Cellphone Customer Churn Report

2019

Sep 10, 2019

Cellphone Customer Churn Report

Authored By: Deepti Lobo

CONTENTS

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.

Known Facts

The dataset has data on 3333 customers. 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. Among these 3333 customers, only 483 (15%) who have canceled their service.

Exploratory Data Analysis (EDA)

The given dataset consists of 3333 observations and 11 variables.

```
#Names of the columns
names(cellphone)

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

Attributes Details

Binary Variables:

- **Churn** - 1 if customer cancelled service, 0 if not. About 85% have not cancelled, while only 15% have cancelled the services. We will consider this as our target variable.
- **Contract Renewal** - 1 if customer recently renewed contract, 0 if not. Around 90% of the customers have renewed the contract, while 10% haven’t.
- **Data Plan** - 1 if customer has data plan, 0 if not. Only 28% have taken the data plan, while the remaining 72% haven’t taken any plan.

Ordinal variables:

- **CustServCalls** - Number of calls into customer service. The highest no of customers of around 1181 have done only 1 call to the customer service. While around 35 have done more than 6 calls, with 9 number of calls by any customer being the highest.

Interval variables:

- **AccountWeeks** - Number of weeks customer has had active account. The least active account was for 1 week. The highest was for 243 weeks. While the average was 101 weeks.
- **DataUsage** - Gigabytes of monthly data usage. The max monthly usage was 5.4GHz while average usage was only 0.8GHz.
- **DayMins** - Average daytime minutes per month. The max was around 350 min while min was 0. With an average of 79.8 min.
- **DayCalls** - Average number of daytime calls. Average calls made was 100 with the max up to 165 calls.
- **MonthlyCharge** - Average monthly bill. Max monthly charges was 111 while the min was 14.
- **OverageFee** - Largest overage fee in last 12 months. The largest fee was 18.19 while the average was 10.05.
- **RoamMins** - Average number of roaming minutes. Max roaming min was 20 min while the average was 10.24 min.

```
#Display the first six rows
head(cellphone)

##   Churn AccountWeeks ContractRenewal DataPlan DataUsage CustServCalls
## 1     0          128               1        1       2.7             1
## 2     0          107               1        1       3.7             1
## 3     0          137               1        0       0.0             0
## 4     0           84               0        0       0.0             2
## 5     0           75               0        0       0.0             3
## 6     0          118               0        0       0.0             0
##   DayMins DayCalls MonthlyCharge OverageFee RoamMins
## 1   265.1      110            89       9.87     10.0
## 2   161.6      123            82       9.78     13.7
## 3   243.4      114            52       6.06     12.2
## 4   299.4       71            57       3.10      6.6
## 5   166.7      113            41       7.42     10.1
## 6   223.4       98            57      11.03      6.3
```

```r
#Is there any values missing?

anyNA(cellphone)

## [1] FALSE
```

There are no values missing in this dataset.

Descriptive Analysis

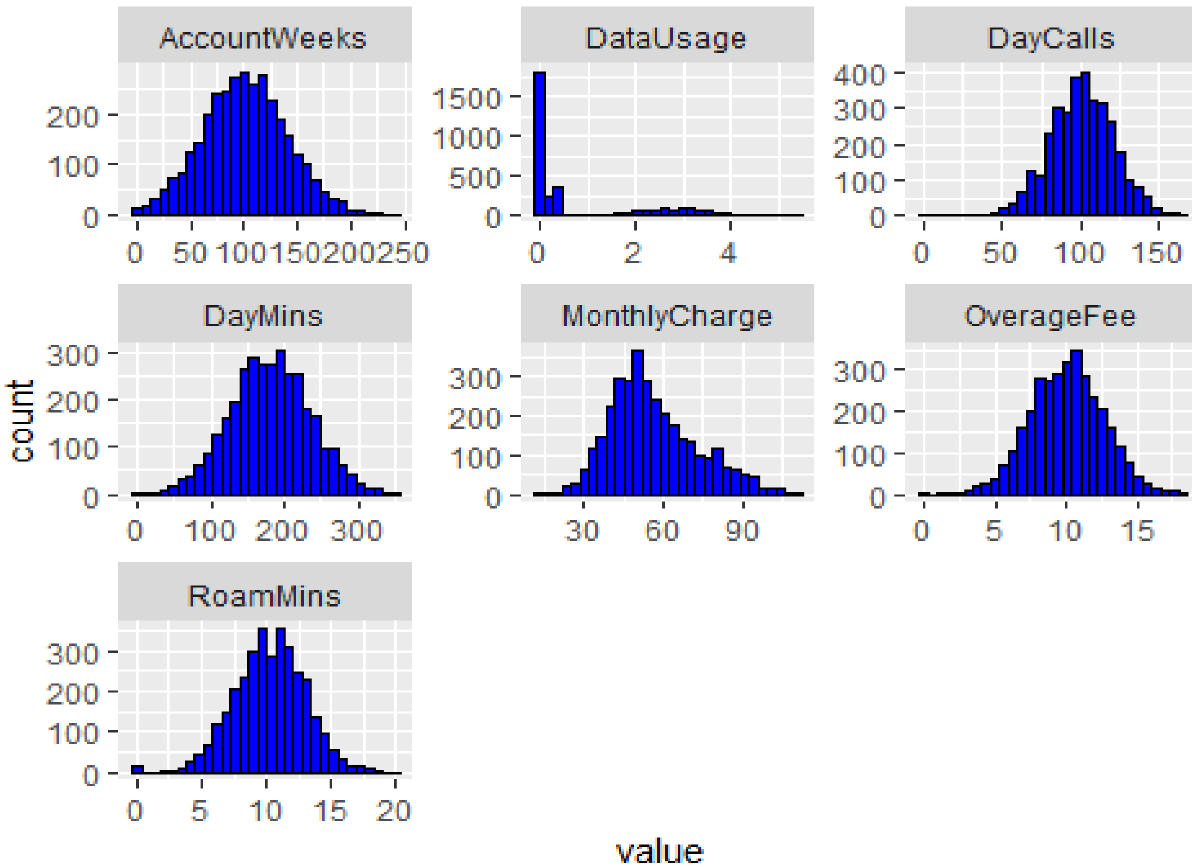
```r
#Data types of all the columns
str(cellphone)

## '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 ...

#Converting Churn,Contract Renewal, Data Plan and CustServCalls into factors.
cellphone$Churn = as.factor(cellphone$Churn)
cellphone$ContractRenewal = as.factor(cellphone$ContractRenewal)
cellphone$DataPlan = as.factor(cellphone$DataPlan)
cellphone$CustServCalls = as.factor(cellphone$CustServCalls)


#summary of the dataset
summary(cellphone)

##  Churn     AccountWeeks   ContractRenewal DataPlan   DataUsage
##  0:2850   Min.   :  1.0   0: 323          0:2411   Min.   :0.0000
##  1: 483   1st Qu.: 74.0   1:3010          1: 922   1st Qu.:0.0000
##           Median :101.0                            Median :0.0000
##           Mean   :101.1                            Mean   :0.8165
##           3rd Qu.:127.0                            3rd Qu.:1.7800
##           Max.   :243.0                            Max.   :5.4000
##
##  CustServCalls     DayMins          DayCalls      MonthlyCharge
##  1      :1181   Min.   :  0.0   Min.   :  0.0   Min.   : 14.00
##  2      : 759   1st Qu.:143.7   1st Qu.: 87.0   1st Qu.: 45.00
##  0      : 697   Median :179.4   Median :101.0   Median : 53.50
##  3      : 429   Mean   :179.8   Mean   :100.4   Mean   : 56.31
##  4      : 166   3rd Qu.:216.4   3rd Qu.:114.0   3rd Qu.: 66.20
```

```
## 5      :  66   Max.   :350.8   Max.   :165.0   Max.   :111.30
## (Other):  35
##    OverageFee       RoamMins
## Min.   : 0.00   Min.   : 0.00
## 1st Qu.: 8.33   1st Qu.: 8.50
## Median :10.07   Median :10.30
## Mean   :10.05   Mean   :10.24
## 3rd Qu.:11.77   3rd Qu.:12.10
## Max.   :18.19   Max.   :20.00
##

##Summary Statistics Measure of central tendency and dispersion (Univariate A
nalysis)
describe(cellphone[,-c(1,3,4,6)],na.rm = TRUE,
         quant = c(0.01,0.05,0.10,0.25,0.75,0.90,0.95,0.99),IQR=TRUE,check=TR
UE)

##              vars    n   mean    sd median trimmed   mad min    max
## AccountWeeks    1 3333 101.06 39.82 101.00  100.77 40.03   1 243.00
## DataUsage       2 3333   0.82  1.27   0.00    0.58  0.00   0   5.40
## DayMins         3 3333 179.78 54.47 179.40  179.85 53.82   0 350.80
## DayCalls        4 3333 100.44 20.07 101.00  100.57 19.27   0 165.00
## MonthlyCharge   5 3333  56.31 16.43  53.50   55.22 15.57  14 111.30
## OverageFee      6 3333  10.05  2.54  10.07   10.05  2.55   0  18.19
## RoamMins        7 3333  10.24  2.79  10.30   10.28  2.67   0  20.00
##               range  skew kurtosis   se   IQR Q0.01 Q0.05   Q0.1  Q0.25
## AccountWeeks  242.00  0.10    -0.11 0.69 53.00 12.32 35.00  50.00  74.00
## DataUsage       5.40  1.27     0.04 0.02  1.78  0.00  0.00   0.00   0.00
## DayMins       350.80 -0.03    -0.02 0.94 72.70 51.83 89.92 110.32 143.70
## DayCalls      165.00 -0.11     0.24 0.35 27.00 54.00 67.00  74.20  87.00
## MonthlyCharge  97.30  0.59    -0.02 0.28 21.20 26.00 33.26  38.00  45.00
## OverageFee     18.19 -0.02     0.02 0.04  3.44  3.98  5.94   6.84   8.33
## RoamMins       20.00 -0.24     0.60 0.05  3.60  3.33  5.70   6.70   8.50
##               Q0.75  Q0.9  Q0.95  Q0.99
## AccountWeeks  127.00 152.00 167.00 195.00
## DataUsage       1.78   3.05   3.46   4.10
## DayMins       216.40 249.58 270.74 305.17
## DayCalls      114.00 126.00 133.00 146.00
## MonthlyCharge  66.20  80.50  87.80  98.28
## OverageFee     11.77  13.29  14.22  15.95
## RoamMins       12.10  13.70  14.70  16.67
```

Data Visualization

```
#Histogram for numerical variables (Univariate Analysis)

