# Telecom – Customer Churn Predictive Modeling

Submission Date: Jan 05, 2020

Author: Ayush Jain
Mentor: Deepak Gupta

# Table of Content

## Table of Contents

# 1 Project Objective

Customer Churn is a burning problem for Telecom companies. Here, we will 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 will perform modelling which can predict whether a customer will cancel their service in the future or not.

The Data Consumed is in the form of .xlsx file with the name “Cellphone.xlsx”. In this we will first investigate and Analyze the data to understand the insights and the readiness of the data for modelling.

We will further be performing the Data Modelling and Data Cleaning steps on the given data to ensure that the data is ready for Model Building.

Once the Data is ready, we will be building the below Models on the Data and Interpret the best Model that gives the best results.

- Logistic Regression Model
- K – Nearest Neighbor Model
- Naïve Bayes Model

The data file is Structured as below containing 3333 observations.

| S.No | Variables | Description |
|---|---|---|
| 1 | Churn | 1 if customer cancelled service, 0 if not |
| 2 | AccountWeeks | number of weeks customer has had active account |
| 3 | ContractRenewal | 1 if customer recently renewed contract, 0 if not |
| 4 | DataPlan | 1 if customer has data plan, 0 if not |
| 5 | DataUsage | gigabytes of monthly data usage |
| 6 | CustServCalls | number of calls into customer service |
| 7 | DayMins | average daytime minutes per month |
| 8 | DayCalls | average number of daytime calls |
| 9 | MonthlyCharge | average monthly bill |
| 10 | OverageFee | largest overage fee in last 12 months |
| 11 | RoamMins | average number of roaming minutes |

## 2 Exploratory Data Analysis – Step by step approach

Exploratory Data Analysis is one of the important phases in the data Analysis in understanding the significance and accuracy of the data. It usually consists of setting up the environment to work in R, loading the data and checking the validity of data loaded.

A Typical Data exploration activity consists of the following steps:

- Environment Set up and Data Import.
    - Install Necessary Package in R.
    - Setting Up Working Directory.
    - Reading Dataset in R.
    - Checking for Outliers and Null Values in the Dataset
    - Checking for Multicollinearity within the Independent variables.
    - Performing Univariate Analysis on independent variables.
    - Performing Bi-variate Analysis.
    - Preparing the Data for Model Building.

- Variable Identification.

We shall follow these steps in exploring the provided dataset.

## 2.1 Environment Set up and Data Import

### 2.1.1 Deploying necessary Packages in R.

In this section, we will install and invoke the necessary Packages and Libraries that are going to be the part of our work throughout the project. Having all the packages at the same places increases code readability and Understandability.

```r
# Deploying necessary Libraries
library(readxl)
library(DataExplorer)
library(corrplot)
library(dplyr)
library(InformationValue)
library(caret)
library(ROCR)
library(caTools)
library(blorr)
library(MASS)
library(class)
library(e1071)
library(car)
library(lmtest)
library(pscl)
library(ineq)
```

### 2.1.2 Setting Up Working Directory.

Setting a working directory on starting of the R session makes importing and exporting data files and code files easier. Basically, working directory is the location/ folder on the PC where you have the data, codes etc. related to the project. This helps maintain the code readability and avoid unwanted errors.

```r
### Setting up Working Directory

setwd("D:/Great Learning/Predictive Modeling/Project 4")
```

### 2.1.3 Reading Dataset in R.

The given dataset is in .xlsx format. Hence, the command 'read.xslx' from readxl package is used for importing the file.

```r
### Reading the Dataset from Excel

cellphone <- read_xlsx("Cellphone.xlsx",sheet = 2)
```

### 2.1.4 Performing basic Data checks.

This section of the report checks for the basic steps to ensure that the data is imported properly and also checks the Structure of the dataset and Summary to have the basic understanding of the Data.

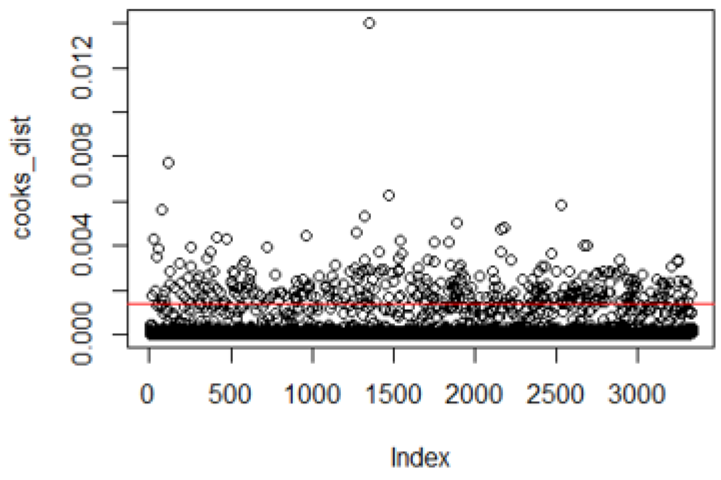
```
head(cellphone)

## # A tibble: 6 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
## # ... with 5 more variables: DayMins <dbl>, DayCalls <dbl>,
## #   MonthlyCharge <dbl>, OverageFee <dbl>, RoamMins <dbl>

str(cellphone)

## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11 variables:
##  $ Churn          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
##  $ ContractRenewal: num  1 1 1 0 0 0 1 0 1 0 ...
##  $ DataPlan       : num  1 1 0 0 0 0 1 0 0 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
##  $ CustServCalls  : num  1 1 0 2 3 0 3 0 1 0 ...
##  $ DayMins        : num  265 162 243 299 167 ...
##  $ DayCalls       : num  110 123 114 71 113 98 88 79 97 84 ...
##  $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
```

```
summary(cellphone)

##      Churn          AccountWeeks    ContractRenewal     DataPlan
##  Min.   :0.0000   Min.   :  1.0   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.: 74.0   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.0000   Median :101.0   Median :1.0000   Median :0.0000
##  Mean   :0.1449   Mean   :101.1   Mean   :0.9031   Mean   :0.2766
##  3rd Qu.:0.0000   3rd Qu.:127.0   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :243.0   Max.   :1.0000   Max.   :1.0000
##    DataUsage      CustServCalls      DayMins          DayCalls
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.0   Min.   :  0.0
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:143.7   1st Qu.: 87.0
##  Median :0.0000   Median :1.000   Median :179.4   Median :101.0
##  Mean   :0.8165   Mean   :1.563   Mean   :179.8   Mean   :100.4
##  3rd Qu.:1.7800   3rd Qu.:2.000   3rd Qu.:216.4   3rd Qu.:114.0
##  Max.   :5.4000   Max.   :9.000   Max.   :350.8   Max.   :165.0
##  MonthlyCharge      OverageFee       RoamMins
##  Min.   : 14.00   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 45.00   1st Qu.: 8.33   1st Qu.: 8.50
##  Median : 53.50   Median :10.07   Median :10.30
##  Mean   : 56.31   Mean   :10.05   Mean   :10.24
##  3rd Qu.: 66.20   3rd Qu.:11.77   3rd Qu.:12.10
##  Max.   :111.30   Max.   :18.19   Max.   :20.00
```

### 2.1.5 Checking for Outliers and Null Values in the Dataset.

Outliers: An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Examination of the data for unusual observations that are far removed from the mass of data. These points are often referred to as outliers.

```r
# Performing Cooks Distance to check the Outliers.
cellphone$Churn<- as.numeric(cellphone$Churn)
cook_lm <- lm(cellphone$Churn~. -ContractRenewal -DataPlan, data =
cellphone,)

cooks_dist<- cooks.distance(cook_lm)

plot(cooks_dist)

abline(h=4*mean(cooks_dist,na.rm = TRUE),col="red")
```



```r
influential <- as.numeric(names(cooks_dist)[(cooks_dist > 4*mean(cooks_dist,
na.rm=T))])

out <- (cellphone[influential, ])

View(out)

cellphone$Churn<- as.factor(cellphone$Churn)
```

We found 341 observations that are present as the Outliers in the dataset, which is >10% of the overall data. We will not be performing any treatment on this as all the models that we will be building are immune to Outliers and have no impact on the performance.

Null Values: These are the missing values in the dataset that needs to be treated to get the proper accuracy of the Model.

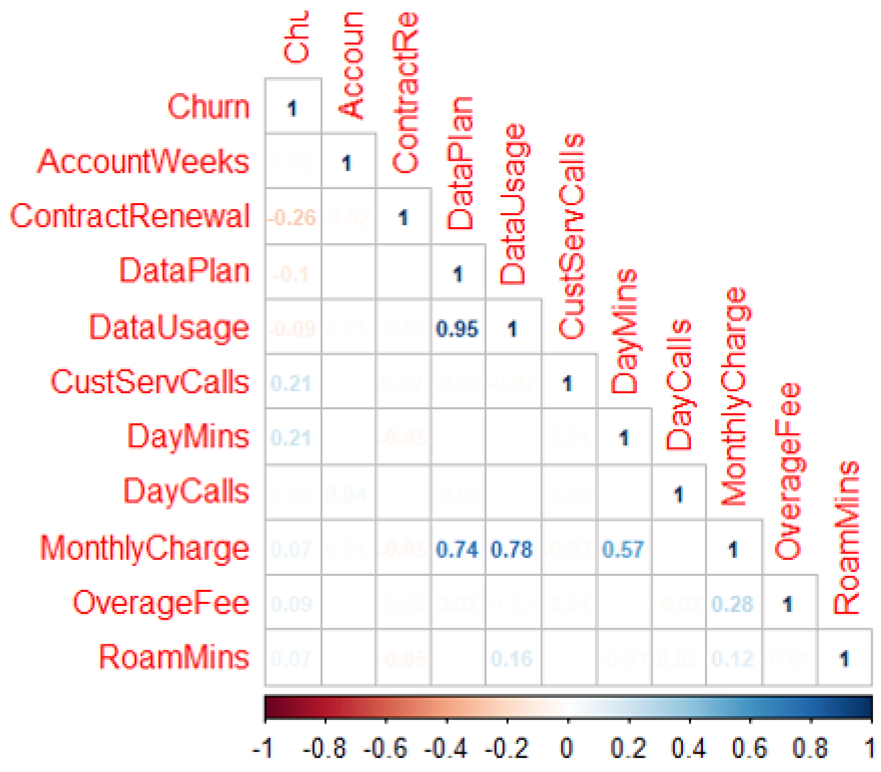
```r
# Checking for any Null Values in the data

sum(is.na(cellphone))

## [1] 0
```

Since there are no Null Values present, we will not be performing any treatment on the data.

### 2.1.6 Checking for Multicollinearity within the Independent variables.

Multicollinearity is a state of very high intercorrelations or inter-associations among the independent variables. It is therefore a type of disturbance in the data, and if present in the data the statistical inferences made about the data may not be reliable.

```r
# Checking for Multicollinearity

mat<- cor(cellphone)
corrplot(mat,method = "number",type = "lower", number.cex = 0.7)
```



### 2.1.7 Performing Univariate Analysis on independent variables.

Univariate analysis is perhaps the simplest form of statistical analysis. Like other forms of statistics, it can be inferential or descriptive. The key fact is that only one variable is involved.

For Numeric variables, default plot is histogram and boxplot while for Categorical variables it is Bar plot.

**Histogram**: A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.

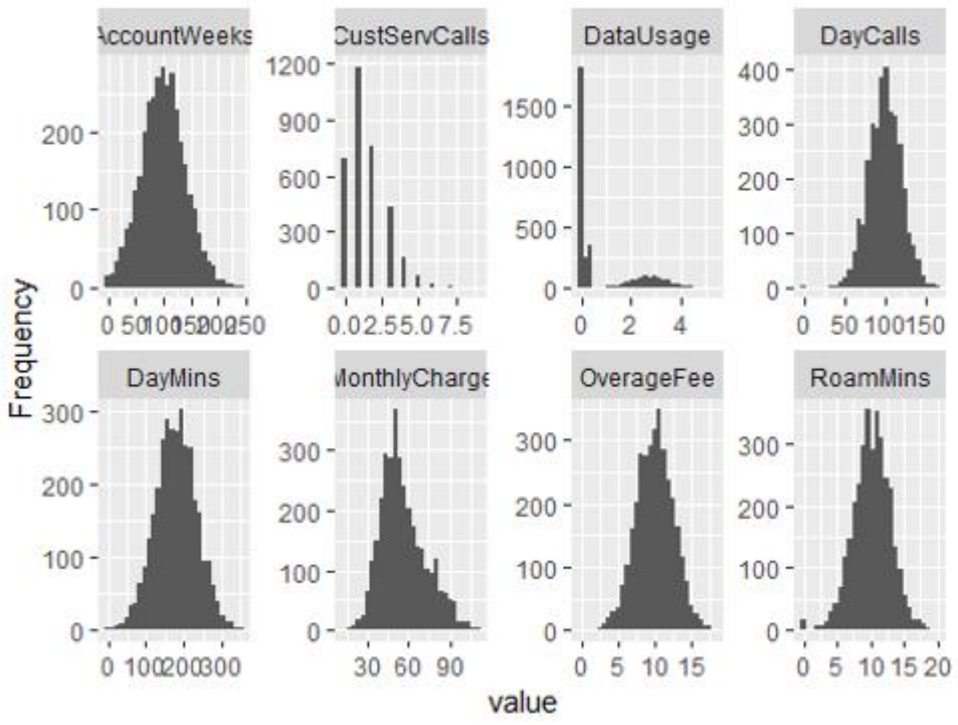
**Boxplot**: A box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles. Outliers may be plotted as individual points.

