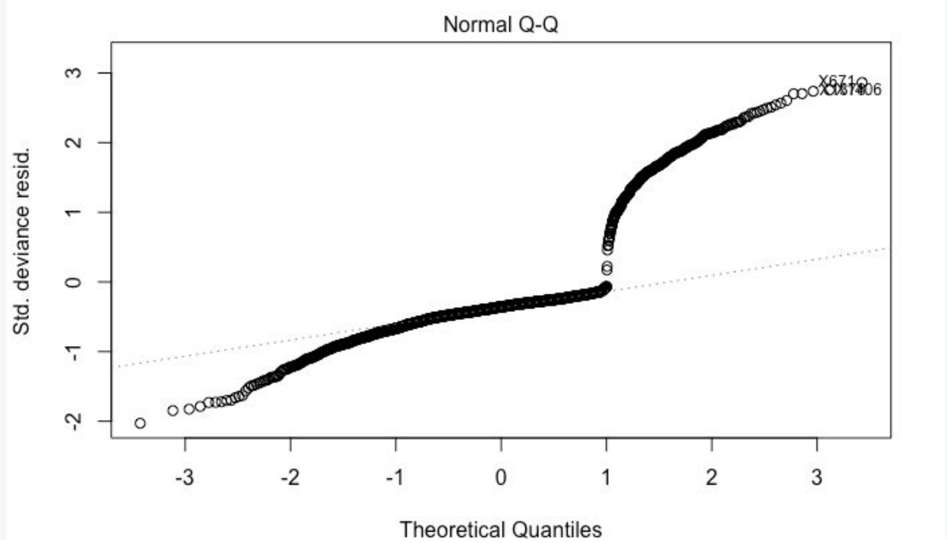

```
modelNB <- train(Churn ~ ., data = trainset, method = "nb",
trControl = tcontrol)
```

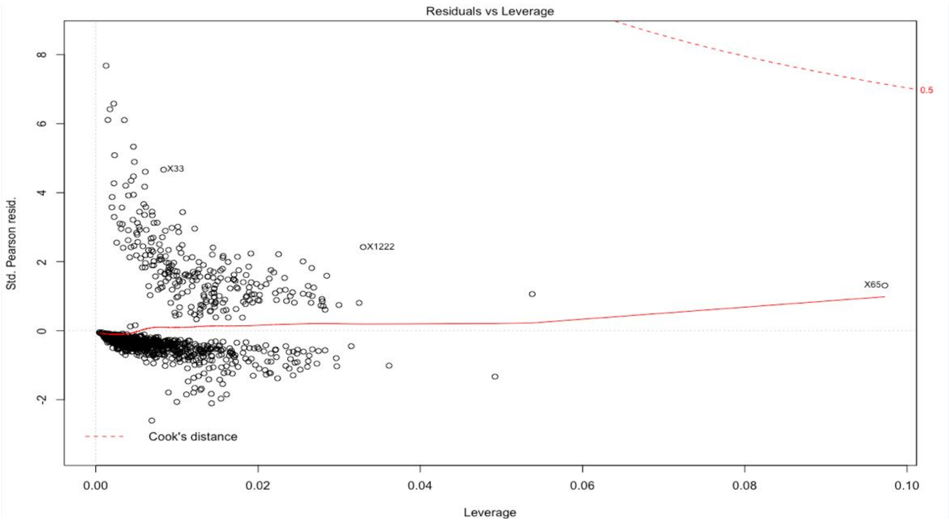



```
modelLG <- train(Churn ~ ., data = trainset, method = "glm", family = binomial,
  trControl = tcontrol)
plot(modelLG$finalModel)
```


Residuals vs Fitted
X671
Residuals
Predicted values
-6
-4
-2
0
2
4
3
2
1
0
-1
-2


Normal Q-Q
X671
Std. deviance resid.
Theoretical Quantiles
-3
-2
-1
0
1
2
3


Normal Q-Q
X671
Std. deviance resid.
Theoretical Quantiles
-3
-2
-1
0
1
2
3




```
# KNN
pKNN = predict(modelKNN, validset)
pKNN_test = predict(modelKNN, testset)
# Naive Bayes
pNB = predict(modelNB, validset)
pNB_test = predict(modelNB, testset)
# Logistic Regression
pLG = predict(modelLG, validset)
pLG_test = predict(modelLG, testset)
```

```
# KNN
cmKNN = confusionMatrix(validset$Churn, pKNN)
cmKNN_test = confusionMatrix(testset$Churn, pKNN_test)

# Naive Bayes
cmNB = confusionMatrix(validset$Churn, pNB)
cmNB_test = confusionMatrix(testset$Churn, pNB_test)
# Logisitic Regression
cmLG <- confusionMatrix(validset$Churn, pLG)
cmLG = confusionMatrix(testset$Churn, pLG_test)

ModelType <- c("K nearest neighbor", "Naive Bayes",
"Logistic regression")

# classification accuracy
TrainAccuracy <- c(max(modelKNN$results$Accuracy),
max(modelNB$results$Accuracy),
max(modelLG$results$Accuracy))

# Training misclassification error
Train_missclass_Error <- 1 - TrainAccuracy

# validation classification accuracy
ValidationAccuracy <- c(cmKNN$overall[1], cmNB$overall[1],
    cmLG$overall[1])
```

```r
# Validation misclassification error or out-of-sample-error
Validation_missclass_Error <- 1 - ValidationAccuracy

metrics <- data.frame(ModelType, TrainAccuracy,
Train_missclass_Error, ValidationAccuracy,
    Validation_missclass_Error)  # data frame with above
metrics

knitr::kable(metrics, digits = 5)  # print table using
kable() from knitr package
```

ModelType	TrainAccuracy	Train_missclass_Error	ValidationAccuracy	Validation_missclass_Error
K nearest neighbor	0.90089	0.09911	0.89787	0.10213
Naive Bayes	0.87312	0.12688	0.88227	0.11773
Logistic regression	0.85621	0.14379	0.86845	0.13155

Accuracy and Sensitivity is relatively high for KNN among the above methods. Yet, insights from logistic regression model can still be utilized to assist decision makers.

An organization loses its customers to its competition for various reasons. Churn can affect the company’s overall growth.The list of the factors that affect the most are listed as :
1.Contract Renewal
2. Data Plan
3.Customer Service Calls
4.Day Mins
5.Overage Fee
6.Roaming Minutes

Keeping a strict check on the above factors we can reduce the churn rate amongst the customers.