cellphone[] %>% keep(is.numeric) %>% gather() %>%
```

```r
ggplot(aes(value)) + facet_wrap(~key, scales = "free") + geom_histogram(color = "black", fill = "blue")
```



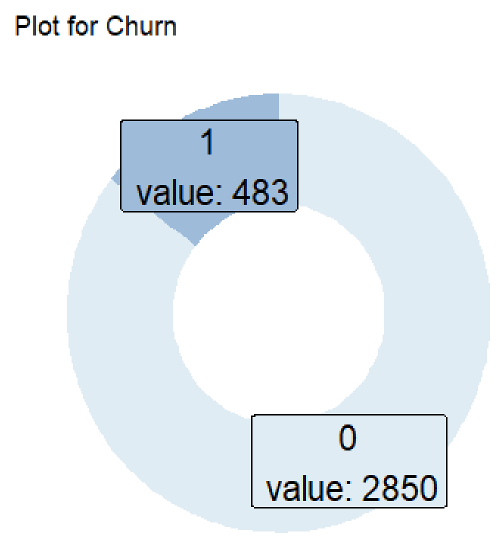
Observation

- **Account Weeks** - Is normally distributed. Most of the customers fall between the range of 0 to 200.
- **Data Usage** - shows that majority of the people do not use the data but the ones who use has a normal distribution.
- **Day Calls** - Is normally distributed. Most of the customers fall between the range of 50 to 150. With a max call of 165.
- **Day Mins** - Is normally distributed. Most of the customers fall between the range of 50 to 300. With a max min of 350.
- **Overage Fee** - Is normally distributed. Most of the customers fall between the range of 5 to 15. With a max min of 18.19.
- **Monthly Charge** - Is positively skewed, with an average income of 56.31 and max of 111.30.

- **Roam Mins –** Is normally distributed. Most of the customers fall between the range of 5 to 15. With a max min of 20.

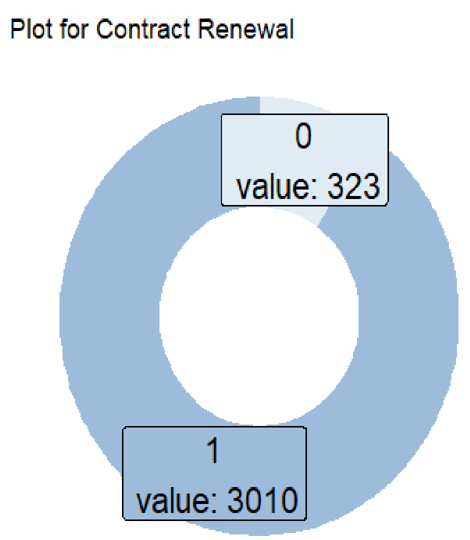
```
##Frequency distribution for categorical variable (Univariate Analysis)
```

Plot for Churn



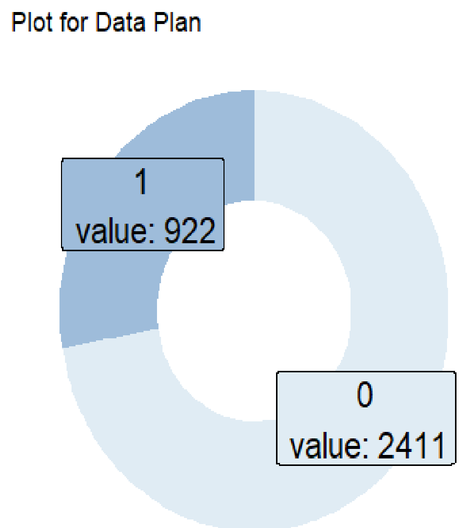
From the plot we can see that about 85% have not cancelled, while only 15% have cancelled the services (i.e) customers having the value 1 is 483 while having 0 is 2850.

Plot for Contract Renewal



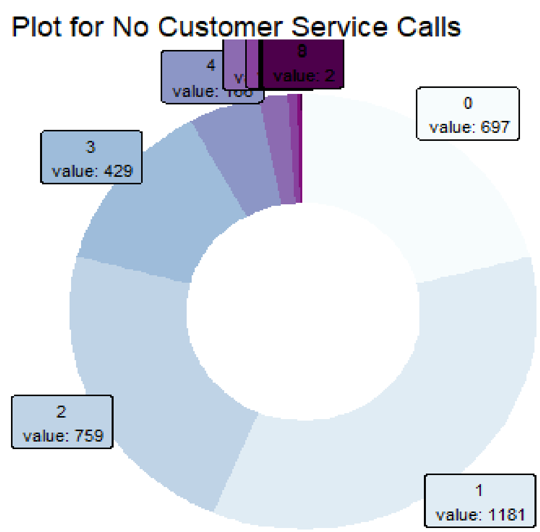
From the graph we can see that around 90% of the customers have renewed the contract, while 10% haven't. Customers having the value 1 is 3010 while having 0 is 323.

Plot for Data Plan



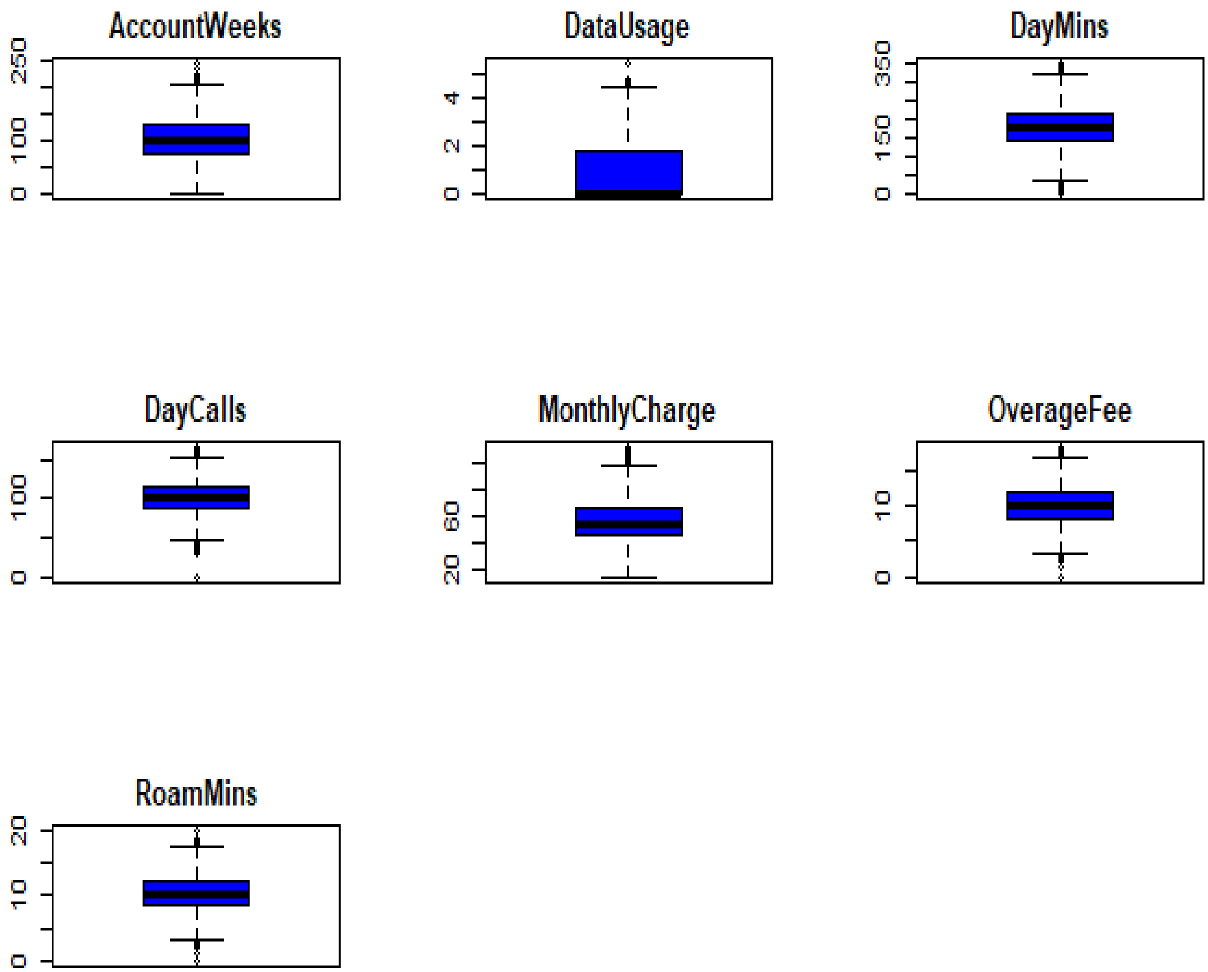
From the graph we can see that only 28% have taken the data plan, while the remaining 72% haven't taken any plan. Customers having the value 1 is 922 while having 0 is 2411.

Plot for Customer Service Calls



From the graph we can see that, the highest no of customers of around 1181 have done only 1 call to the customer service. While around 35 have done more than 6 calls, with 9 number of calls by any customer being the highest.

Boxplot



All the continuous variables have outliers. Account Weeks, Data Usage, Monthly Charge has outliers on the max side. While Day Mins, Day Calls, Overage Fee and Roam Mins has outliers on both the sides. Flooring and capping have been applied on all the continuous variables. We are using 1 and 99 percentiles for outlier treatment.

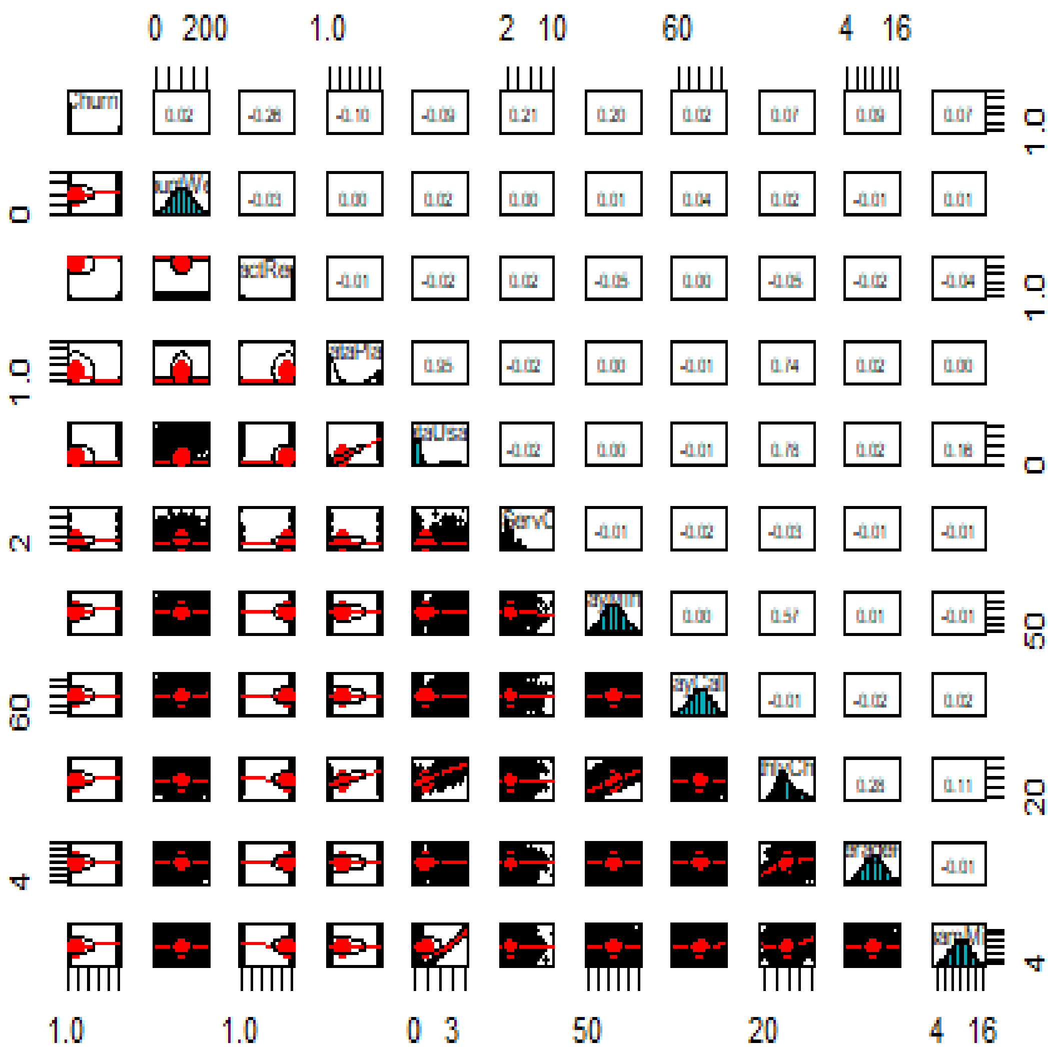
```
summary(cellphone)

##  Churn     AccountWeeks   ContractRenewal DataPlan    DataUsage
##  0:2850   Min.   :  1.0   0: 323          0:2411   Min.   :0.0000
```