In the problem given, we will be using the above two plotting functions to perform the Univariate analysis on the dataset and identify any outliners present in the data.

**Plotting the histogram for all the numeric variables in the dataset.**

To analyze each variable, we plot the histogram for the variables.

```
# Performing Univariate Analysis

plot_histogram(cellphone)
```



## 2.1.8 Performing Bi-variate Analysis.

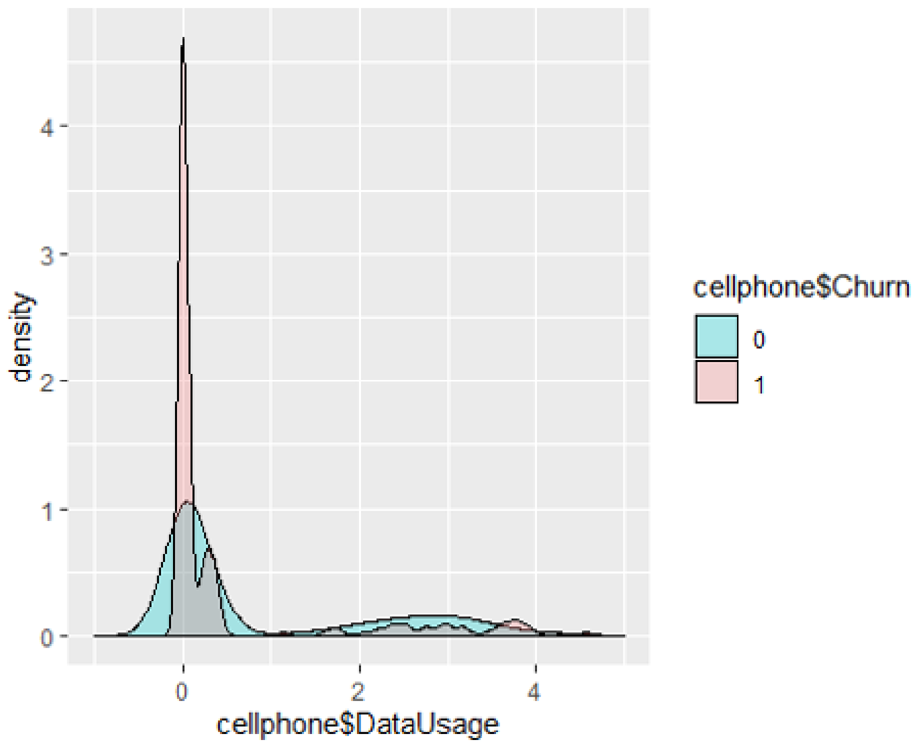
Multivariate analysis is a set of techniques used for analysis of data sets that contain more than one variable, and the techniques are especially valuable when working with correlated variables. The techniques provide an empirical method for information extraction, regression, or classification.

For Multivariate analysis, the default plot is the Scatter Plot or Density Plot. We will be plotting the correlation between the different variables with Churn to understand the relation between the dependent variable Churn with the Independent variables.

- Density Plot of Data Usage with respect to Churn.

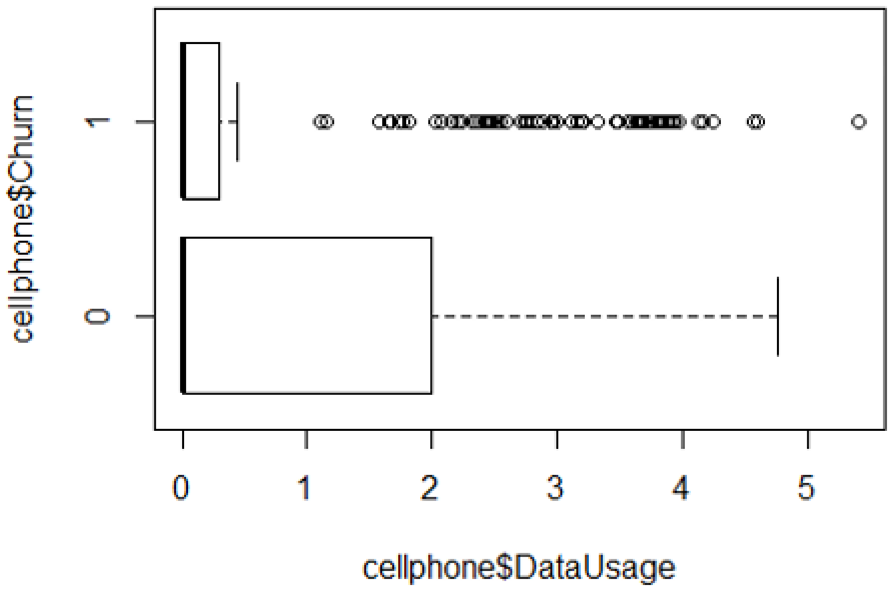
```
ggplot(cellphone, aes(x = cellphone$DataUsage)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_color_manual(values = c("#868686FF", "#EFC000FF")) +
  scale_fill_manual(values = c("darkturquoise", "lightcoral")) + xlim(-1,5)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



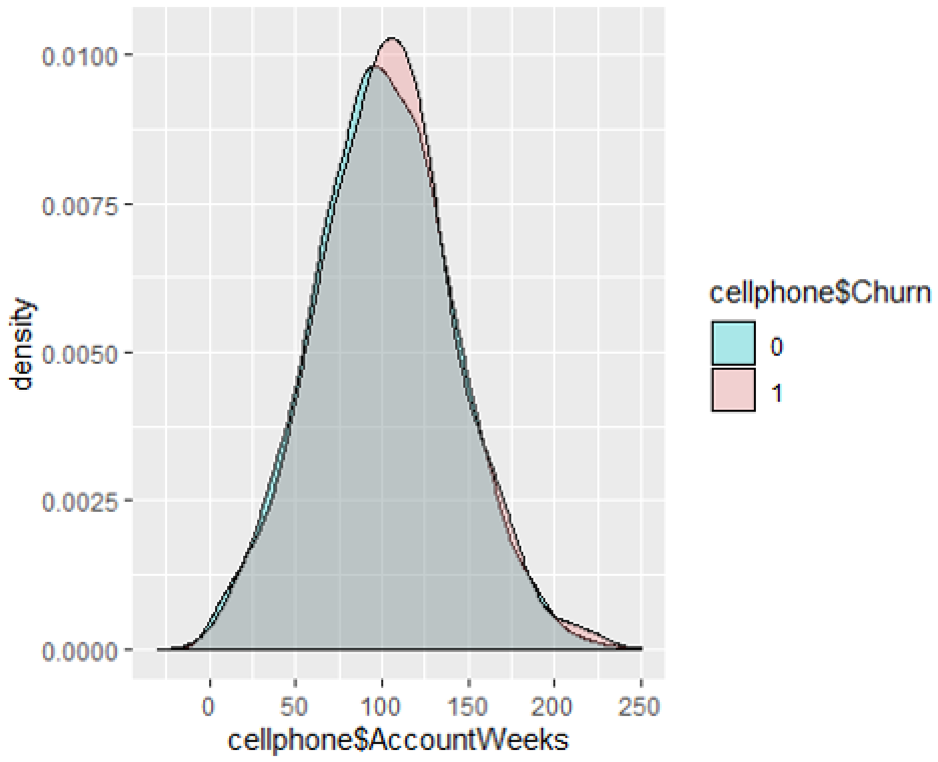
- Box Plot of Data Usage with respect to Churn.

```r
boxplot(cellphone$DataUsage~cellphone$Churn,horizontal = TRUE)
```



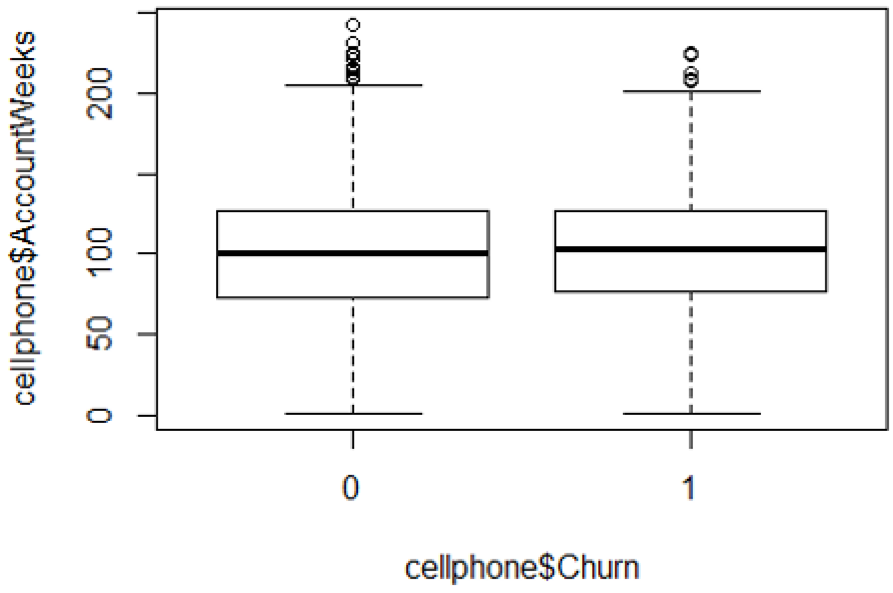
➢ Density Plot of Account Weeks with respect to Churn.

```r
ggplot(cellphone, aes(x = cellphone$AccountWeeks)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) + xlim(-30,250)
```



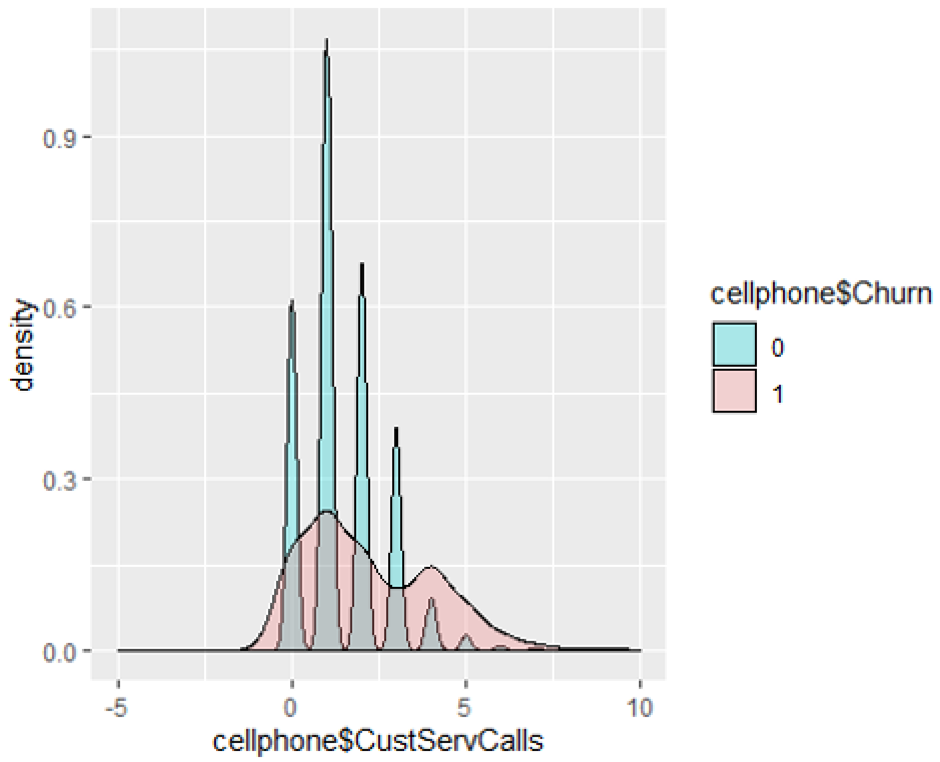
➢ Box Plot of Account Weeks with respect to Churn.