```
##  1: 483   1st Qu.: 74.0   1:3010          1: 922   1st Qu.:0.0000
##           Median :101.0                            Median :0.0000
##           Mean   :100.9                            Mean   :0.8136
##           3rd Qu.:127.0                            3rd Qu.:1.7800
##           Max.   :195.0                            Max.   :4.1000
##
##  CustServCalls     DayMins          DayCalls     MonthlyCharge
##  1       :1181   Min.   : 51.83   Min.   : 54.0   Min.   :14.00
##  2       : 759   1st Qu.:143.70   1st Qu.: 87.0   1st Qu.:45.00
##  0       : 697   Median :179.40   Median :101.0   Median :53.50
##  3       : 429   Mean   :179.79   Mean   :100.5   Mean   :56.25
##  4       : 166   3rd Qu.:216.40   3rd Qu.:114.0   3rd Qu.:66.20
##  5       :  66   Max.   :305.17   Max.   :146.0   Max.   :98.28
##  (Other):  35
##    OverageFee       RoamMins
##  Min.   : 3.98   Min.   : 3.332
##  1st Qu.: 8.33   1st Qu.: 8.500
##  Median :10.07   Median :10.300
##  Mean   :10.05   Mean   :10.251
##  3rd Qu.:11.77   3rd Qu.:12.100
##  Max.   :15.95   Max.   :16.668
##
```

Post the outlier treatment on the continuous variables.

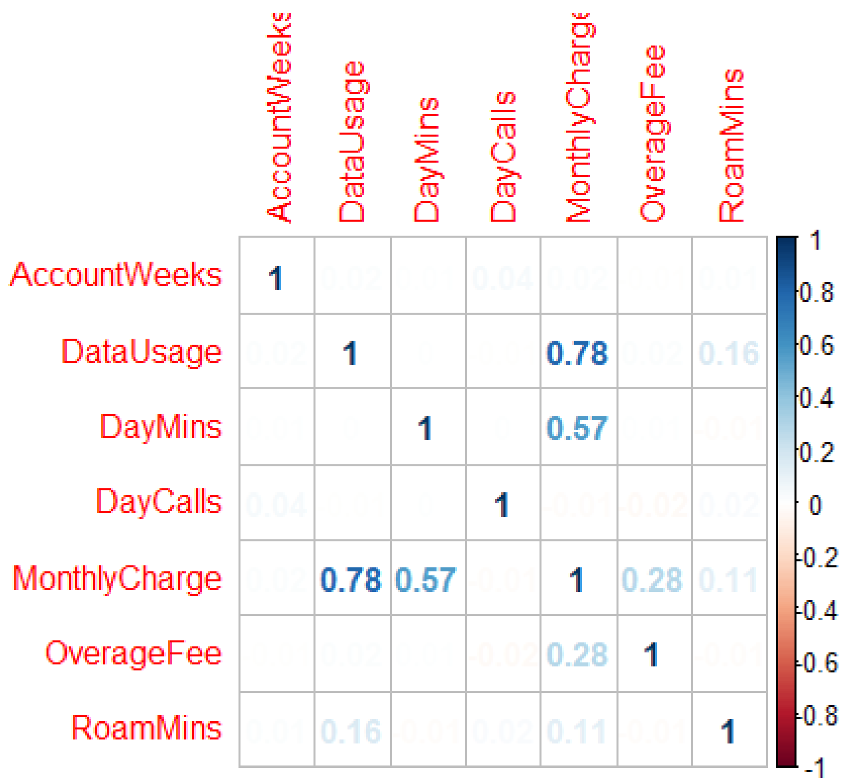
- **Account Weeks** - the max value has reduced from 243 to 195 and mean has been reduced to 100.9.
- **Data Usages** - the max value has reduced from 5.4 to 4.1 and mean has been reduced to 0.81369.
- **Day Mins** - the max value has reduced from 350.8 to 305.17 and min has been increased from 0 to 51.83.
- **Day Calls** - the max value has reduced from 165 to 146 and min has been increased from 0 to 54.
- **Monthly Charge** - the max value has reduced from 111.30 to 98.28. Average Fee - the max value has reduced from 18.19 to 15.95.
- **Roam Mins** - the max value has reduced from 20 to 16.6 and min has been increased from 0 to 3.33.

Correlation graph based on Pearson for Bivariate Analysis



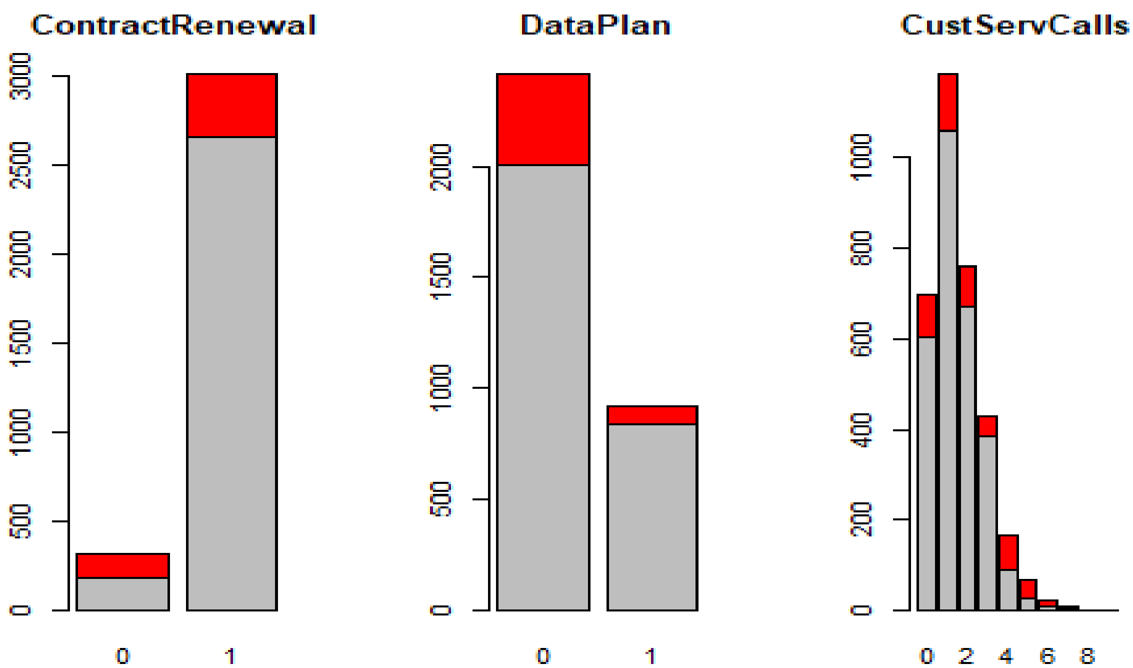

```
##correlation between continuous variables (Bivariate Analysis)

##               AccountWeeks DataUsage DayMins DayCalls MonthlyCharge
## AccountWeeks         1.00      0.02    0.01     0.04          0.02
## DataUsage            0.02      1.00    0.00    -0.01          0.78
## DayMins              0.01      0.00    1.00     0.00          0.57
## DayCalls             0.04     -0.01    0.00     1.00         -0.01
## MonthlyCharge        0.02      0.78    0.57    -0.01          1.00
## OverageFee          -0.01      0.02    0.01    -0.02          0.28
## RoamMins             0.01      0.16   -0.01     0.02          0.11
```

```
##                  OverageFee RoamMins
## AccountWeeks          -0.01     0.01
## DataUsage              0.02     0.16
## DayMins                0.01    -0.01
## DayCalls              -0.02     0.02
## MonthlyCharge          0.28     0.11
## OverageFee             1.00    -0.01
## RoamMins              -0.01     1.00
```



The correlation graph shows the values in blue having the highest correlation, while in red having the lowest correlation. From the graph as well as the correlation matrix, we can see that Data Usage and Day Mins have high correlation with Monthly Charge. Overage Fee also has a slight correlation with Monthly Charges.

```
# contingency table of dicotomous variables with target variable
# for 3 categorical variables draw the bar plot w.r.t to target variable
```



The grey color is represented for the churn variable.

- **Contract Renewal** - Churn is not majorly dependent on this variable as, high rate of churn means low contract renewal.
- **Data Plan** - the people without data plan have a churn rate almost as much as the people without data plan.
- **Customer Service Calls** - The people who have called customer service less than 3 times, seems to have a higher churn rate.

Chi Square test for Categorical variables (Bivariate Analysis)

Chi-Square test is a statistical method which is used to determine if two categorical variables have a significant correlation between them. Here we are taking a significance level of 0.05 and comparing the p-value to accept or reject the Null Hypothesis.

Categorical Variables	P-Value	Analysis
Churn and ContractRenewal	`p-value < 2.2e-16`	Null hypothesis is rejected.
Churn and DataPlan	`p-value = 5.151e-09`	Failed to reject the Null hypothesis.
Churn and CustServCalls	`p-value < 2.2e-16`	Null hypothesis is rejected.
ContractRenewal and DataPlan	`p-value = 0.785`	Failed to reject the Null hypothesis.
ContractRenewal and CustServCalls	`p-value = 0.08388`	Failed to reject the Null hypothesis.
DataPlan and CustServCalls	`p-value = 0.3461`	Failed to reject the Null hypothesis.

Model Building

The numerical variables have been scaled. The categorical variables have been smoothened to create dummy variables. The numerical and categorical variables have been combined to create a new dataset.

```
#Scaling the numerical variables

sampleDS <- cellphone[,c("AccountWeeks", "DataUsage", "DayMins","DayCalls","M
onthlyCharge","OverageFee","RoamMins")]
sampleDS <- data.frame(scale(sampleDS))


#Creating dummy variables for the categorical varianles

cat.sampleDS2 = cbind(cat.sampleDS,cellphone$Churn)
```

```r
dummy<- data.frame(sapply(cat.sampleDS2,function(x) data.frame(model.matrix(~
x-1,data =cat.sampleDS2))[,-1]))


#Integrate categorical and numerical dataset

full.data = cbind(sampleDS, dummy)
names(full.data)

##  [1] "AccountWeeks"     "DataUsage"        "DayMins"
##  [4] "DayCalls"         "MonthlyCharge"    "OverageFee"
##  [7] "RoamMins"         "ContractRenewal"  "DataPlan"
## [10] "CustServCalls.x1" "CustServCalls.x2" "CustServCalls.x3"
## [13] "CustServCalls.x4" "CustServCalls.x5" "CustServCalls.x6"
## [16] "CustServCalls.x7" "CustServCalls.x8" "CustServCalls.x9"
## [19] "Churn"

head(full.data)

##   AccountWeeks  DataUsage    DayMins   DayCalls MonthlyCharge  OverageFee
## 1    0.6868924  1.4914933  1.5934878  0.4848728    2.01287140 -0.07326067
## 2    0.1542763  2.2821644 -0.3398239  1.1471803    1.58264577 -0.10939069
## 3    0.9151565 -0.6433188  1.1881461  0.6886597   -0.26117836 -1.60276514
## 4   -0.4290652 -0.6433188  2.2341891 -1.5020495    0.04612566 -2.43793078
## 5   -0.6573293 -0.6433188 -0.2445592  0.6377130   -0.93724721 -1.05680029
## 6    0.4332657 -0.6433188  0.8145593 -0.1264879    0.04612566  0.39241524
##      RoamMins ContractRenewal DataPlan CustServCalls.x1 CustServCalls.x2
## 1 -0.09292618               1        1                1                0
## 2  1.27720165               1        1                1                0
## 3  0.72174442               1        0                0                0
## 4 -1.35196256               0        0                0                1
## 5 -0.05589570               0        0                0                0
## 6 -1.46305400               0        0                0                0
##   CustServCalls.x3 CustServCalls.x4 CustServCalls.x5 CustServCalls.x6
## 1                0                0                0                0
## 2                0                0                0                0
## 3                0                0                0                0
## 4                0                0                0                0
## 5                1                0                0                0
## 6                0                0                0                0
##   CustServCalls.x7 CustServCalls.x8 CustServCalls.x9 Churn
## 1                0                0                0     0
## 2                0                0                0     0
## 3                0                0                0     0
## 4                0                0                0     0
## 5                0                0                0     0
## 6                0                0                0     0

#Split data into train and test dataset
set.seed(1000)
```

```
ds = sample.split(full.data$Churn, SplitRatio = 0.70)
train = subset(full.data, ds == T)
test = subset(full.data, ds == F)

#check split consistency
sum(as.integer(as.character(train$Churn)))/nrow(train)

## [1] 0.1448778

sum(as.integer(as.character(test$Churn)))/nrow(test)

## [1] 0.145

sum(as.integer(as.character(full.data$Churn)))/nrow(full.data)

## [1] 0.1449145
```

This is an imbalanced dataset. The data has been split equally, with a churn rate of 14.5% across test and train dataset.

Logistic Regression

Logistic Regression is a predictive analysis model. It 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.

As multicollinearity exists, the model needs to be treated using VIF. The model was tested using VIF and Blorr package. Since both are giving the same prediction, I'm going ahead with the Blorr package.