```
boxplot(cellphone$AccountWeeks~cellphone$Churn)
```



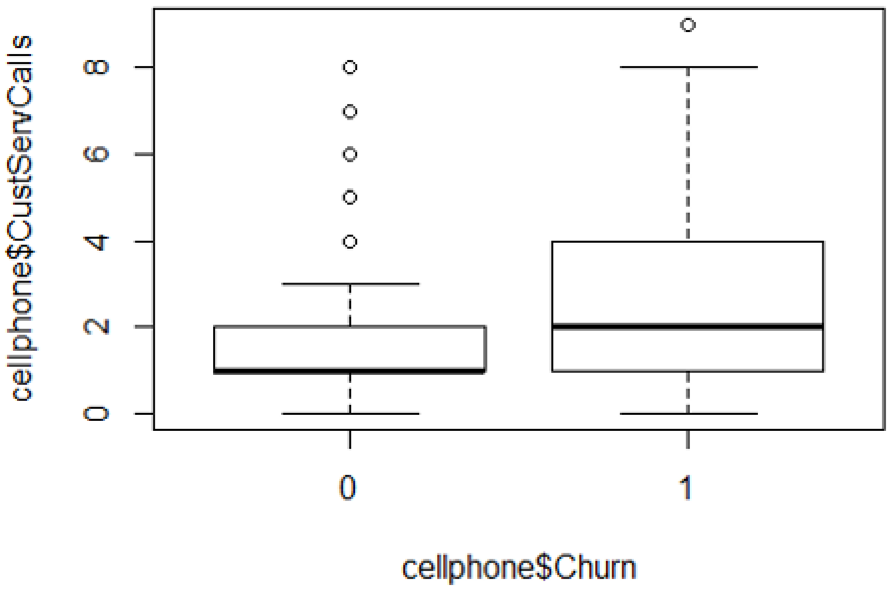
➢ Density Plot of Cust Serv Calls with respect to Churn.

```
ggplot(cellphone, aes(x = cellphone$CustServCalls)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) + xlim(-5,10)
```



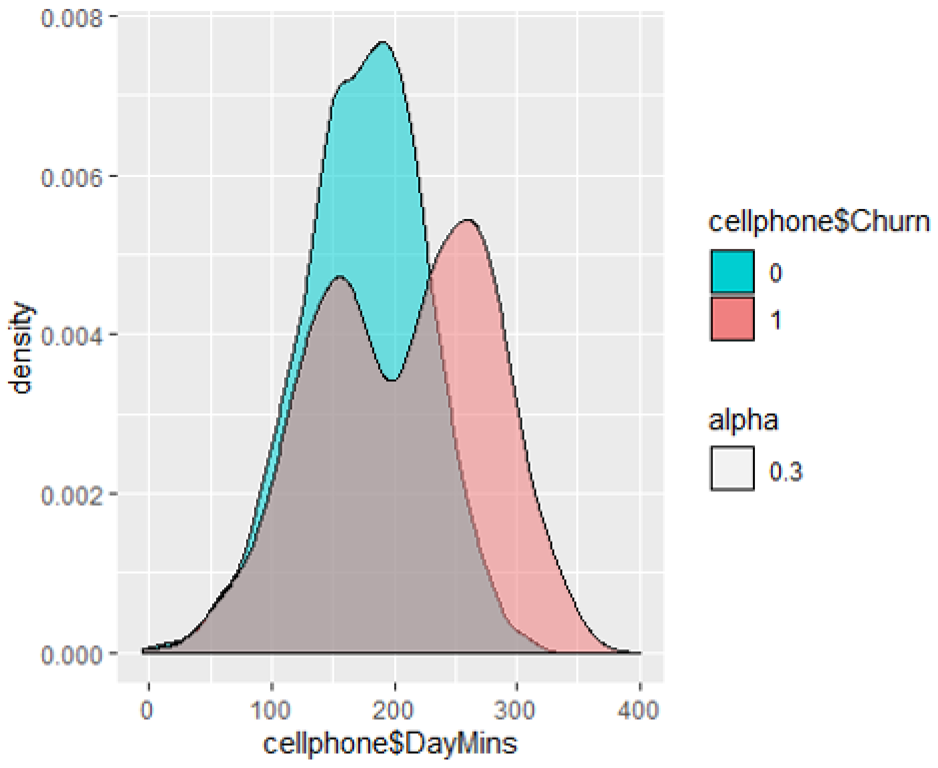
➢ Box Plot of Cust Serv Calls with respect to Churn.

```
boxplot(cellphone$CustServCalls~cellphone$Churn)
```



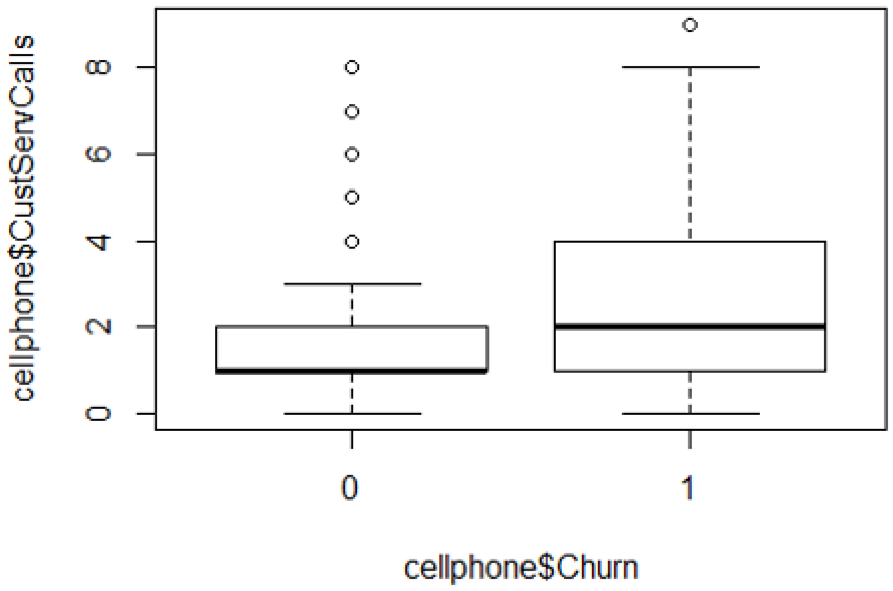
- Density Plot of Day Mins with respect to Churn.

```
ggplot(cellphone,aes(x = cellphone$DayMins)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,400)
```



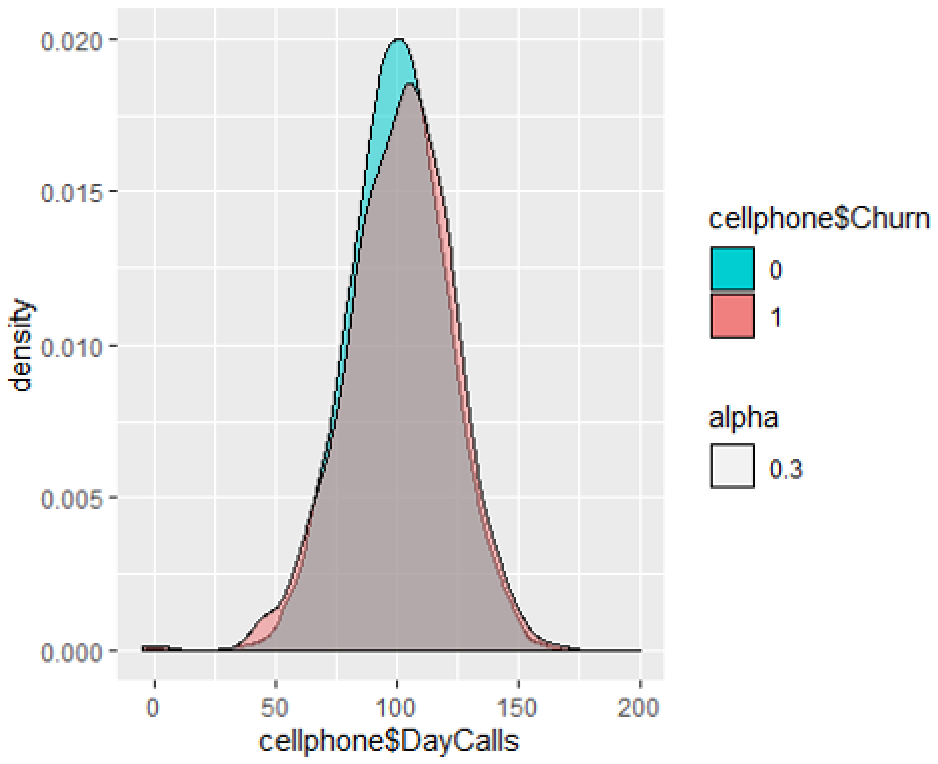
- Box Plot of Day Mins with respect to Churn.

```
boxplot(cellphone$CustServCalls~cellphone$Churn)
```



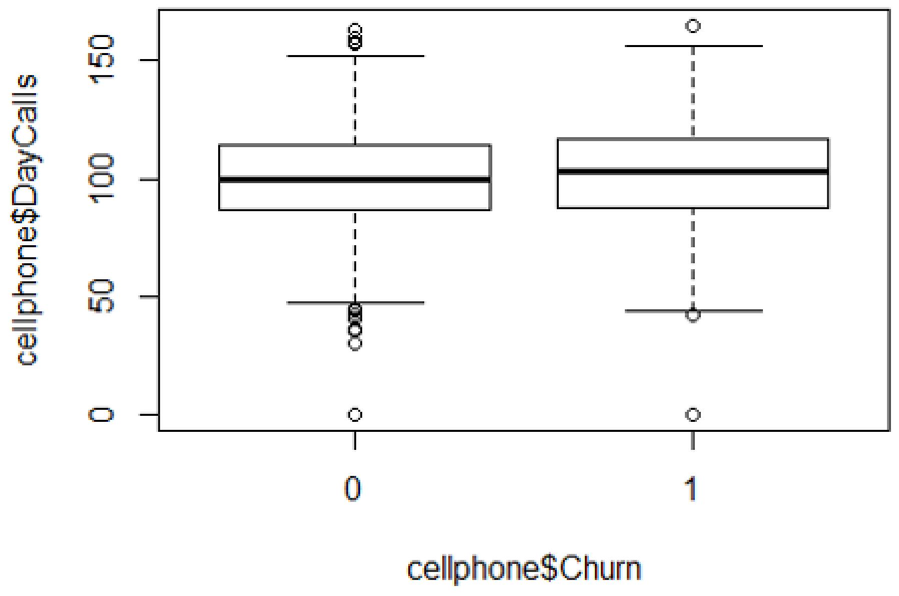
➢ Density Plot of Day Calls with respect to Churn.

```
ggplot(cellphone,aes(x = cellphone$DayCalls)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,200)
```



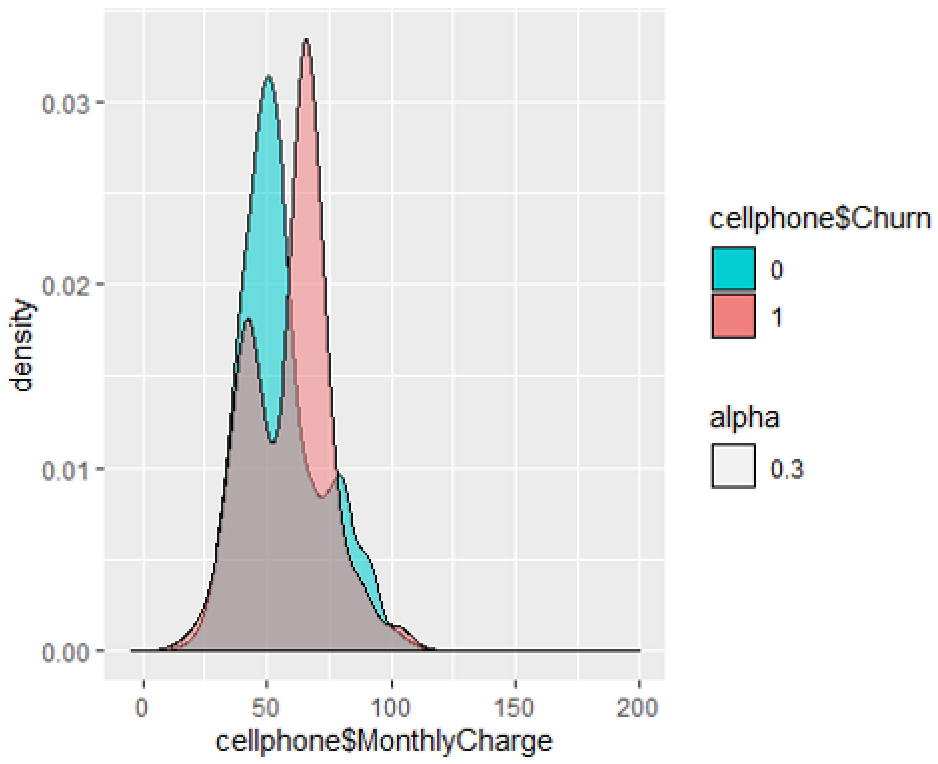
➢ Box Plot of Day Calls with respect to Churn.