```
#build the logistic model
LR_Train_model = glm(Churn ~ ., data = train, family = binomial)
summary(LR_Train_model)

##
## Call:
## glm(formula = Churn ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4692  -0.4666  -0.3206  -0.1873   3.2490
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.23936    0.26811  -0.893  0.37199
## AccountWeeks         0.06105    0.07019   0.870  0.38439
## DataUsage           -3.38475    1.13864  -2.973  0.00295 **
## DayMins             -1.93600    0.80891  -2.393  0.01670 *
```

```
## DayCalls           0.02041    0.06771   0.301  0.76312    
## MonthlyCharge      4.64165    1.43554   3.233  0.00122 ** 
## OverageFee        -0.82311    0.38371  -2.145  0.03194 *  
## RoamMins           0.34134    0.07591   4.497 6.90e-06 ***
## ContractRenewal   -2.08941    0.17359 -12.036  < 2e-16 ***
## DataPlan          -1.64841    0.71758  -2.297  0.02161 *  
## CustServCalls.x1  -0.05681    0.19884  -0.286  0.77511    
## CustServCalls.x2   0.15861    0.21640   0.733  0.46359    
## CustServCalls.x3  -0.23206    0.26091  -0.889  0.37377    
## CustServCalls.x4   2.18778    0.26809   8.161 3.34e-16 ***
## CustServCalls.x5   3.13322    0.36389   8.610  < 2e-16 ***
## CustServCalls.x6   4.56123    0.65872   6.924 4.38e-12 ***
## CustServCalls.x7   3.76144    0.89187   4.217 2.47e-05 ***
## CustServCalls.x8  16.72355  535.41121   0.031  0.97508    
## CustServCalls.x9  13.88911  535.41131   0.026  0.97930    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1930.4  on 2332  degrees of freedom
## Residual deviance: 1415.6  on 2314  degrees of freedom
## AIC: 1453.6
## 
## Number of Fisher Scoring iterations: 12

#We are considering the VIF values greater than 2.5 as having high multicolli
nearlity.
vif(LR_Train_model)

##      AccountWeeks        DataUsage          DayMins         DayCalls 
##          1.015714       219.250171       129.722324         1.014619 
##     MonthlyCharge       OverageFee         RoamMins  ContractRenewal 
##        372.516185        30.018910         1.194221         1.076886 
##          DataPlan CustServCalls.x1 CustServCalls.x2 CustServCalls.x3 
##         15.536773         1.773958         1.637500         1.388706 
## CustServCalls.x4 CustServCalls.x5 CustServCalls.x6 CustServCalls.x7 
##          1.425129         1.248335         1.120221         1.047203 
## CustServCalls.x8 CustServCalls.x9 
##          1.000000         1.000000
```

We can use variance inflation factor (VIF) to get rid of redundant predictors or the variables that have high multicollinearity between them. Multicollinearity exists when two or more predictor variables are highly related to each other and then it becomes difficult to understand the impact of an independent variable on the dependent variable.

The Variance Inflation Factor (VIF) is used to measure the multicollinearity between predictor variables in a model. A predictor having a VIF of 2.5 or less is generally considered safe and it can be assumed that it is not correlated with other predictor variables. Higher the VIF, greater is the correlation of the predictor variable w.r.t other predictor variables. However, Predictors with high VIF may have high p-value (or highly significant), hence, we need to see the significance of the Predictor variable before removing it from our model.

From the test we can see that DataUsage, DayMins, MonthlyCharge, OverageFee and DataPlan have high VIF values.

Likelihood – Ratio Test

The **Likelihood-Ratio test** (sometimes called the likelihood-ratio chi-squared test) is a hypothesis test that helps you choose the "best" model between two nested models. Basically, the test compares the fit of two models. The null hypothesis is that the smaller model is the "best" model; It is rejected when the test statistic is large. In other words, if the null hypothesis is rejected, then the larger model is a significant improvement over the smaller one.

```
#Log likihood test: To ensure if logit model is valid or not
lrtest(LR_Train_model)

## Likelihood ratio test
##
## Model 1: Churn ~ AccountWeeks + DataUsage + DayMins + DayCalls + MonthlyCh
arge +
##     OverageFee + RoamMins + ContractRenewal + DataPlan + CustServCalls.x1
+
##     CustServCalls.x2 + CustServCalls.x3 + CustServCalls.x4 +
##     CustServCalls.x5 + CustServCalls.x6 + CustServCalls.x7 +
##     CustServCalls.x8 + CustServCalls.x9
## Model 2: Churn ~ 1
##   #Df  LogLik  Df  Chisq Pr(>Chisq)
## 1  19 -707.81
## 2   1 -965.21 -18 514.79  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

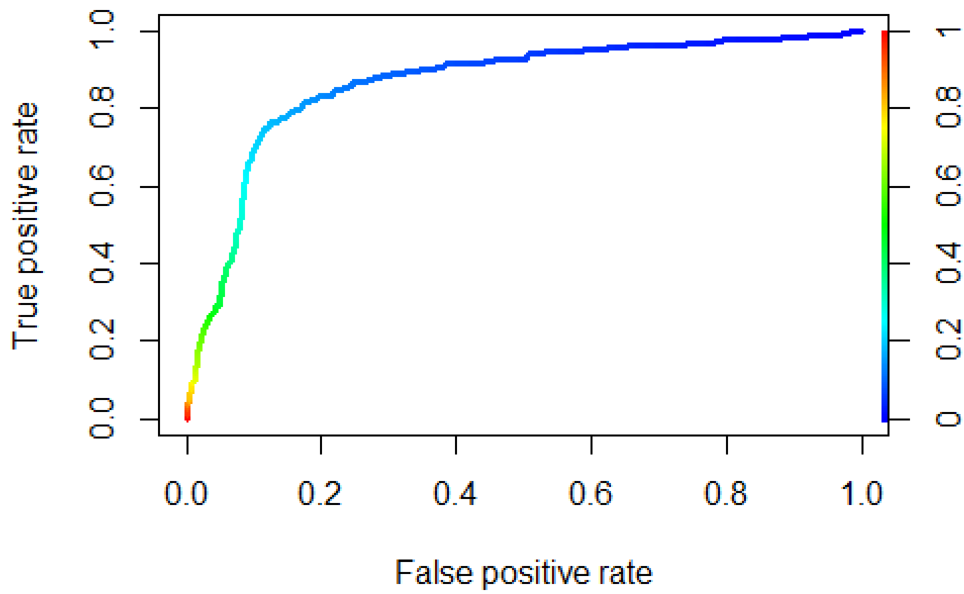
The full model has a score of -707.81, while the null model has -965.21. We need to choose the model with a lower score. Therefore, the null hypothesis is not rejected.

```
# To get the logit R2 of goodness
pR2(LR_Train_model)

##          llh      llhNull           G2     McFadden         r2ML
## -707.8123532 -965.2094900  514.7942734    0.2666749    0.1980088
##         r2CU
##    0.3518072
```

```
# Trust only McFadden since its conservative
#if my McFadden > is between 0 to 10 - Goodness of fit is weak
#if my McFadden > is between 10 to 20 - Goodness of fit is fare
#if my McFadden > is between 20 to 30 - Goodness of fit is Moderately is robu
st
#if my McFadden > is between 30 and above - Goodness of fit is reasonably rob
ust model
#Typical in non-linear model R2 will be less as against linear regression
```

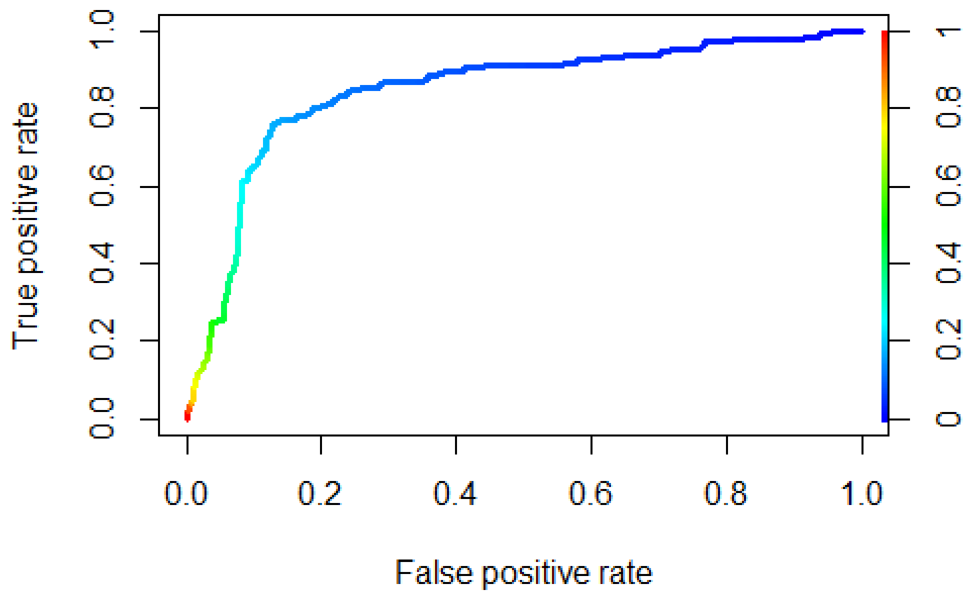
Since our McFadden value is around 26.67%, we can consider our model to be moderately robust.

```
odds = exp(coef(LR_Train_model))

#for identifying the relative importance of variables we have to use ODDS ins
tead of PROB
prob=odds/(1+odds)
relativeImportance=(odds[-1]/sum(odds[-1]))*100
relativeImportance[order(relativeImportance)]

##        DataUsage  ContractRenewal          DayMins         DataPlan
##     1.746921e-07     6.380211e-07     7.438066e-07     9.916494e-07
##       OverageFee CustServCalls.x3 CustServCalls.x1         DayCalls
##     2.263500e-06     4.087644e-06     4.870596e-06     5.261575e-06
##     AccountWeeks CustServCalls.x2         RoamMins CustServCalls.x4
##     5.479840e-06     6.041421e-06     7.252670e-06     4.596144e-05
## CustServCalls.x5 CustServCalls.x7 CustServCalls.x6    MonthlyCharge
##     1.183025e-04     2.217309e-04     4.933684e-04     5.346849e-04
## CustServCalls.x9 CustServCalls.x8
##     5.549054e+00     9.444949e+01

# Performance on TRAIN dataset
predTrain = predict(LR_Train_model, newdata = train, type="response")
ptrain = table(train$Churn, predTrain>0.3)
sum(diag(ptrain)) / nrow(train)

## [1] 0.8602658

ROCRpred = prediction(predTrain, train$Churn)
as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.8650413

perf = performance(ROCRpred, "tpr","fpr")
plot(perf,col="black",lty=2, lwd=2)

plot(perf,lwd=3,colorize = TRUE)
```



ROC stands for **Receiver Operating Characteristic**. ... The **ROC curve** does this by plotting sensitivity, the probability of predicting a real positive will be a positive, against 1-specificity, the probability of predicting a real negative will be a positive. The ROC for the train data set is 0.865.

```
# Performance on TEST dataset
predTest = predict(LR_Train_model, newdata = test, type="response")
ptr=table(test$Churn, predTest>0.3)

sum(diag(ptr)) / nrow(test)

## [1] 0.856

ROCRpred = prediction(predTest, test$Churn)
as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.846703

perf = performance(ROCRpred, "tpr","fpr")
plot(perf,col="black",lty=2, lwd=2)

plot(perf,lwd=3,colorize = TRUE)
```



The ROC for the test data set is 0.847.

The variables are ordered based on their importance in the model. A prediction was done on the train model, which gave an accuracy of 86%. Similarly, the test model gave an accuracy of 85.6%.

```
### AREA UNDER THE CURVE
logit_auc = performance(ROCRpred, "auc")
as.numeric(logit_auc@y.values) ##AUC Value

## [1] 0.846703

### KS STATISTIC
logit_ks = max(perf@y.values[[1]]-perf@x.values[[1]])
logit_ks

## [1] 0.6344021

# gains table

gains.cross <- gains(actual=test$Churn , predicted=predTest, groups=10)
print(gains.cross)

## Depth                            Cume    Cume Pct                     Mean
##  of             Cume      Mean    Mean    of Total     Lift    Cume    Model
## File      N       N       Resp    Resp        Resp    Index    Lift    Score
## -------------------------------------------------------------------------
```

```
##   10   100   100      0.49     0.49     33.8%    338    338      0.63
##   20   100   200      0.51     0.50     69.0%    352    345      0.27
##   30   100   300      0.18     0.39     81.4%    124    271      0.15
##   40   100   400      0.08     0.32     86.9%     55    217      0.11
##   50   100   500      0.05     0.26     90.3%     34    181      0.09
##   60   100   600      0.01     0.22     91.0%      7    152      0.07
##   70   100   700      0.04     0.19     93.8%     28    134      0.05
##   80   100   800      0.05     0.18     97.2%     34    122      0.04
##   90   100   900      0.01     0.16     97.9%      7    109      0.03
##  100   100  1000      0.03     0.14    100.0%     21    100      0.01


############### MODEL PERFORMANCE ###################

blr_step_aic_forward(LR_Train_model, details = FALSE)

## Forward Selection Method
## ------------------------
##
## Candidate Terms:
##
## 1 . AccountWeeks
## 2 . DataUsage
## 3 . DayMins
## 4 . DayCalls
## 5 . MonthlyCharge
## 6 . OverageFee
## 7 . RoamMins
## 8 . ContractRenewal
## 9 . DataPlan
## 10 . CustServCalls.x1
## 11 . CustServCalls.x2
## 12 . CustServCalls.x3
## 13 . CustServCalls.x4
## 14 . CustServCalls.x5
## 15 . CustServCalls.x6
## 16 . CustServCalls.x7
## 17 . CustServCalls.x8
## 18 . CustServCalls.x9
##
##
## Variables Entered:
##
## - ContractRenewal
## - DayMins
## - CustServCalls.x4
## - CustServCalls.x5
## - CustServCalls.x6
## - DataPlan
```

```
## - MonthlyCharge
## - CustServCalls.x7
## - RoamMins
## - DataUsage
## - CustServCalls.x8
## - OverageFee
##
## No more variables to be added.

##
##                      Selection Summary
## ------------------------------------------------------------
## Step    Variable              AIC         BIC        Deviance
## ------------------------------------------------------------
## 1       ContractRenewal     1811.954    1823.464    1807.954
## 2       DayMins             1739.364    1739.364    1739.364
## 3       CustServCalls.x4    1675.126    1675.126    1675.126
## 4       CustServCalls.x5    1606.063    1606.063    1606.063
## 5       CustServCalls.x6    1566.224    1566.224    1566.224
## 6       DataPlan            1524.995    1524.995    1524.995
## 7       MonthlyCharge       1487.875    1487.875    1487.875
## 8       CustServCalls.x7    1473.562    1473.562    1473.562
## 9       RoamMins            1460.519    1460.519    1460.519
## 10      DataUsage           1453.936    1453.936    1453.936
## 11      CustServCalls.x8    1449.587    1449.587    1449.587
## 12      OverageFee          1446.819    1446.819    1446.819
## ------------------------------------------------------------

blr_step_aic_backward(LR_Train_model,details = FALSE)

## Backward Elimination Method
## ---------------------------
##
## Candidate Terms:
##
## 1 . AccountWeeks
## 2 . DataUsage
## 3 . DayMins
## 4 . DayCalls
## 5 . MonthlyCharge
## 6 . OverageFee
## 7 . RoamMins
## 8 . ContractRenewal
## 9 . DataPlan
## 10 . CustServCalls.x1
## 11 . CustServCalls.x2
## 12 . CustServCalls.x3
## 13 . CustServCalls.x4
## 14 . CustServCalls.x5
## 15 . CustServCalls.x6
```

```
## 16 . CustServCalls.x7
## 17 . CustServCalls.x8
## 18 . CustServCalls.x9
##
##
## Variables Removed:
##
## - CustServCalls.x1
## - DayCalls
## - AccountWeeks
## - CustServCalls.x3
## - CustServCalls.x2
## - CustServCalls.x9

##
##
##              Backward Elimination Summary
## ----------------------------------------------------
## Variable              AIC         BIC       Deviance
## ----------------------------------------------------
## Full Model          1453.625    1562.968    1415.625
## CustServCalls.x1    1451.706    1555.295    1415.706
## DayCalls            1449.797    1547.630    1415.797
## AccountWeeks        1448.555    1540.633    1416.555
## CustServCalls.x3    1447.332    1533.655    1417.332
## CustServCalls.x2    1447.006    1527.575    1419.006
## CustServCalls.x9    1446.819    1521.633    1420.819
## ----------------------------------------------------
```

```r
blr_step_aic_both(LR_Train_model, details = FALSE)
```

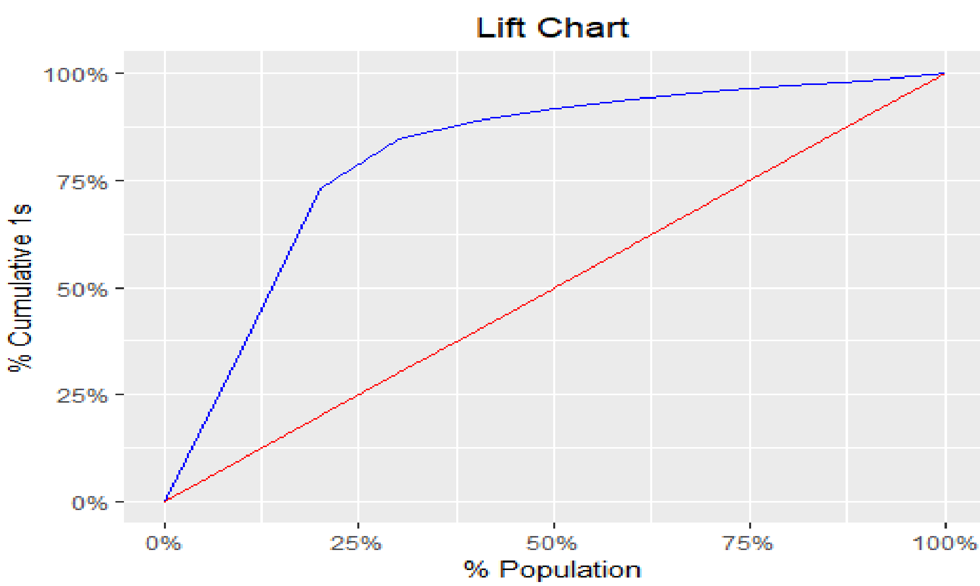
```
## Stepwise Selection Method
## -------------------------
##
## Candidate Terms:
##
## 1 . AccountWeeks
## 2 . DataUsage
## 3 . DayMins
## 4 . DayCalls
## 5 . MonthlyCharge
## 6 . OverageFee
## 7 . RoamMins
## 8 . ContractRenewal
## 9 . DataPlan
## 10 . CustServCalls.x1
## 11 . CustServCalls.x2
## 12 . CustServCalls.x3
## 13 . CustServCalls.x4
## 14 . CustServCalls.x5
```

```
## 15 . CustServCalls.x6
## 16 . CustServCalls.x7
## 17 . CustServCalls.x8
## 18 . CustServCalls.x9
##
##
## Variables Entered/Removed:
##
## - ContractRenewal added
## - DayMins added
## - CustServCalls.x4 added
## - CustServCalls.x5 added
## - CustServCalls.x6 added
## - DataPlan added
## - MonthlyCharge added
## - DayMins removed
## - CustServCalls.x7 added
## - RoamMins added
## - DataUsage added
## - CustServCalls.x8 added
##
## No more variables to be added or removed.

##
##
##                          Stepwise Summary
## ----------------------------------------------------------------
## Variable             Method       AIC         BIC       Deviance
## ----------------------------------------------------------------
## ContractRenewal     addition    1811.954    1823.464    1807.954
## DayMins             addition    1739.364    1756.629    1733.364
## CustServCalls.x4    addition    1675.126    1698.146    1667.126
## CustServCalls.x5    addition    1606.063    1634.838    1596.063
## CustServCalls.x6    addition    1566.224    1600.754    1554.224
## DataPlan            addition    1524.995    1565.279    1510.995
## MonthlyCharge       addition    1487.875    1533.915    1471.875
## DayMins             removal     1486.515    1526.799    1472.515
## CustServCalls.x7    addition    1472.012    1518.051    1456.012
## RoamMins            addition    1458.526    1510.320    1440.526
## DataUsage           addition    1454.061    1511.610    1434.061
## CustServCalls.x8    addition    1449.804    1513.108    1427.804
## ----------------------------------------------------------------

#Using the backward model
final.model = glm(Churn ~ DayMins + ContractRenewal + CustServCalls.x4 + Cust
ServCalls.x5 + OverageFee + CustServCalls.x6 + DataPlan + CustServCalls.x8 +
CustServCalls.x9 + CustServCalls.x7 + RoamMins + MonthlyCharge + DataUsage, d
ata = train, family = binomial)
```

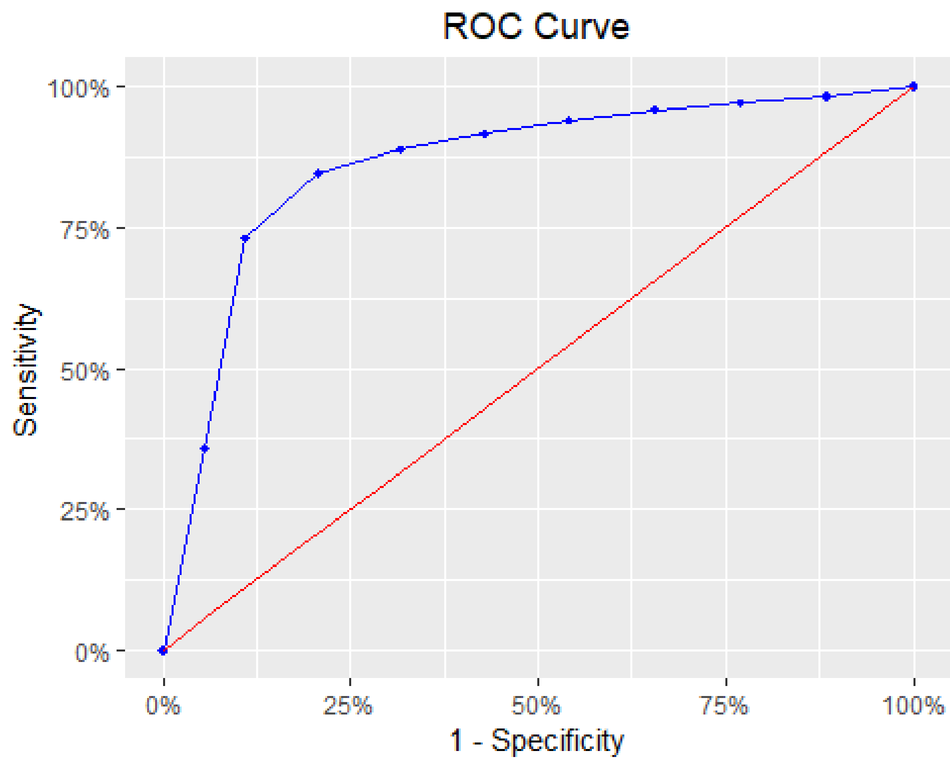
```r
summary(final.model)

##
## Call:
## glm(formula = Churn ~ DayMins + ContractRenewal + CustServCalls.x4 +
##     CustServCalls.x5 + OverageFee + CustServCalls.x6 + DataPlan +
##     CustServCalls.x8 + CustServCalls.x9 + CustServCalls.x7 +
##     RoamMins + MonthlyCharge + DataUsage, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4684  -0.4692  -0.3211  -0.1873   3.2128
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.2446     0.2354  -1.039  0.29867
## DayMins            -1.9539     0.8055  -2.426  0.01527 *
## ContractRenewal    -2.0917     0.1729 -12.101  < 2e-16 ***
## CustServCalls.x4    2.2048     0.2295   9.605  < 2e-16 ***
## CustServCalls.x5    3.1478     0.3353   9.388  < 2e-16 ***
## OverageFee         -0.8335     0.3821  -2.181  0.02915 *
## CustServCalls.x6    4.5259     0.6392   7.081 1.43e-12 ***
## DataPlan           -1.6680     0.7176  -2.325  0.02009 *
## CustServCalls.x8   16.7681   535.4112   0.031  0.97502
## CustServCalls.x9   13.9911   535.4113   0.026  0.97915
## CustServCalls.x7    3.7830     0.8781   4.308 1.64e-05 ***
## RoamMins            0.3402     0.0755   4.506 6.61e-06 ***
## MonthlyCharge       4.6747     1.4293   3.271  0.00107 **
## DataUsage          -3.3979     1.1340  -2.996  0.00273 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1930.4  on 2332  degrees of freedom
## Residual deviance: 1419.0  on 2319  degrees of freedom
## AIC: 1447
##
## Number of Fisher Scoring iterations: 12

predTest = predict(final.model, newdata= test, type="response")
npt = table(test$Churn, predTest>0.3)
sum(diag(npt))/nrow(test)

## [1] 0.858

blr_model_fit_stats(final.model)
```

```
##                                Model Fit Statistics
## ---------------------------------------------------------------------------
-------
## Warning: `prepend()` is deprecated as of rlang 0.4.0.
##
## Vector tools are now out of scope for rlang to make it a more
## focused package.
## This warning is displayed once per session.
## Log-Lik Intercept Only:      -965.209    Log-Lik Full Model:              -
709.503
## Deviance(2319):             1419.006    LR(13):
511.413
##                                          Prob > LR:
0.000
## MCFadden's R2                   0.265    McFadden's Adj R2:
0.250
## ML (Cox-Snell) R2:              0.197    Cragg-Uhler(Nagelkerke) R2:
0.350
## McKelvey & Zavoina's R2:        0.414    Efron's R2:
0.247
## Count R2:                       0.859    Adj Count R2:
0.027
## BIC:                        1527.575    AIC:                             1
447.006
## ---------------------------------------------------------------------------
-------
blr_regress(final.model)
## - Creating model overview.
## - Creating response profile.
## - Extracting maximum likelihood estimates.
## - Estimating concordant and discordant pairs.
##                               Model Overview
## -------------------------------------------------------------------------
## Data Set    Resp Var    Obs.    Df. Model    Df. Residual    Convergence
## -------------------------------------------------------------------------
##   data       Churn      2333      2332           2319            TRUE
## -------------------------------------------------------------------------
##
##                          Response Summary
## ----------------------------------------------------------
## Outcome         Frequency        Outcome         Frequency
## ----------------------------------------------------------
##    0              1995              1               338
## ----------------------------------------------------------
##
##                          Maximum Likelihood Estimates
```