```
boxplot(cellphone$DayCalls~cellphone$Churn)
```



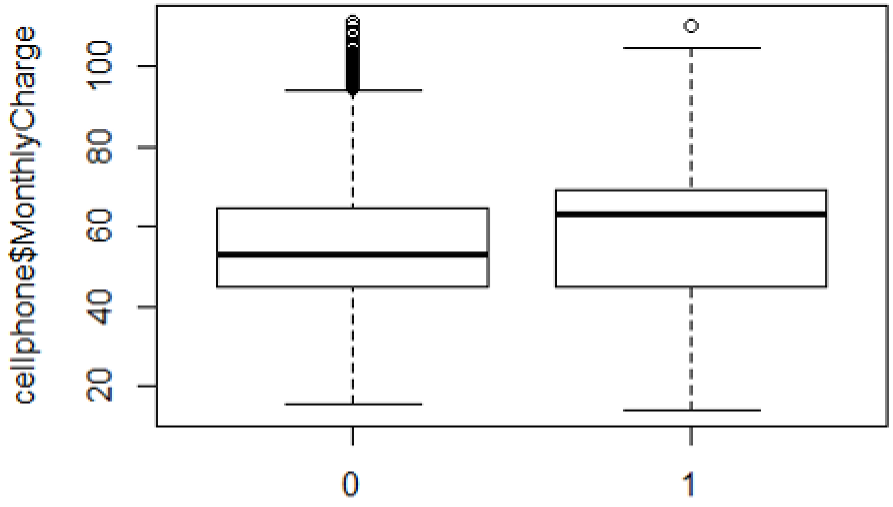
➢ Density Plot of Monthly Charges with respect to Churn.

```
ggplot(cellphone,aes(x = cellphone$MonthlyCharge)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,200)
```



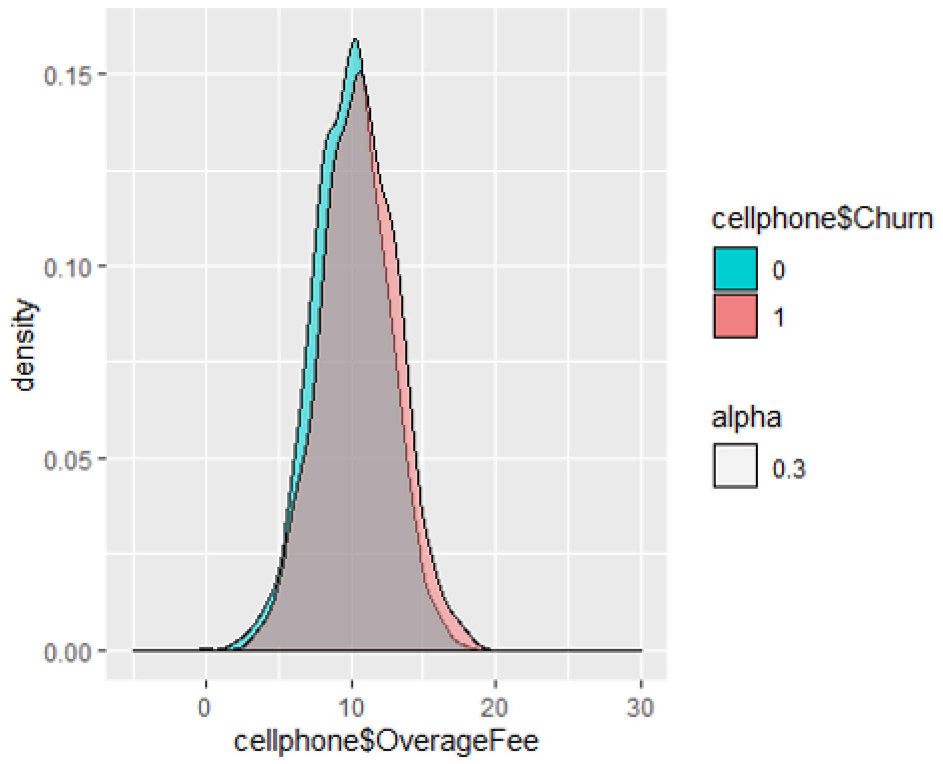
➢ Box Plot of Monthly Charges with respect to Churn.

```
boxplot(cellphone$MonthlyCharge~cellphone$Churn)
```



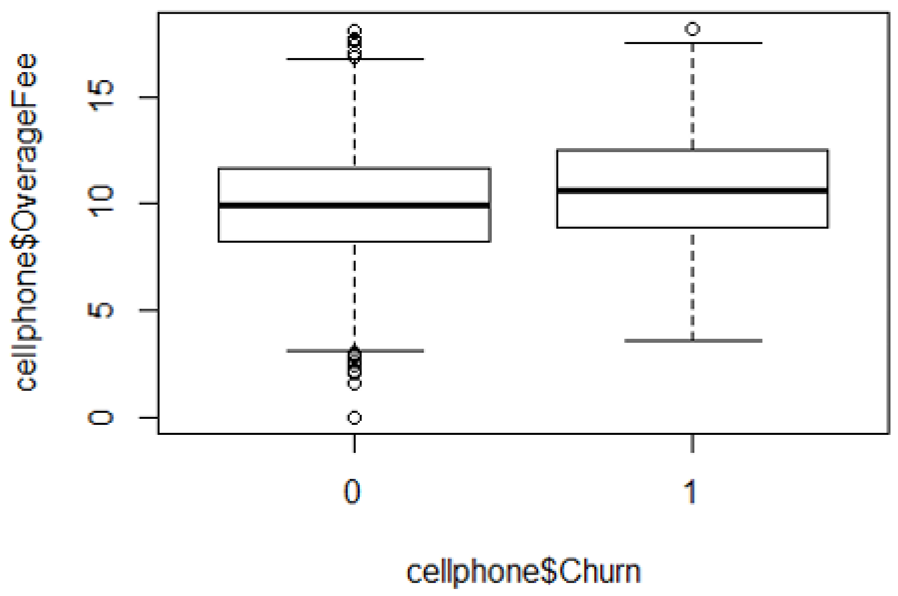
➢ Density Plot of Overage fee with respect to Churn.

```
ggplot(cellphone,aes(x = cellphone$OverageFee)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,30)
```



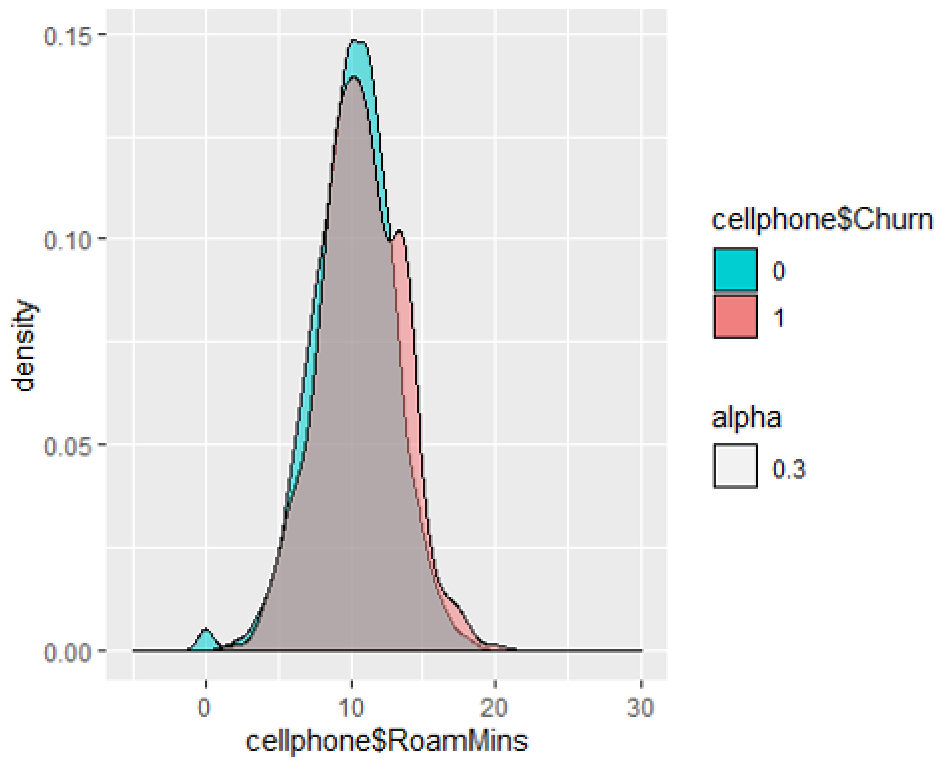
- Box Plot of Data Usage with respect to Churn.

```
boxplot(cellphone$OverageFee~cellphone$Churn)
```



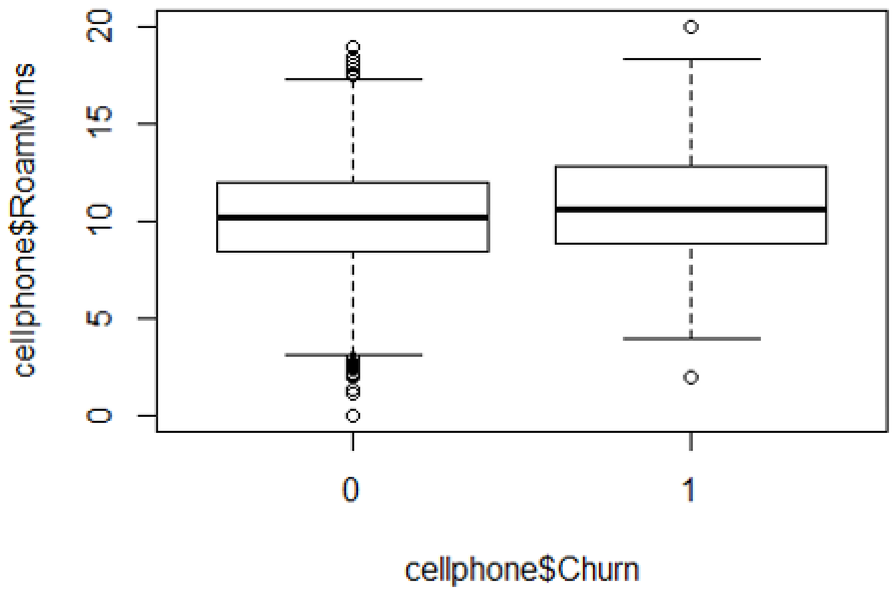
➢ Density Plot of Overage fee with respect to Churn.

```
ggplot(cellphone,aes(x = cellphone$RoamMins)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,30)
```



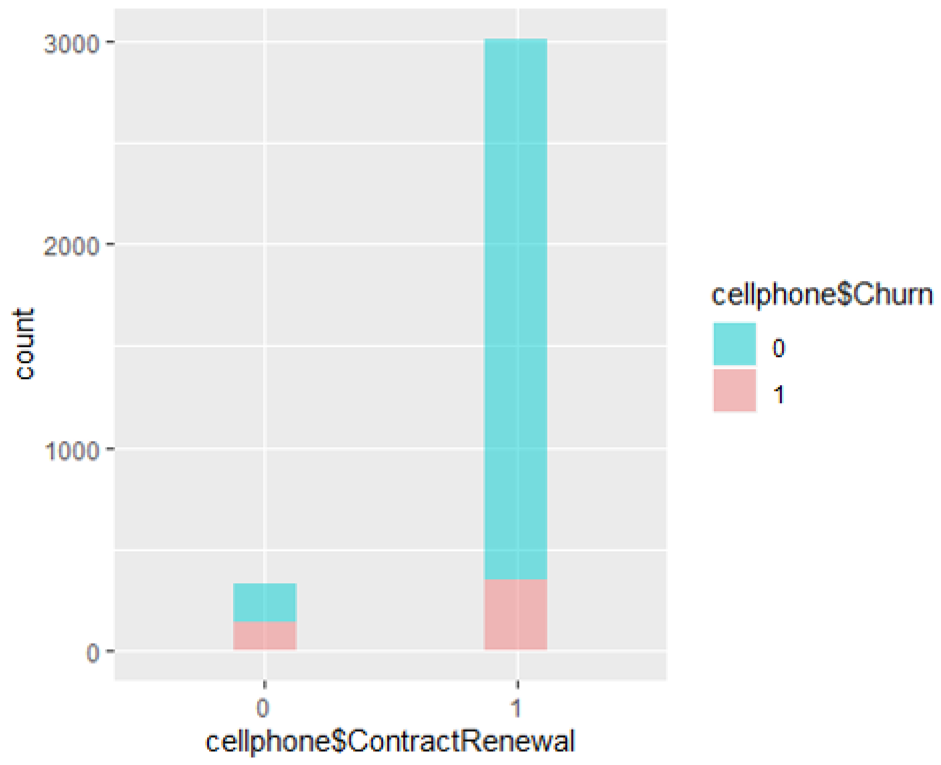
➢ Box Plot of Overage fee with respect to Churn.