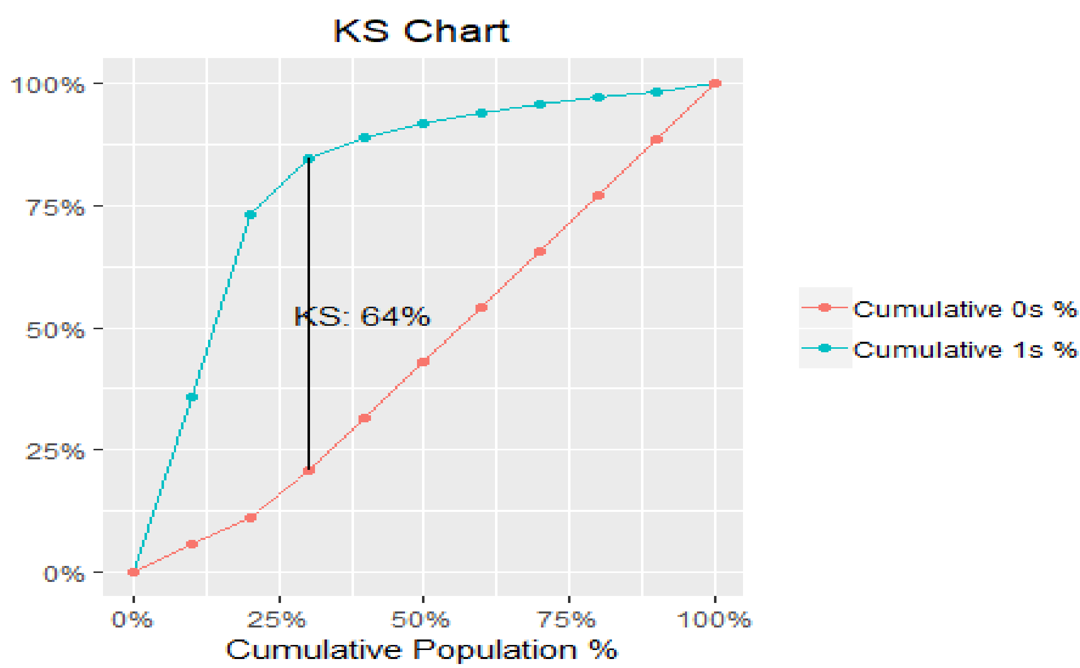
```
## -------------------------------------------------------------------------
##    Parameter          DF     Estimate     Std. Error     z value      Pr(>|z|)
## -------------------------------------------------------------------------
##   (Intercept)         1      -0.2446         0.2354     -1.0393        0.2987
##      DayMins          1      -1.9539         0.8055     -2.4258        0.0153
## ContractRenewal       1      -2.0917         0.1729    -12.1006        0.0000
## CustServCalls.x4      1       2.2048         0.2295      9.6051        0.0000
## CustServCalls.x5      1       3.1478         0.3353      9.3883        0.0000
##    OverageFee         1      -0.8335         0.3821     -2.1814        0.0292
## CustServCalls.x6      1       4.5259         0.6392      7.0807        0.0000
##      DataPlan         1      -1.6680         0.7175     -2.3246        0.0201
## CustServCalls.x8      1      16.7681       535.4112      0.0313        0.9750
## CustServCalls.x9      1      13.9911       535.4113      0.0261        0.9792
## CustServCalls.x7      1       3.7830         0.8781      4.3084        0.0000
##      RoamMins         1       0.3402         0.0755      4.5058        0.0000
##   MonthlyCharge       1       4.6747         1.4293      3.2706        0.0011
##    DataUsage          1      -3.3979         1.1340     -2.9963        0.0027
## -------------------------------------------------------------------------
##
##  Association of Predicted Probabilities and Observed Responses
## ---------------------------------------------------------------
## % Concordant          0.8644          Somers' D        0.7288
## % Discordant          0.1356          Gamma            0.7288
## % Tied                0.0000          Tau-a            0.1807
## Pairs                 674310          c                0.8644
## ---------------------------------------------------------------

################# Other package for model performance ############

k = blr_gains_table(final.model,train)
plot(k)
```



```
blr_roc_curve(k)
```

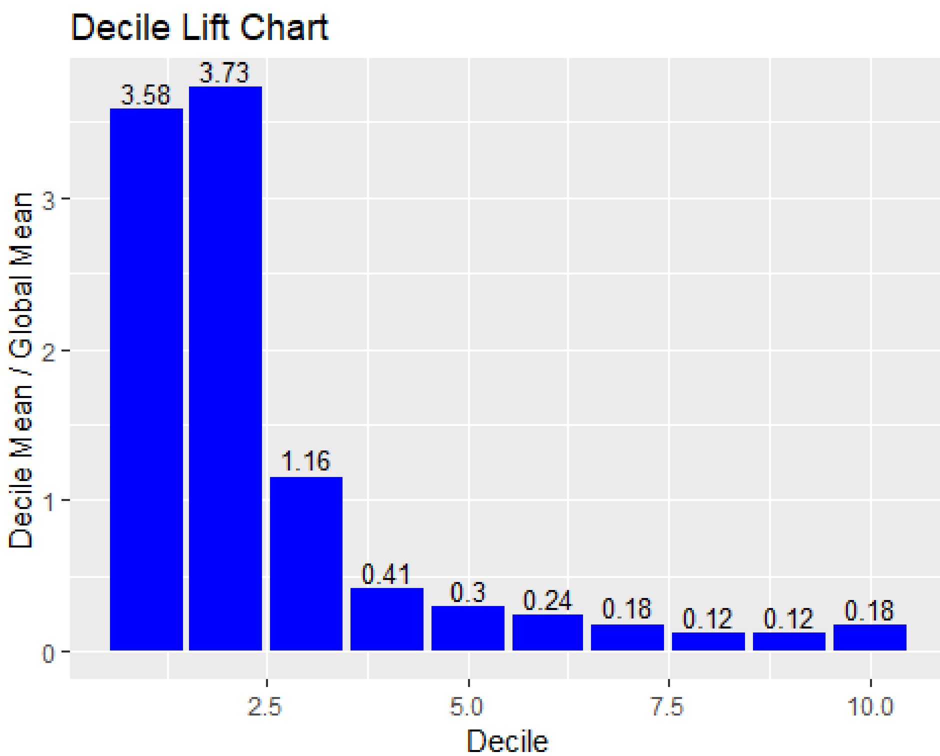



```
blr_ks_chart(k, title = "KS Chart",
          yaxis_title = " ",xaxis_title = "Cumulative Population %",
          ks_line_color = "black")
```



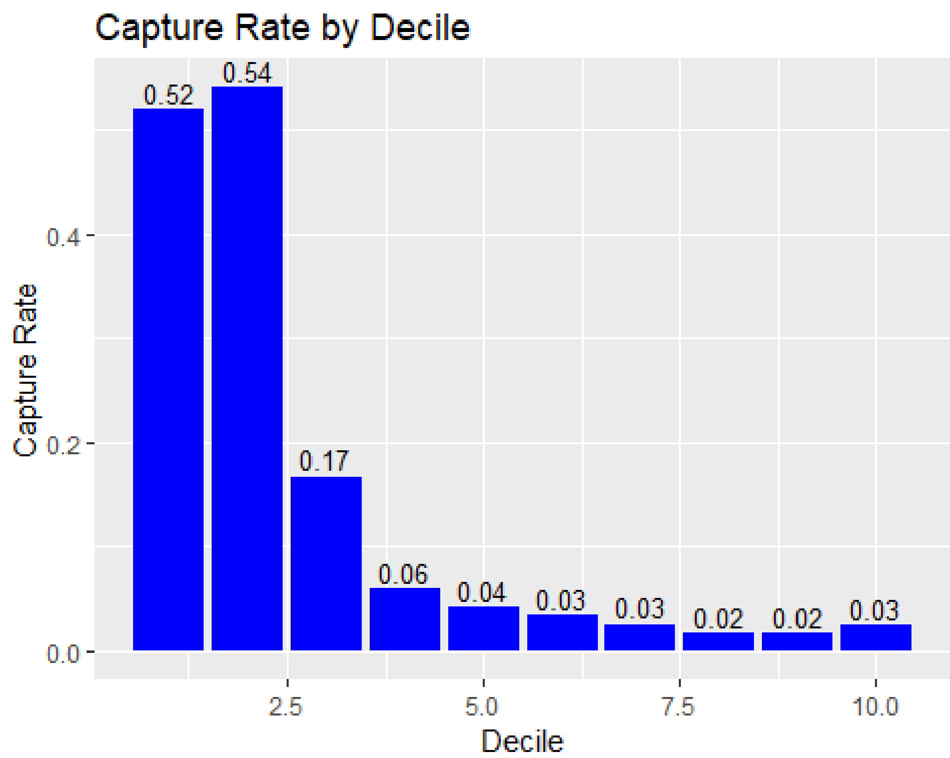
The lift, ROC and KS charts have been plotted to give a better understand. The KS is about 64%.

```
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)
```



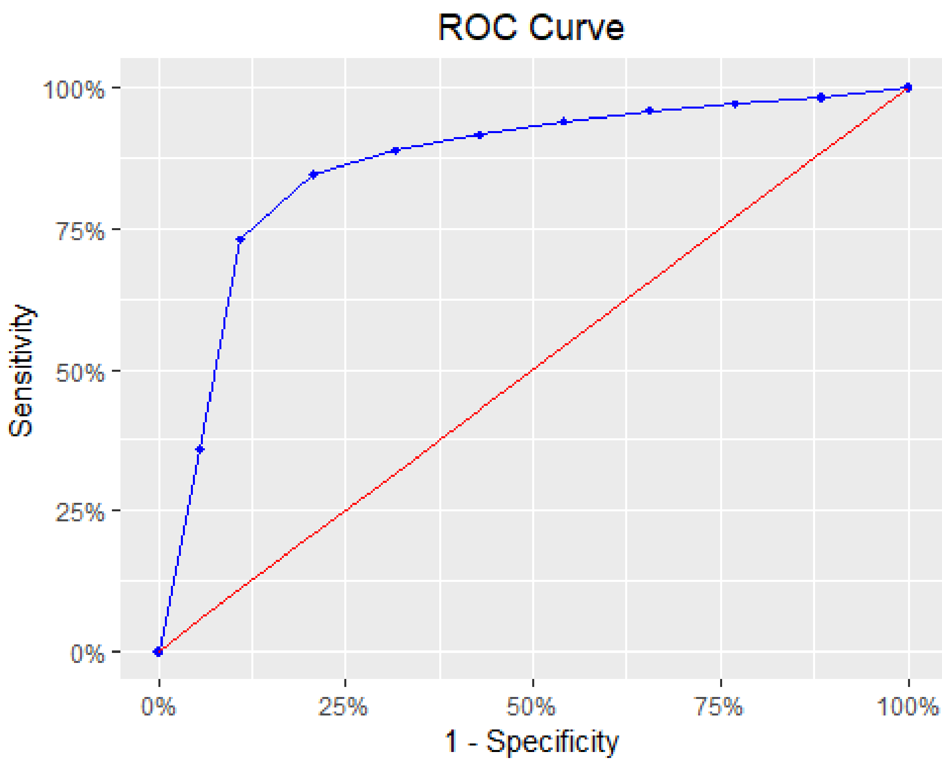
Most of the data falls in 2.5 decile for the decile lift.

```
blr_decile_capture_rate(k, xaxis_title = "Decile",
                        yaxis_title = "Capture Rate",
                        title = "Capture Rate by Decile",
                        bar_color = "blue", text_size = 3.5,
                        text_vjust =-0.3)
```



Most of the data falls in 2.5 decile for the decile for capture rate.

```
blr_gini_index(final.model, data = test)

## [1] 0.5677006

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_rsq_mcfadden(final.model)

## [1] 0.2649232

blr_rsq_mcfadden_adj(final.model)

## [1] 0.2504186
```

The Logistic Regression Model has given a GINI Index of 0.5677. The KS is 64%.