```
boxplot(cellphone$RoamMins~cellphone$Churn)
```



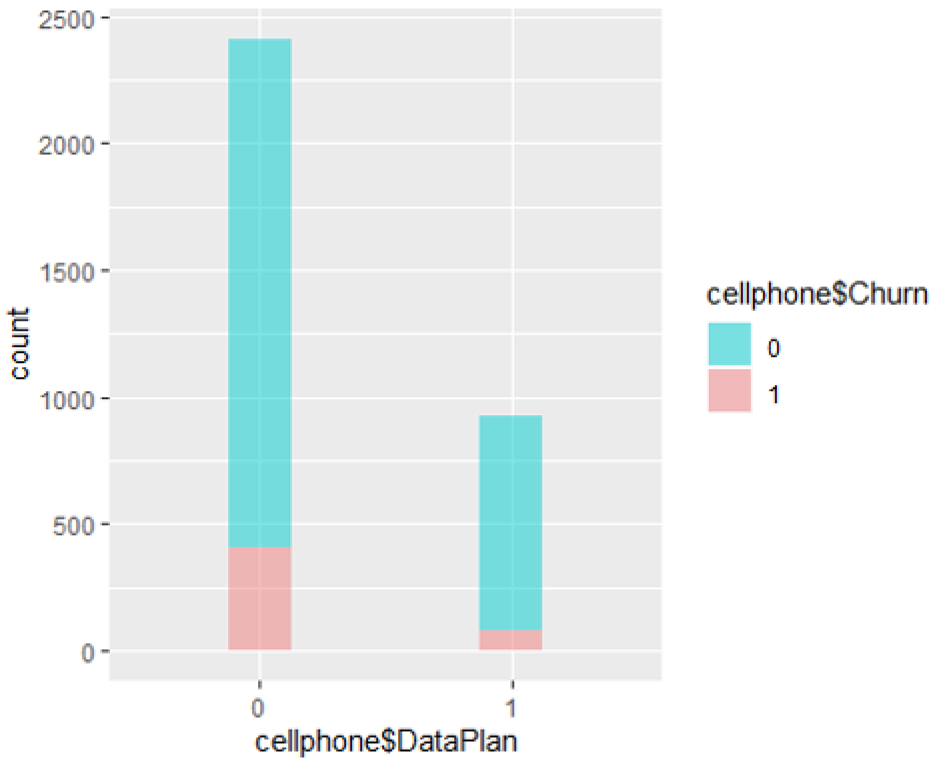
➢ Bar Plot of Categorical Variables with respect to Churn.

```
# Performing Bi-Variate analysis on Categorical Variables

ggplot(cellphone,aes(x = cellphone$ContractRenewal, fill = cellphone$Churn))
+
  geom_bar(width = 0.25,alpha = 0.5) +
  scale_fill_manual(values = c("darkturquoise","lightcoral"))
```



➢ Bar Plot of Categorical Variables with respect to Churn.

```
ggplot(cellphone, aes(x = cellphone$DataPlan, fill = cellphone$Churn)) +
  geom_bar(width = 0.25, alpha=0.5) +
  scale_fill_manual(values = c('darkturquoise', 'lightcoral'))
```



## 2.1.9 Preparing the Data for Model Building.

In this part of the report we will be changing the Categorical variables to factors from numbers and dividing the dataset into Train and Test dataset.

```
# Converting Binary variables to Factors

cellphone$Churn <- as.factor(cellphone$Churn)
cellphone$ContractRenewal <- as.factor(cellphone$ContractRenewal)
cellphone$DataPlan <- as.factor(cellphone$DataPlan)
```

```
#Splitting the Data to Test and Train.

set.seed(1234)

index<- sample.split(cellphone$Churn,SplitRatio = .70)

trainData <- subset(cellphone, index == TRUE)
testData <- subset(cellphone,index == FALSE)

str(trainData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    2333 obs. of  11 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ AccountWeeks   : num  128 107 137 84 118 121 147 117 141 65 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 2 1 2 ...
##  $ DataPlan       : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 1 1 2 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 2.03 0 0.19 3.02 0.29 ...
##  $ CustServCalls  : num  1 1 0 2 0 3 0 1 0 4 ...
##  $ DayMins        : num  265 162 243 299 223 ...
##  $ DayCalls       : num  110 123 114 71 98 88 79 97 84 137 ...
##  $ MonthlyCharge  : num  89 82 52 57 57 87.3 36 63.9 93.2 44.9 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 11.03 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 6.3 7.5 7.1 8.7 11.2 12.7 ...
```

## 2.2 Variable Identification

This section holds the Methods that are used during the Analysis of the problem. Below are the Functions that we have used for the Analysis.

- **setwd()**
  Set the working directory to dir.
- **read.xlsx()**
  Reads a file in table format and creates a data frame from it.
- **head()**
  Returns the first parts of a vector, matrix, table, data frame or function.
- **str()**
  Compactly display the internal Structure of an R object
- **summary()**
  Summary is a generic function used to produce result summaries of the results of various model fitting functions.
- **sum()**
  Sum returns the sum of all the values present in its arguments.
- **colsum()**
  Form row and column sums and means for numeric arrays.

- **is.null()**
  NULL is often returned by expressions and functions whose value is undefined. is.null returns TRUE if its argument's value is NULL and FALSE otherwise.
- **boxplot()**
  It is plotting technique, which is used to identify if there any outliners are present in the data.
- **plot_histogram()**
  Plot histogram for each continuous feature on a single area.
- **cor()**
  cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices then the covariances (correlations) between the columns of x and the columns of y are computed.
- **corrplot()**
  This is used to plot the correlation matrix for better visualization and presentation.
- **set.seed()**
  set.seed is the recommended way to specify seeds.
- **sample.split()**
  Split data from vector Y into two sets in predefined ratio while preserving relative ratios of different labels in Y. Used to split the data used during classification into train and test subsets.
- **cooks.distance()**
  Cooks Distance is the method to identify the Outliers in the data.
- **glm()**
  glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. Here we will be building the Logistic Model using this
- **vif()**
  Calculates variance-inflation and generalized variance-inflation factors for linear, generalized linear, and other models.
- **blr_step_aic_both()**
  Build regression model from a set of candidate predictor variables by entering and removing predictors based on akaike information criterion, in a stepwise manner until there is no variable left to enter or remove any more.
- **lrtest()**
  lrtest is a generic function for carrying out likelihood ratio tests. The default method can be employed for comparing nested (generalized) linear models (see details below).
- **pR2()**

Compute various pseudo-R2 measures for various GLMs

- **predict()**

  predict is a generic function for predictions from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.
- **table()**

  table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.
- **prediction()**

  Every classifier evaluation using ROCR starts with creating a prediction object. This function is used to transform the input data (which can be in vector, matrix, data frame, or list form) into a standardized format.
- **performance():**

  All kinds of predictor evaluations are performed using this function.
- **ineq()**

  Computes the inequality within a vector according to the specified inequality measure. Used to Calculate Gini Gain for the Model.
- **concordance()**

  Computes the inequality within a vector according to the specified inequality measure.
- **confusionmatrix()**

  Calculate the confusion matrix for the fitted values for a logistic regression model.
- **scale()**

  scale is generic function whose default method centers and/or scales the columns of a numeric matrix.
- **cbind()**

  Take a sequence of vector, matrix or data-frame arguments and combine by columns or rows, respectively. These are generic functions with methods for other R classes.
- **traincontrol()**

  Control the computational nuances of the train function
- **train()**

  This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.
- **naïveBayes()**

  Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

## 3 Conclusion

Once the Data is ready and divided into the Train and Test, we will further be performing the given Models on the data and check which Model performs the best in Train and Test and we will also be performing the various Performance measures to validate the model.

- Logistic Model
- K- Nearest Neighbor model
- Naïve Bayes Model

### 3.1 Logistic Model

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

#### 3.1.1 **Building Logistic Model on Train Data**

```
# Creating the Logistic Regression Model on Train data.

logModel <- glm(trainData$Churn~., data = trainData,family = "binomial")

summary(logModel)

##
## Call:
## glm(formula = trainData$Churn ~ ., family = "binomial", data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0417  -0.5097  -0.3504  -0.2088   3.0459
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -6.1567684  0.6674321  -9.225  < 2e-16 ***
## AccountWeeks    -0.0001507  0.0016498  -0.091  0.92722
## ContractRenewal1 -1.9669347  0.1762725 -11.158  < 2e-16 ***
## DataPlan1       -0.6793909  0.6385085  -1.064  0.28732
## DataUsage        1.3441189  2.3006497   0.584  0.55906
## CustServCalls    0.5529267  0.0468420  11.804  < 2e-16 ***
## DayMins          0.0360104  0.0387869   0.928  0.35319
## DayCalls         0.0054809  0.0032975   1.662  0.09648 .
## MonthlyCharge   -0.1399872  0.2280180  -0.614  0.53926
## OverageFee       0.4001170  0.3896952   1.027  0.30454
## RoamMins         0.0715722  0.0265797   2.693  0.00709 **
## ---
## 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: 1533.6  on 2322  degrees of freedom
## AIC: 1555.6
##
## Number of Fisher Scoring iterations: 5
```

The first Model has been created with all the Independent Variables and we identified that there were only a few variables that are significant to the model.

We will further be checking the Multicollinearity in the variables using the Variance Inflation Factors aka VIF.

```
# Checking for Multicollinearity using VIF

vif(logModel)

##    AccountWeeks ContractRenewal        DataPlan       DataUsage 
##        1.002719        1.060203       14.134368     1581.374480 
##    CustServCalls          DayMins        DayCalls   MonthlyCharge 
##        1.096619       938.800975        1.005125     2719.647020 
##      OverageFee         RoamMins 
##      207.886910        1.197360
```

We found that the variables "Data Usage" and "Monthly Charges" are extremely correlated and hence needs to be treated.

**Model Refinement**

Now we will perform the Step wise AIC technique to check for the significant variables for the model and will rebuild the model again on those variables.