KNN Model

K nearest neighbors is a simple **algorithm** that stores all available cases and classifies new cases based on a similarity measure. Normalizing the numeric data is important since the scale used for the values for each variable might be different. We will be using the same test and train dataset as logistic regression.

```
set.seed(1000)
#Removing he dependent variable from the train and test dataset
train_knn = train[-19]
test_knn = test[-19]

#Check the dimensions of the train and test dataset
dim(train_knn)
```

```
## [1] 2333   18

dim(test_knn)

## [1] 1000   18

#Storing the target variable for training and test dataset
knn_train_label = train$Churn
knn_test_label = test$Churn

#KNN Model building
knn_test_pred = knn(train = train_knn, test = test_knn, cl = knn_train_label,
k =7, prob = T)

knn_tab = table(knn_test_pred, knn_test_label)

##Error
1 - sum(diag(knn_tab))/sum(knn_tab)

## [1] 0.119
```

The value of k was tested for different values,

k = 3, error is 0.126

k = 5, error is 0.123

k = 7, error is 0.119

k = 9, error is 0.121

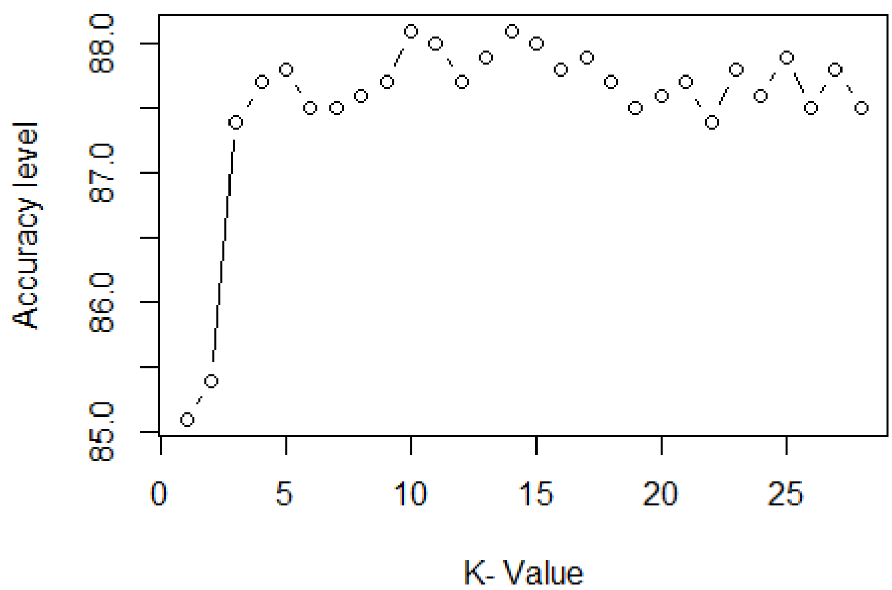
We will take the k value having the lowest error. As 0.119 is the lowest, we will take k = 7.

The accuracy of the prediction is **88.1%** and error of **11.9%**.

Optimization

The below graph is plotted with the value of k from 1 to 28, to show the values having the highest accuracy rate.

```
#Accuracy plot
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



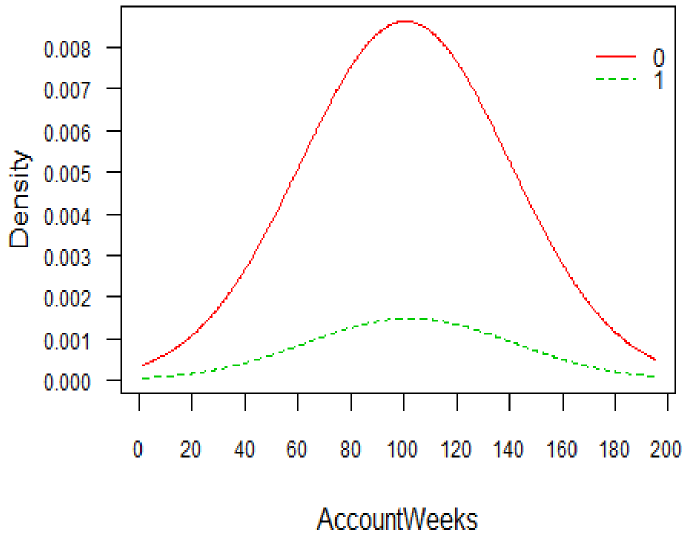
Naïve Bayes Model

Naive Bayes is a Supervised Machine Learning algorithm based on the Bayes Theorem that is used to solve classification problems by following a probabilistic approach.

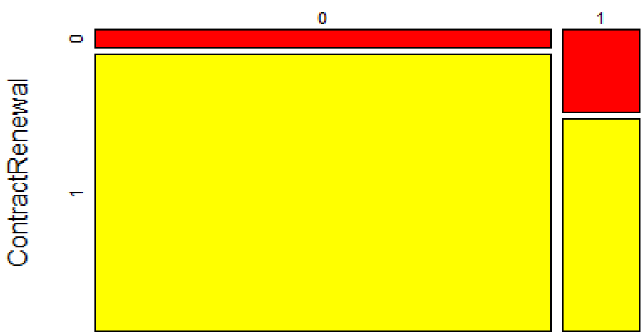
In real-world problems, predictor variables aren't always independent of each other, there are always some correlations between them. Since Naive Bayes considers each predictor variable to be independent of any other variable in the model, it is called 'Naive'.

```
#Building the Naive Bayes Model
naive_cellphone = naive_bayes(cellphone$Churn ~ ., data = cellphone,laplace =
0.5)

#Split data into train and test dataset
set.seed(1000)

nb_ds = sample.split(cellphone$Churn, SplitRatio = 0.70)
nb_train = subset(cellphone, nb_ds == T)
nb_test = subset(cellphone, nb_ds == F)

#predicting the model
naive_predict = predict(naive_cellphone, nb_train, type = 'prob')
```

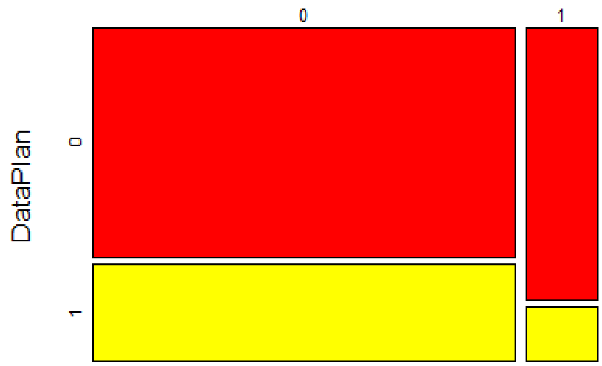
```r
#To plot the features with Naive Bayes
plot(naive_cellphone)
```



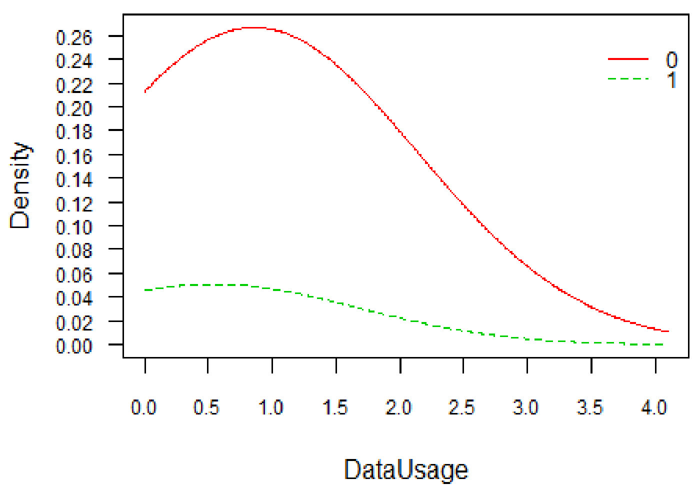
The no of Account Weeks having Churn as 0 is higher than the ones having 1. Both are normally distributed.



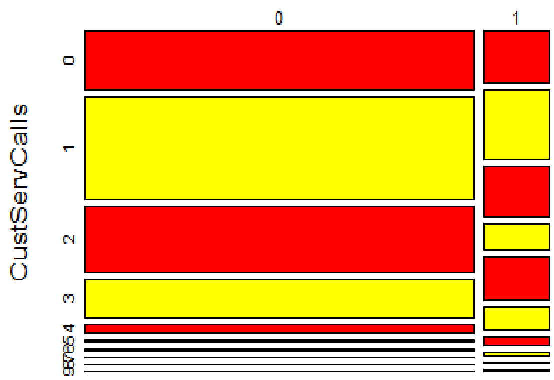
The no of customers who renewed the contract have a lower churn rate.



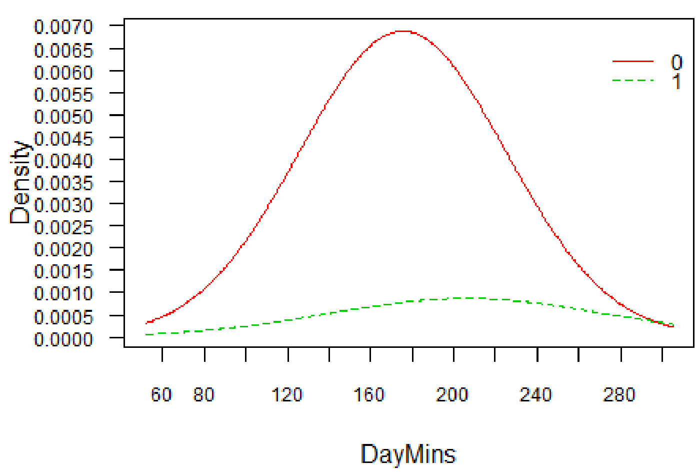
The customers having a data plan have lower churn rate.



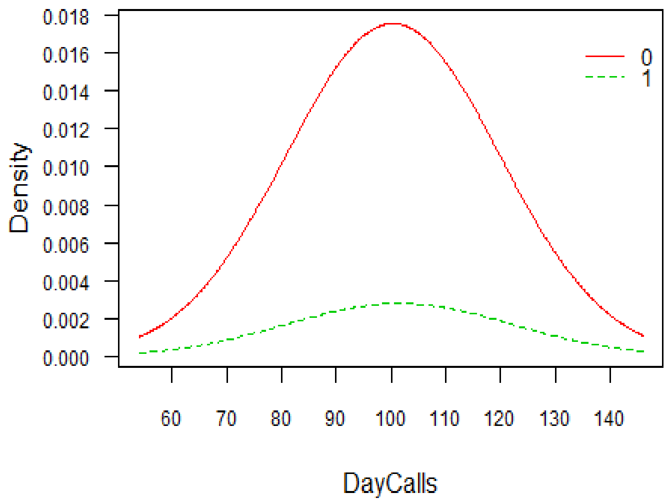
The customers having higher data usage have lower churn rate.



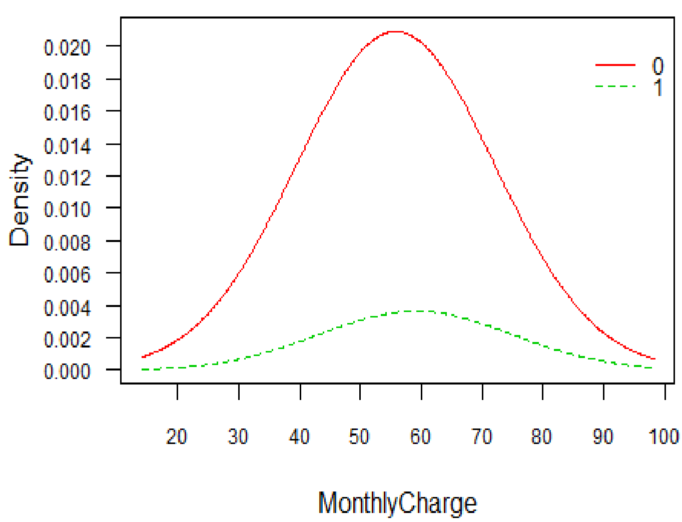
The customers having made the service calls have lower churn rate.



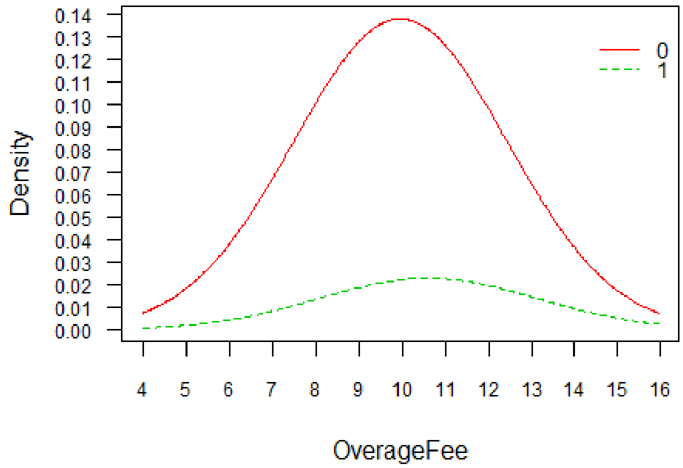
Customers having higher Day Mins have lower churn rate and it is normally distributed.



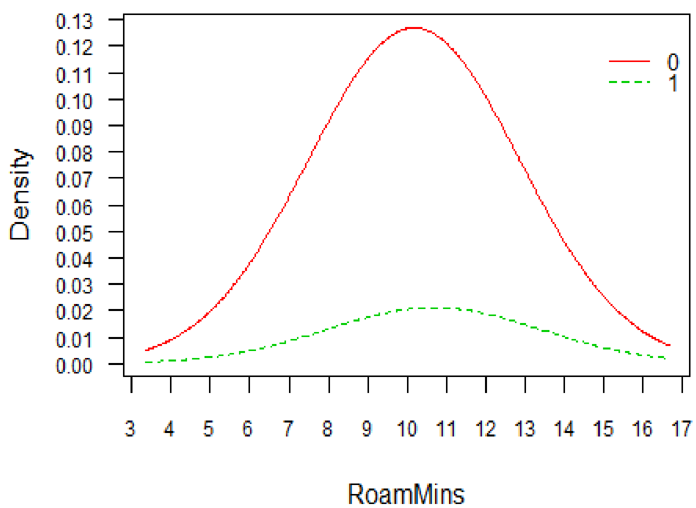
Customers having higher Day Calls have lower churn rate and it is normally distributed.



Customers having higher Monthly Charges have lower churn rate and it is normally distributed.



Customers having higher Overage Fee have lower churn rate and it is normally distributed.



Customers having higher Roam Mins have lower churn rate and it is normally distributed.