```
# Model refining - Logistic Model

# Performing Step Wise AIC on Logistic Model
blr_step_aic_both(logModel)
```

```
## Stepwise Selection Method
## -------------------------
##
## Candidate Terms:
##
## 1 . AccountWeeks
## 2 . ContractRenewal
## 3 . DataPlan
## 4 . DataUsage
## 5 . CustServCalls
## 6 . DayMins
## 7 . DayCalls
## 8 . MonthlyCharge
## 9 . OverageFee
## 10 . RoamMins
##
##
## Variables Entered/Removed:
##
## - CustServCalls added
## - ContractRenewal added
## - DayMins added
## - OverageFee added
## - DataPlan added
## - RoamMins added
## - DayCalls added
##
## No more variables to be added or removed.
```

```
##
##
##                          Stepwise Summary
## ------------------------------------------------------------------
## Variable              Method       AIC          BIC        Deviance
## ------------------------------------------------------------------
## CustServCalls        addition    1821.620     1833.130     1817.620
## ContractRenewal      addition    1700.531     1717.795     1694.531
## DayMins              addition    1609.925     1632.945     1601.925
## OverageFee           addition    1580.904     1609.678     1570.904
## DataPlan             addition    1557.154     1591.684     1545.154
## RoamMins             addition    1550.903     1591.187     1536.903
## DayCalls             addition    1550.107     1596.146     1534.107
## ------------------------------------------------------------------
```

## Rebuilding the Model on the variables achieved from Step wise AIC technique

```
logModel2 <- glm(trainData$Churn ~ trainData$ContractRenewal +
trainData$DataPlan +trainData$CustServCalls+trainData$DayMins+
trainData$DayCalls+ trainData$OverageFee+ trainData$RoamMins,
                 data= trainData,
                 family = "binomial")
summary(logModel2)
```

```
## Call:
## glm(formula = trainData$Churn ~ trainData$ContractRenewal +
trainData$DataPlan +
##     trainData$CustServCalls + trainData$DayMins + trainData$DayCalls +
##     trainData$OverageFee + trainData$RoamMins, family = "binomial",
##     data = trainData)
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0392  -0.5088  -0.3515  -0.2083   3.0190
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -6.212947   0.636348  -9.763  < 2e-16 ***
## trainData$ContractRenewal1 -1.966915   0.175849 -11.185  < 2e-16 ***
## trainData$DataPlan1        -0.834454   0.170899  -4.883 1.05e-06 ***
## trainData$CustServCalls     0.552820   0.046818  11.808  < 2e-16 ***
## trainData$DayMins           0.012211   0.001291   9.457  < 2e-16 ***
## trainData$DayCalls          0.005498   0.003294   1.669   0.0952 .
## trainData$OverageFee        0.161485   0.027597   5.851 4.87e-09 ***
## trainData$RoamMins          0.069105   0.024446   2.827   0.0047 **
## ---
## 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: 1534.1  on 2325  degrees of freedom
## AIC: 1550.1
##
## Number of Fisher Scoring iterations: 5
```

## Performing Likelihood Ratio Test

```
#Likelihood Ratio

lrtest(logModel2)

## Likelihood ratio test
##
## Model 1: trainData$Churn ~ trainData$ContractRenewal + trainData$DataPlan
+
##     trainData$CustServCalls + trainData$DayMins + trainData$DayCalls +
##     trainData$OverageFee + trainData$RoamMins
## Model 2: trainData$Churn ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   8 -767.05
## 2   1 -965.21 -7 396.31  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After performing the Likelihood Ratio test on the Model, we receive the P Value as 2.2 e-16 which is extremely smaller than the alpha, hence the model is significant.

## Calculating McFadden Psudo R-sq

```
# Calculating Psudo R Sq Value

pR2(logModel)

##          llh      llhNull           G2     McFadden         r2ML
## -766.8221376 -965.2094900  396.7747047    0.2055381    0.1563947
##         r2CU
##    0.2778705
```

The value for McFadden Psudo R Sq is 0.205, which is in the Range of 0.2 – 0.4 making the Model significant Model.

**Calculating Odds Ratio for Logistic Model**

```
#Odds Ratio

exp(logModel2$coefficients)

##                (Intercept) trainData$ContractRenewal1
##                0.002003324                0.139887679
##        trainData$DataPlan1    trainData$CustServCalls
##                0.434111570                1.738147802
##          trainData$DayMins          trainData$DayCalls
##                1.012285612                1.005513014
##       trainData$OverageFee          trainData$RoamMins
##                1.175255283                1.071548942
```

This represents that with One-unit change in the Independent variable what will be the change in Dependent Variable. In our Model Cust Service Call explains the maximum variation. With one-unit change in Cust Service Calls there will be a significant change of 73% in Churn.

### 3.1.2 **Predicting Logistic Model on Train**

Going forward, we will be performing the Prediction on Train Data to check the accuracy of the Model built.

```
# Predicting the Model on Train Set.
log_pred_train <- predict(logModel2,type = "response",data = trainData)

table_pred_train <- table(trainData$Churn, log_pred_train>.5)
table_pred_train

##
##     FALSE TRUE
##   0  1946   49
##   1   273   65

pred_train_num <- ifelse(log_pred_train >.5,1,0)
pred_train_num <- as.factor(pred_train_num)
actual_train_num <- trainData$Churn
```

## Performing Confusion Matrix on the Predicted Values

```
confusionMatrix(pred_train_num,actual_train_num)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1946  273
##          1   49   65
##
##                Accuracy : 0.862
##                  95% CI : (0.8473, 0.8757)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.1813
##
##                   Kappa : 0.2314
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9754
##             Specificity : 0.1923
##          Pos Pred Value : 0.8770
##          Neg Pred Value : 0.5702
##              Prevalence : 0.8551
##          Detection Rate : 0.8341
##    Detection Prevalence : 0.9511
##       Balanced Accuracy : 0.5839
##
##        'Positive' Class : 0
##
```

As per the above Confusion Matrix, we have achieved the accuracy of 86.2% with a Sensitivity of 97.5% and Specificity of 19.2%.

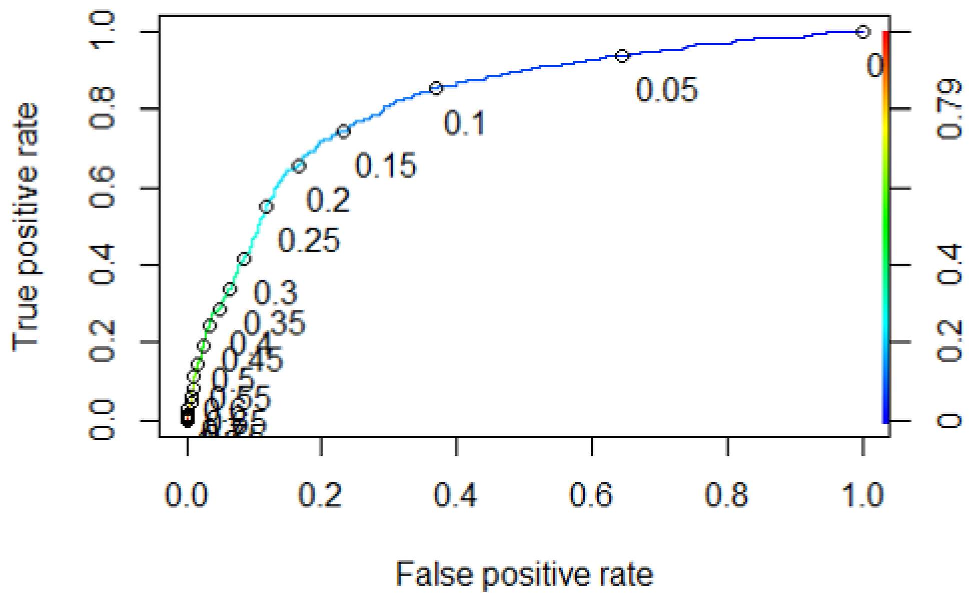
## Calculating the Base Line Accuracy on Train Data

```
# Baseline Accuracy
nrow(trainData[trainData$Churn ==0,])/nrow(trainData)

## [1] 0.8551222
```

The base line model explains the error rate that can occur if all the observations are predicted False. In case of Train Data, we achieved the Base Line Accuracy of 85.5%.

## Plotting ROC Curve

```
# Plotting ROC Curve

perfObj_train <- prediction(log_pred_train,trainData$Churn)
perf_train <- performance(perfObj_train,"tpr","fpr")
plot(perf_train, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05),text.adj =
c(-0.2,1.7))
```

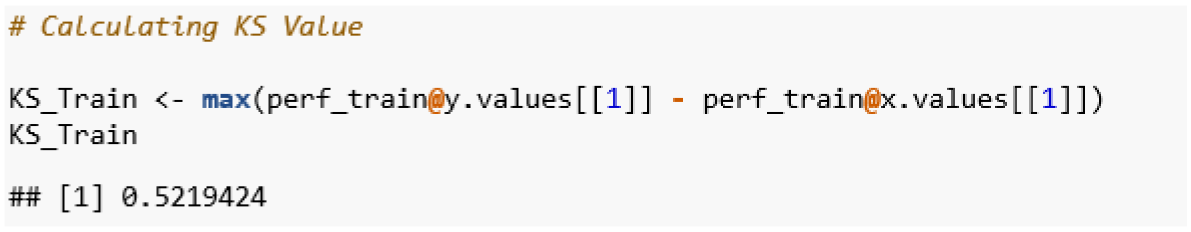


## Calculating the AUC Value

```
# Calculating AUC Value

auc <- performance(perfObj_train,"auc")
auc@y.values

## [[1]]
## [1] 0.8180318
```

Calculated AUC Value for Train data is 81.8%

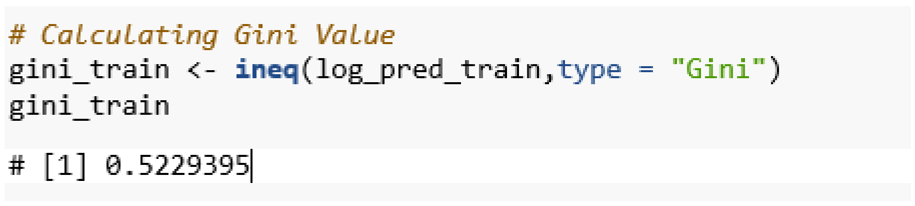
## Calculating the KS Value



```
# Calculating KS Value

KS_Train <- max(perf_train@y.values[[1]] - perf_train@x.values[[1]])
KS_Train

## [1] 0.5219424
```

Calculated KS Value for Train Data is 52.19%

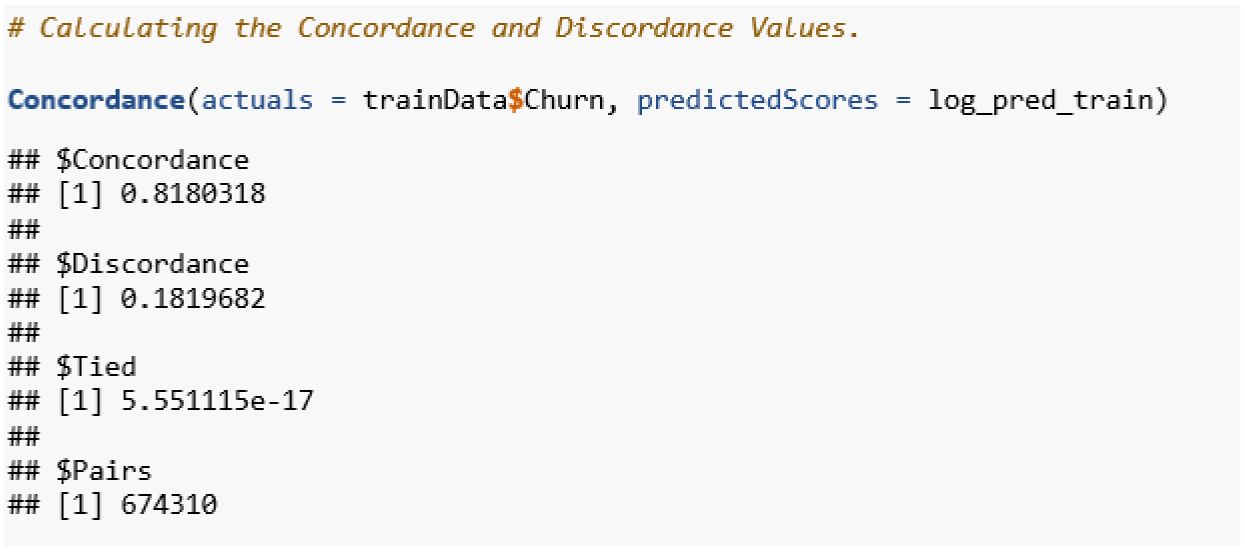
## Calculating the Gini Value



```
# Calculating Gini Value
gini_train <- ineq(log_pred_train,type = "Gini")
gini_train

# [1] 0.5229395
```

Calculated Gini Value for Train Data is 52.2%

## Calculating the Concordance and Discordance Values



```
# Calculating the Concordance and Discordance Values.

Concordance(actuals = trainData$Churn, predictedScores = log_pred_train)

## $Concordance
## [1] 0.8180318
##
## $Discordance
## [1] 0.1819682
##
## $Tied
## [1] 5.551115e-17
##
## $Pairs
## [1] 674310
```

### 3.1.3 **Predicting Logistic Model on Test**

Now we will be predicting the values on the unseen dataset i.e. Test Data.

```
# Predicting values on Test Dataset.


log_pred_test <- predict(logModel, newdata = testData, type = "response")


pred_test_num <- ifelse(log_pred_test >.5,1,0)
pred_test_num <- as.factor(pred_test_num)
str(pred_test_num)

##  Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 1 1 1 ...
##  - attr(*, "names")= chr [1:1000] "1" "2" "3" "4" ...

actual_test_num <- (testData$Churn)
```

**Performing Confusion Matrix on the Predicted Values**

```
confusionMatrix(pred_test_num,actual_test_num)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 826 116
##          1  29  29
##
##                Accuracy : 0.855
##                  95% CI : (0.8316, 0.8763)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.5221
##
##                   Kappa : 0.2212
##
##  Mcnemar's Test P-Value : 9.204e-13
##
##             Sensitivity : 0.9661
##             Specificity : 0.2000
##          Pos Pred Value : 0.8769
##          Neg Pred Value : 0.5000
##              Prevalence : 0.8550
##          Detection Rate : 0.8260
##    Detection Prevalence : 0.9420
##       Balanced Accuracy : 0.5830
##
##        'Positive' Class : 0
##
```

As per the above Confusion Matrix, we have achieved the accuracy of 85.5% with a Sensitivity of 96.6% and Specificity of 20.0%.

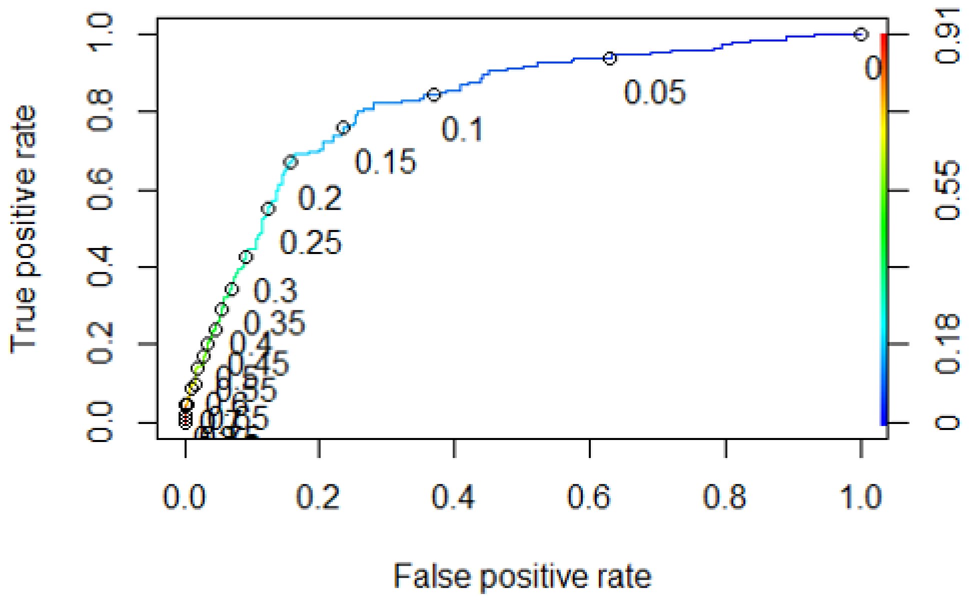
## Calculating the Base Line Accuracy on Train Data

```
# Baseline Accuracy
nrow(testData[testData$Churn ==0,])/nrow(testData)

## [1] 0.855
```

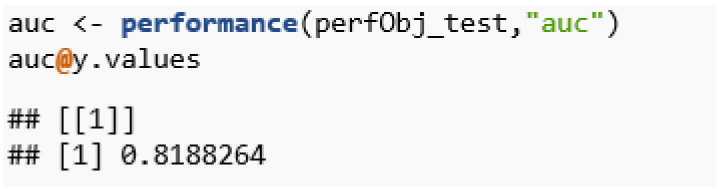
The base line model explains the error rate that can occur if all the observations are predicted False. In case of Test Data, we achieved the Base Line Accuracy of 85.5%.

## Plotting ROC Curve

```
# Plotting ROC Curve
perfObj_test <- prediction(log_pred_test,testData$Churn)
perf_test <- performance(perfObj_test,"tpr","fpr")


plot(perf_test, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05),text.adj =
c(-0.2,1.7))
```



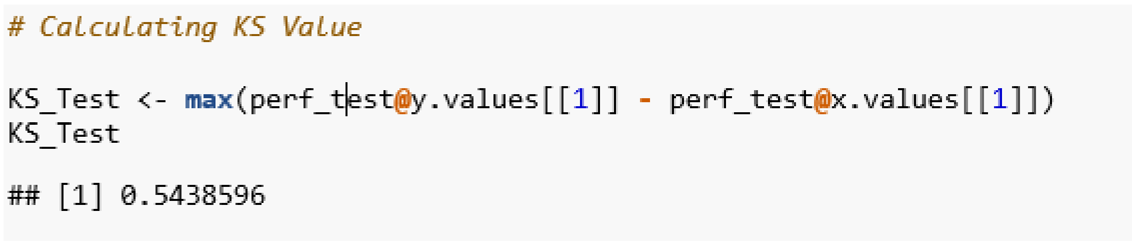
## Calculating the AUC Value



```
auc <- performance(perfObj_test,"auc")
auc@y.values

## [[1]]
## [1] 0.8188264
```

Calculated AUC Value for Test data is 81.8%

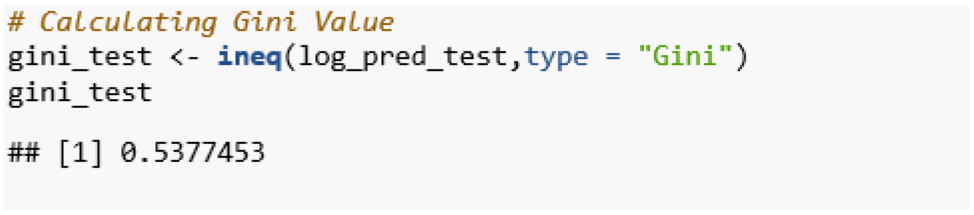
## Calculating the KS Value



```
# Calculating KS Value

KS_Test <- max(perf_test@y.values[[1]] - perf_test@x.values[[1]])
KS_Test

## [1] 0.5438596
```

Calculated KS Value for Test Data is 54.38%

## Calculating the Gini Value



```
# Calculating Gini Value
gini_test <- ineq(log_pred_test,type = "Gini")
gini_test

## [1] 0.5377453
```

Calculated Gini Value for Test Data is 52.2%

**Calculating the Concordance and Discordance Values**

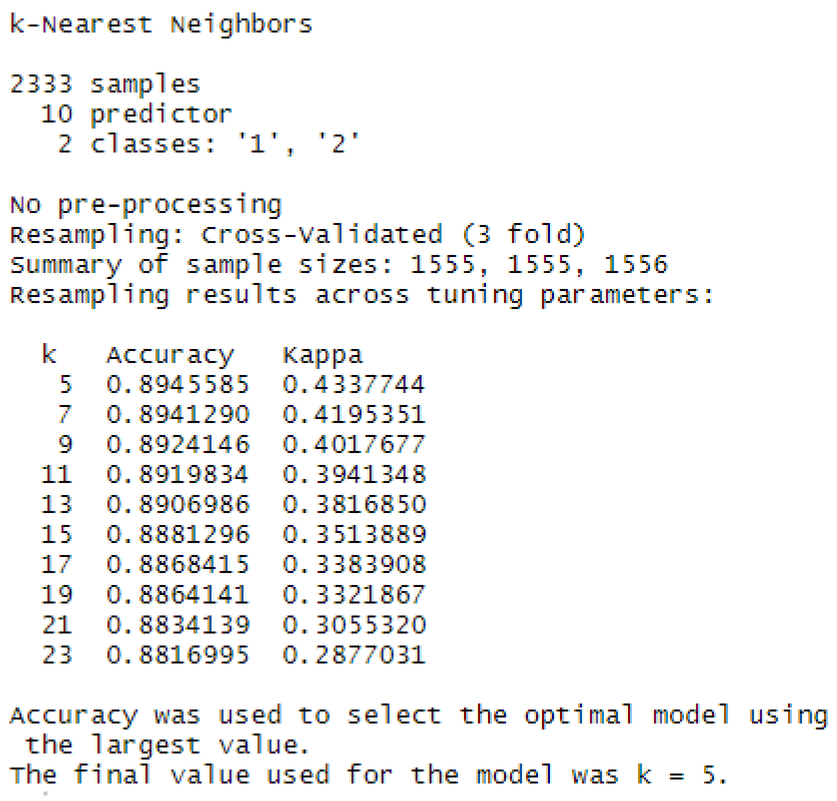
```
# Calculating the Concordance and Discordance Values.

Concordance(actuals = testData$Churn, predictedScores = log_pred_test)

## $Concordance
## [1] 0.8188264
##
## $Discordance
## [1] 0.1811736
##
## $Tied
## [1] 0
##
## $Pairs
## [1] 123975
```

## 3.2 K-Nearest Neighbor Model

k-nearest neighbor classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

### 3.2.1 **Building KNN Model on Train**

```r
# Performing KNN Model


trainData_scale <- scale(trainData[,-c(1,3,4)])
trainData_scale1 <- cbind(trainData[,c(1,3,4)], trainData_scale)

testData_scale <- scale(testData[,-c(1,3,4)])
testData_scale1 <- cbind(testData[,c(1,3,4)], testData_scale)
trctrl <- trainControl(method = "cv", number = 3)
knn_fit = train(Churn ~., data = trainData_scale1, method = "knn",
                trControl = trctrl,
                tuneLength = 10)

knn_fit$bestTune$k

## [1] 5
```



```
k-Nearest Neighbors

2333 samples
  10 predictor
   2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 1555, 1555, 1556
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.8945585  0.4337744
   7  0.8941290  0.4195351
   9  0.8924146  0.4017677
  11  0.8919834  0.3941348
  13  0.8906986  0.3816850
  15  0.8881296  0.3513889
  17  0.8868415  0.3383908
  19  0.8864141  0.3321867
  21  0.8834139  0.3055320
  23  0.8816995  0.2877031

Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was k = 5.
```

We achieve the maximum Accuracy at k = 5.

Now when we have built the KNN Model on Train data we will further be predicting the Model on Train data and build the Confusion Matrix to check for the Accuracy of the Model.

### 3.2.2 Performing Prediction and Creating Confusion Matrix on Train Data

```
#Performance Matrix on Train data

pred_knn_train <- predict(knn_fit, data = trainData_scale1, type = "raw")
confusionMatrix(pred_knn_train, trainData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1973  174
##          1   22  164
##
##                Accuracy : 0.916
##                  95% CI : (0.904, 0.9269)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5831
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9890
##             Specificity : 0.4852
##          Pos Pred Value : 0.9190
##          Neg Pred Value : 0.8817
##              Prevalence : 0.8551
##          Detection Rate : 0.8457
##    Detection Prevalence : 0.9203
##       Balanced Accuracy : 0.7371
##
##        'Positive' Class : 0
##
```

From the above we've gained the Accuracy of 91.6% and Sensitivity of 98.9% and Specificity of 48.5%.

### 3.2.3 **Performing Prediction and Creating Confusion Matrix on Test Data**

```
#Performance Matrix on Test data

pred_knn_test <- predict(knn_fit, newdata = testData_scale1, type = "raw")
confusionMatrix(pred_knn_test, testData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 842  96
##          1  13  49
##
##                Accuracy : 0.891
##                  95% CI : (0.87, 0.9096)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.0004856
##
##                   Kappa : 0.4233
##
##  Mcnemar's Test P-Value : 4.024e-15
##
##             Sensitivity : 0.9848
##             Specificity : 0.3379
##          Pos Pred Value : 0.8977
##          Neg Pred Value : 0.7903
##              Prevalence : 0.8550
##          Detection Rate : 0.8420
##    Detection Prevalence : 0.9380
##       Balanced Accuracy : 0.6614
##
##        'Positive' Class : 0
##
```

From the prediction on Test Data, we achieve the accuracy of 89.1% with a Sensitivity of 98.4% and Sensitivity of 33.7%.

## 3.3 Naïve Bayes Model

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem.

### 3.3.1 <u>**Building the Naïve Bayes Model on Train Data**</u>

```r
# Performing Naive Bayes

NB <- naiveBayes(trainData_scale1$Churn~., data = trainData_scale1)
summary(NB)

##           Length Class  Mode
## apriori    2     table  numeric
## tables    10     -none- list
## levels     2     -none- character
## isnumeric 10     -none- logical
## call       4     -none- call
```

### 3.3.2 Predicting the Model and Creating Performance Matrix on Train Data

```r
#Performance Matrix on Train Data

pred_nb_train <- predict(NB, newdata = trainData_scale1)
confusionMatrix(pred_nb_train, trainData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1933  238
##          1   62  100
##
##                Accuracy : 0.8714
##                  95% CI : (0.8571, 0.8847)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.01273
##
##                   Kappa : 0.3378
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9689
##             Specificity : 0.2959
##          Pos Pred Value : 0.8904
##          Neg Pred Value : 0.6173
##              Prevalence : 0.8551
##          Detection Rate : 0.8285
##    Detection Prevalence : 0.9306
##       Balanced Accuracy : 0.6324
##
##        'Positive' Class : 0
##
```

From the above Model, we achieve the Accuracy of 87.14% with a Sensitivity of 96.8% and Specificity of 29.5%.