```
#Confusion Matrix - train data
predict_train =  predict(naive_cellphone, nb_train, type = "class")

(tab1 = table(predict_train, nb_train$Churn))
```

```
##
## predict_train    0    1
##             0 1940  256
##             1   55   82

#Train Error
1 - sum(diag(tab1)) / sum(tab1)

## [1] 0.1333048

#Confusion Matrix - test data
predict_test =  predict(naive_cellphone, nb_test)

(tab2 = table(predict_test, nb_test$Churn))

##
## predict_test   0   1
##            0 828 112
##            1  27  33

#Train Error
1 -  sum(diag(tab2)) / sum(tab2)

## [1] 0.139
```

For the train dataset, we have got an accuracy of **86.7%** and error of **13.33%.** While for the test dataset, the accuracy is **86.1%** and error is **13.9%.**

Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Matrix for Logistic Regression Model for train.

	False	True
0	1843	152
1	161	177

We can see that around 1843 out of 2004 are predicted as 0 while 177 out of 329 are predicted as 1. About 2020 are predicted correctly and 313 are having wrong predictions. The False Negative or Type II error is 152. The False Positive or Type I error is 161.

Matrix for Logistic Regression Model for test.

	False	True
0	789	66
1	78	67

We can see that around 789 out of 867 are predicted as 0 while 67 out of 133 are predicted as 1. About 856 are predicted correctly and 144 are having wrong predictions. The False Negative or Type II error is 66. The False Positive or Type I error is 78.

```
# Confusion matrix for threshold of 0.3
npt = table(test$Churn, predTest>0.3)

# Sensitivity
npt[2,2]/sum(npt[2,])

## [1] 0.4551724

# Specificity
npt[1,1]/sum(npt[1,])

## [1] 0.9263158
```

The confusion matrix for Logistic Regression Model gives us an accuracy of 0.856.

Matrix for KNN for test.

	False	True
0	845	109
1	10	36

We can see that around 845 out of 855 are predicted as 0 while 36 out of 145 are predicted as 1. About 881 are predicted correctly and 119 are having wrong predictions. The False Negative or Type II error is 109. The False Positive or Type I error is 10.

```
confusionMatrix(table(knn_test_pred, knn_test_label))

## Confusion Matrix and Statistics
##
##                knn_test_label
```

```
## knn_test_pred   0   1
##             0 844 114
##             1  11  31
##
##                Accuracy : 0.875
##                  95% CI : (0.8529, 0.8949)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.03795
##
##                   Kappa : 0.285
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9871
##             Specificity : 0.2138
##          Pos Pred Value : 0.8810
##          Neg Pred Value : 0.7381
##              Prevalence : 0.8550
##          Detection Rate : 0.8440
##    Detection Prevalence : 0.9580
##       Balanced Accuracy : 0.6005
##
##        'Positive' Class : 0
##
```

The confusion matrix for KNN Model gives us an accuracy of 0.85. The sensitivity is 0.9871 and the specificity is 0.2138.

Matrix for Naïve Bayes Model for train.

	False	True
0	1940	256
1	55	82

We can see that around 1940 out of 1995 are predicted as 0 while 82 out of 338 are predicted as 1. About 2022 are predicted correctly and 311 are having wrong predictions. The False Negative or Type II error is 256. The False Positive or Type I error is 55.

Matrix for Naïve Bayes Model for test.

	False	True
0	828	112
1	27	33

We can see that around 828 out of 855 are predicted as 0 while 33 out of 145 are predicted as 1. About 861 are predicted correctly and 139 are having wrong predictions. The False Negative or Type II error is 112. The False Positive or Type I error is 27.

```
confusionMatrix(predict_test, nb_test$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 828 112
##          1  27  33
##
##                Accuracy : 0.861
##                  95% CI : (0.838, 0.8819)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.3135
##
##                   Kappa : 0.2591
##
##  Mcnemar's Test P-Value : 1.042e-12
##
##             Sensitivity : 0.9684
##             Specificity : 0.2276
##          Pos Pred Value : 0.8809
##          Neg Pred Value : 0.5500
##              Prevalence : 0.8550
##          Detection Rate : 0.8280
##    Detection Prevalence : 0.9400
##       Balanced Accuracy : 0.5980
##
##        'Positive' Class : 0
##
```

The confusion matrix for Naïve Bayes Model gives us an accuracy of 0.861. The sensitivity is 0.9684 and the specificity is 0.2276.

Model Performance Comparison Metrics

```
set.seed(1000)

levels(cellphone$Churn) <- make.names(levels(cellphone$Churn))

control = trainControl(method = "repeatedcv",
                       classProbs = T,
                       number = 10,
                       summaryFunction = twoClassSummary,
                       repeats = 3)
```

Logistic Regression

```
modelLOGIT = train(Churn ~ .,
                    data = cellphone,
                    method = "regLogistic",
                    metric = 'ROC',
                    trControl = control)
summary(modelLOGIT)

##              Length Class      Mode
## TypeDetail    1     -none-     character
## Type          1     -none-     numeric
## W            19     -none-     numeric
## Bias          1     -none-     numeric
## ClassNames    2     factor     numeric
## NbClass       1     -none-     numeric
## xNames       18     -none-     character
## problemType   1     -none-     character
## tuneValue     3     data.frame list
## obsLevels     2     -none-     character
## param         0     -none-     list
```

Naïve Bayes

```
set.seed(1000)

modelNB = train(Churn ~ .,
                data = cellphone,
                method = "nb",
                metric = 'ROC',
                trControl = control)

summary(modelNB)

##              Length Class      Mode
## apriori       2     table      numeric
## tables       18     -none-     list
```

```
## levels        2      -none-     character
## call          6      -none-     call
## x            18      data.frame list
## usekernel     1      -none-     logical
## varnames     18      -none-     character
## xNames       18      -none-     character
## problemType   1      -none-     character
## tuneValue     3      data.frame list
## obsLevels     2      -none-     character
## param         0      -none-     list
```

KNN

```
set.seed(1000)

modelKNN  = train(Churn ~ .,
                  data = cellphone,
                  method = "knn",
                  metric = 'ROC',
                  trControl = control)
summary(modelKNN)
```

```
##             Length Class      Mode
## learn        2      -none-     list
## k            1      -none-     numeric
## theDots      0      -none-     list
## xNames      18      -none-     character
## problemType  1      -none-     character
## tuneValue    1      data.frame list
## obsLevels    2      -none-     character
## param        0      -none-     list
```

Collect Resamples

```
results = resamples(list(Logit = modelLOGIT,
                         NB = modelNB,
                         KNN = modelKNN))
```

Compare the results with different models

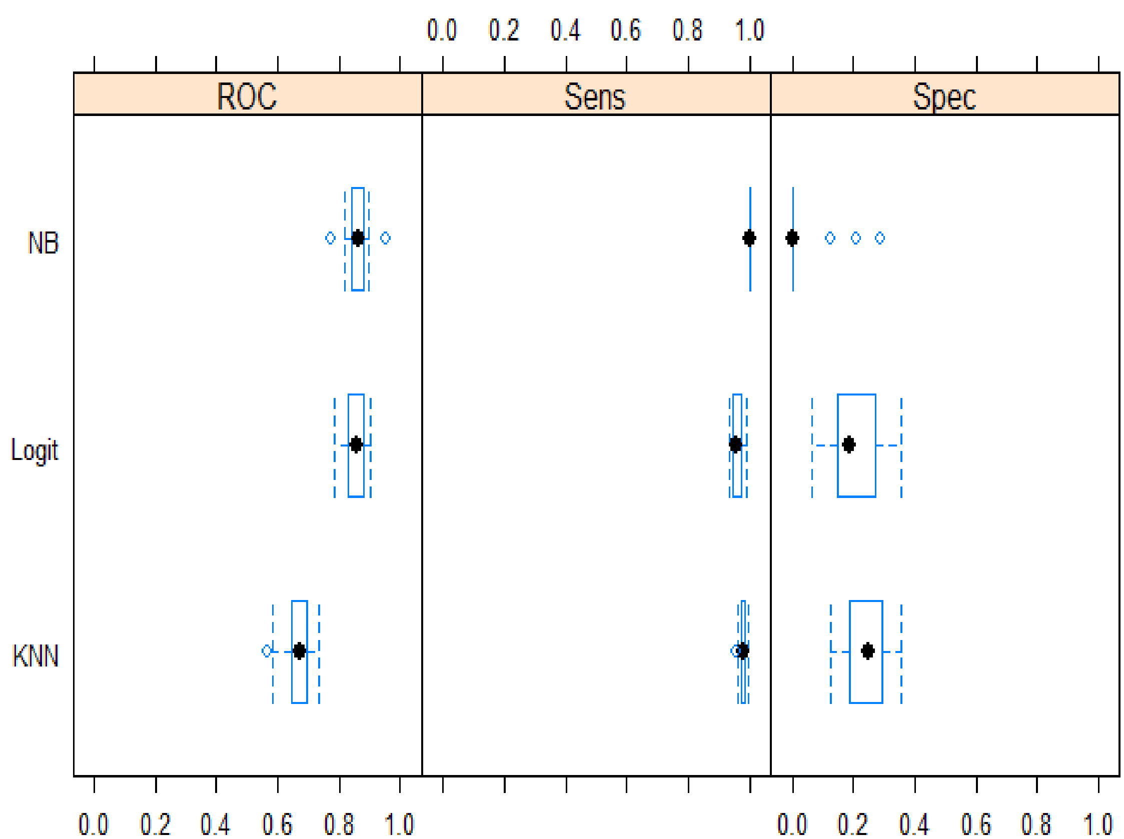
```
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Logit, NB, KNN
## Number of resamples: 30
##
```

```
## ROC
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Logit 0.7826023 0.8434569 0.8549342 0.8549058 0.8767812 0.9063373    0
## NB    0.7737573 0.8404788 0.8618056 0.8579255 0.8785863 0.9512427    0
## KNN   0.5642678 0.6476381 0.6729532 0.6676678 0.6927266 0.7323465    0
##
## Sens
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Logit 0.9192982 0.9543860 0.9596491 0.9583626 0.9649123 0.9789474    0
## NB    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0
## KNN   0.9578947 0.9754386 0.9807018 0.9791813 0.9859649 0.9964912    0
##
## Spec
##               Min.   1st Qu.    Median       Mean   3rd Qu.      Max. NA's
## Logit 0.04166667 0.1836735 0.2040816 0.21186224 0.2500000 0.3750000    0
## NB    0.00000000 0.0000000 0.0000000 0.02063492 0.0000000 0.2857143    0
## KNN   0.12500000 0.1916454 0.2474490 0.24376417 0.2916667 0.3541667    0
```

Box plot for the measures

```
bwplot(results)
```



ROC is a good way to compare model performance. The ROC curve does this by plotting ***sensitivity***, the probability of predicting a real positive will be a positive, against ***1-specificity***, the probability of predicting a real negative will be a positive.

The best decision rule is high on sensitivity and low on 1-specificity. It's a rule that predicts most true positives will be a positive and few true negatives will be a positive.

From the graph, we can see that Naïve Bayes Model has the highest ROC, with high sensitivity and low specificity. Followed by Logistic Regression Model and KNN Model.

Conclusion

The aim of Telephone Customer Churn Prediction is to identify whether a customer may cancel his services in the future. Therefore, they need information about the connection between the variables and the given data.

Three classification algorithms, Logistic Regression, K Nearest Neighbour and Naïve Bayes Forest were used for this study. All the models were trained and compared. Based on the ROC curve of the trained model, Naïve Bayes gave a better performance.

Based on the accuracy got after resolving the multi-collinearity issues and scaling the data. NB gave an accuracy of 86.1%, KNN gave an accuracy of 88.1% and LR with a prediction set to greater than 0.3 gave an accuracy of 85.6%.

As we can see, all the models are giving an accuracy of around 86-88%. We can go ahead with any model.

Naïve Bayes Model has better ROC, it considers each predictor variable to be independent of any other variable in the model. In the telecom industry, charges play a huge role. From the correlation matrix, Monthly Charges was highly correlated with Data Usage and Day Mins. Hence, we cannot ignore the correlation.

The KNN Model has the lowest ROC but has given the best accuracy.

Logistic Regression has a comparatively better ROC and accuracy is also good. This algorithm is specifically built for categorical dependent variables.

I would go for Logistic Regression Model.