### 3.3.3 Predicting Model and Performing Confusion Matrix on Test Data

Now we will be predicting the values on the unseen data after performing on Train data.

```
#Performance Matrix on Test Data

pred_nb_test <- predict(NB, newdata = testData_scale1)
confusionMatrix(pred_nb_test, testData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 818 101
##          1  37  44
##
##                Accuracy : 0.862
##                  95% CI : (0.8391, 0.8828)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.2821
##
##                   Kappa : 0.3186
##
##  Mcnemar's Test P-Value : 8.189e-08
##
##             Sensitivity : 0.9567
##             Specificity : 0.3034
##          Pos Pred Value : 0.8901
##          Neg Pred Value : 0.5432
##              Prevalence : 0.8550
##          Detection Rate : 0.8180
##    Detection Prevalence : 0.9190
##       Balanced Accuracy : 0.6301
##
##        'Positive' Class : 0
##
```

From the above Prediction, we achieve the Accuracy of 86.2% with a Sensitivity of 95.6% and Specificity of 30.3%.

## 3.4 Summary

Now, when we have built the Logistic Model, KNN Model, and Naïve Bayes Model.

Below is the Summary of all the Models and their Performance measures in a matrix format.

| | Logistic Model | | KNN | | Naïve Bayes | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |

| | | | | | | |
|---|---|---|---|---|---|---|
| Accuracy | 86.2 | 85.5 | 91.6 | 89.1 | 87.4 | 86.2 |
| Sensitivity | 97.5 | 96.6 | 98.9 | 98.4 | 96.8 | 95.6 |
| Specificity | 19.2 | 20.0 | 48.5 | 33.7 | 29.5 | 30.3 |
| AUC | 81.8 | 81.8 | - | - | - | - |
| Gini | 52.2 | 52.2 | - | - | - | - |
| KS | 52.19 | 54.3 | - | - | - | - |
| Concordance | 81.9 | 81.9 | - | - | - | - |
| Discordance | 18.1 | 18.1 | - | - | - | - |

Considering the Confusion Matrix created on all the three Models, KNN achieves the highest Accuracy of 91.6% in Train and 89.1% in Test but with a significantly high error rate of 48.5% and 33.7% respectively. Whereas, Logistic Model performed well with the least error rate of approximately 20% in both Train and Test.

Also, with the increase in the Sample size the error rate in KNN and Naïve Bayes will go high.

Since Logistic Regression is a Probability Based Algorithm, the threshold can be altered as per the business requirement giving more precise results.

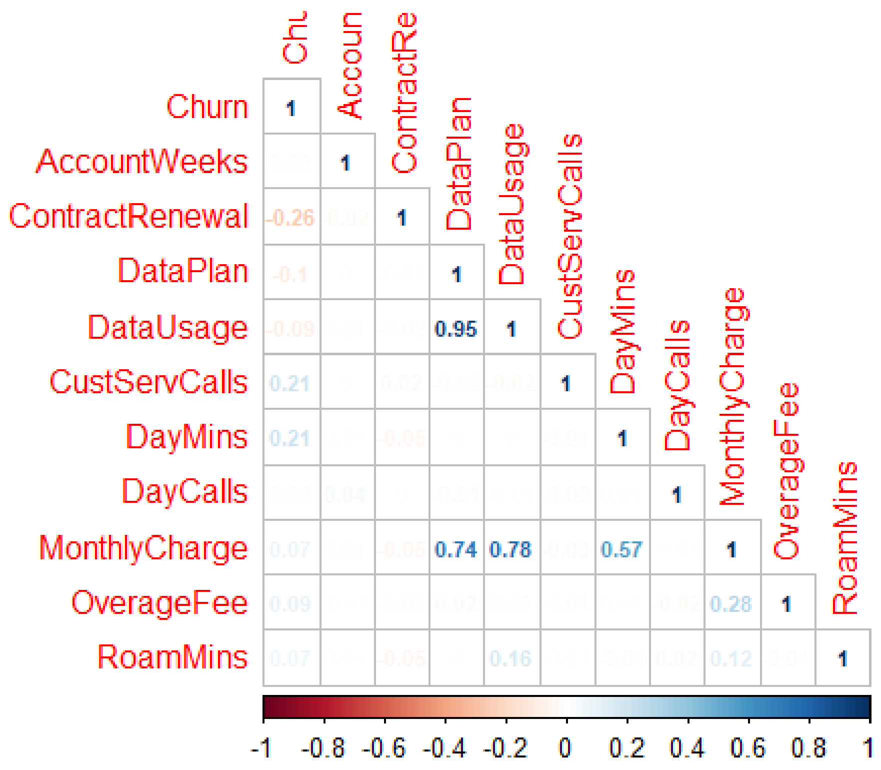
**At the end, the most Recommended would be the Logistic Model to achieve more precise and accurate predictions.**

## 4 Appendix A – Source Code

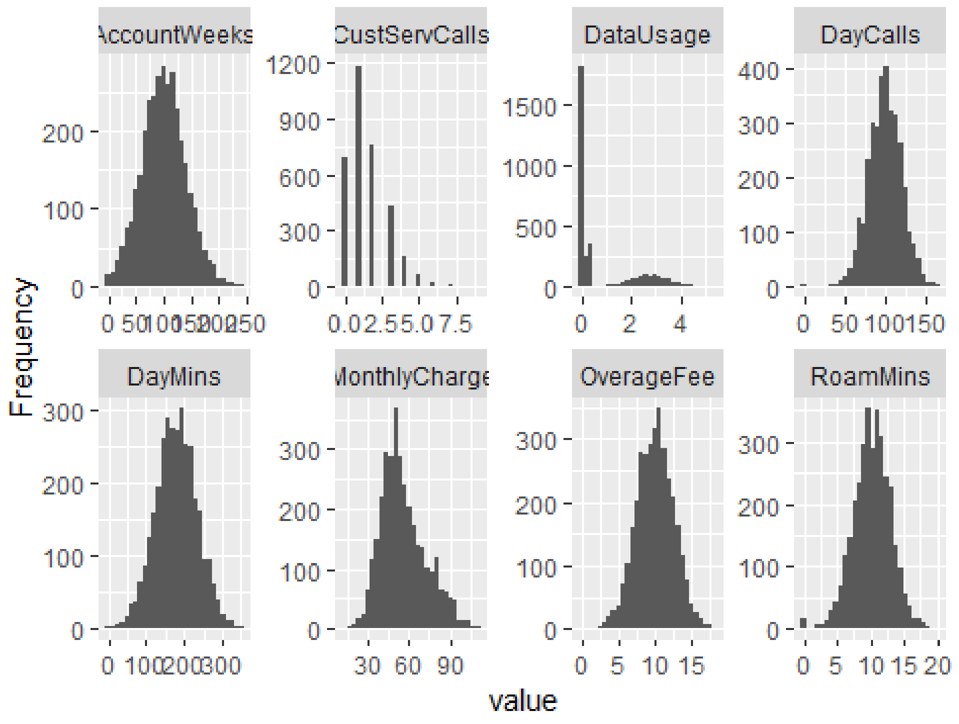
```
##### Project 4 ########

# Deploying necessary Libraries
library(readxl)
library(DataExplorer)
library(corrplot)

## corrplot 0.84 loaded

library(dplyr)

##
## Attaching package: 'dplyr'

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

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

library(InformationValue)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'caret'

## The following objects are masked from 'package:InformationValue':
##
##     confusionMatrix, precision, sensitivity, specificity

library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

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

library(caTools)
library(blorr)

## Warning: package 'blorr' was built under R version 3.6.2

library(MASS)

##
## Attaching package: 'MASS'

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

library(class)
library(e1071)
library(car)

## Loading required package: carData

##
## Attaching package: 'car'

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

library(lmtest)

## Warning: package 'lmtest' was built under R version 3.6.2

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.2

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

library(pscl)

## Warning: package 'pscl' was built under R version 3.6.2

## 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(ineq)

### Setting up Working Directory

setwd("D:/Great Learning/Predictive Modeling/Project 4")

### Reading the Dataset from Excel

cellphone <- read_xlsx("Cellphone.xlsx",sheet = 2)

### Performing Exploratory Data Analysis

head(cellphone)

## # A tibble: 6 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
## # ... with 5 more variables: DayMins <dbl>, DayCalls <dbl>,
## #   MonthlyCharge <dbl>, OverageFee <dbl>, RoamMins <dbl>

str(cellphone)

## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11 variables:
##  $ Churn          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
##  $ ContractRenewal: num  1 1 1 0 0 0 1 0 1 0 ...
##  $ DataPlan       : num  1 1 0 0 0 0 1 0 0 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 0 2.03 0 0.19 3.02 ...
##  $ CustServCalls  : num  1 1 0 2 3 0 3 0 1 0 ...
##  $ DayMins        : num  265 162 243 299 167 ...
##  $ DayCalls       : num  110 123 114 71 113 98 88 79 97 84 ...
##  $ MonthlyCharge  : num  89 82 52 57 41 57 87.3 36 63.9 93.2 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
```

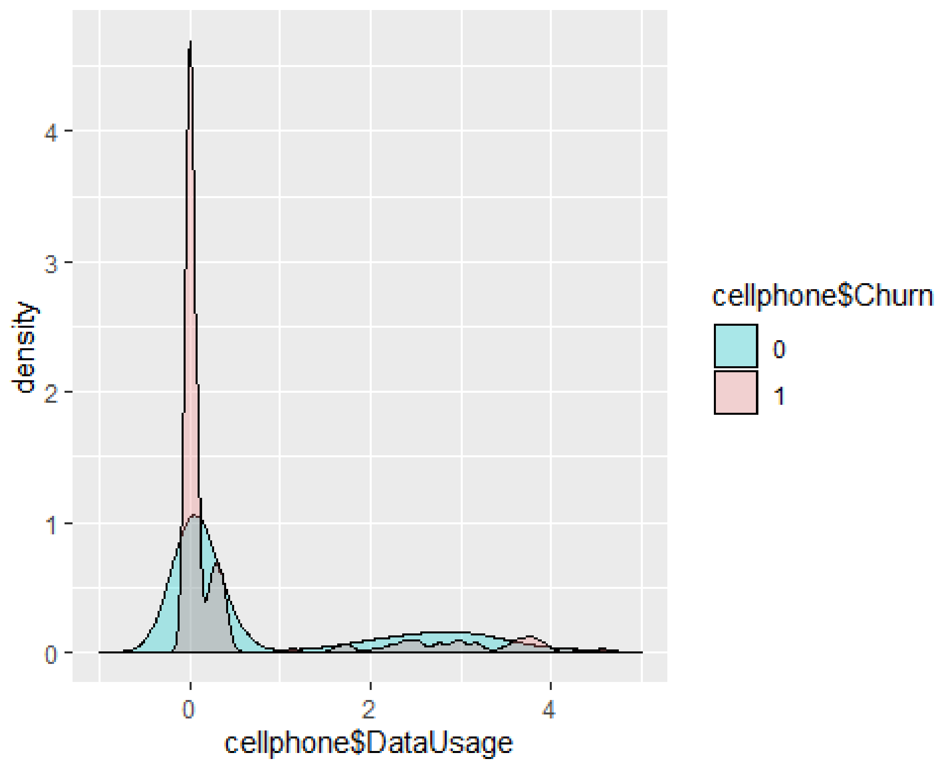
```
summary(cellphone)

##      Churn          AccountWeeks    ContractRenewal     DataPlan
##  Min.   :0.0000   Min.   :  1.0   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.: 74.0   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.0000   Median :101.0   Median :1.0000   Median :0.0000
##  Mean   :0.1449   Mean   :101.1   Mean   :0.9031   Mean   :0.2766
##  3rd Qu.:0.0000   3rd Qu.:127.0   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :243.0   Max.   :1.0000   Max.   :1.0000
##    DataUsage       CustServCalls      DayMins          DayCalls
##  Min.   :0.0000   Min.   :0.000   Min.   :  0.0   Min.   :  0.0
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:143.7   1st Qu.: 87.0
##  Median :0.0000   Median :1.000   Median :179.4   Median :101.0
##  Mean   :0.8165   Mean   :1.563   Mean   :179.8   Mean   :100.4
##  3rd Qu.:1.7800   3rd Qu.:2.000   3rd Qu.:216.4   3rd Qu.:114.0
##  Max.   :5.4000   Max.   :9.000   Max.   :350.8   Max.   :165.0
##  MonthlyCharge      OverageFee       RoamMins
##  Min.   : 14.00   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 45.00   1st Qu.: 8.33   1st Qu.: 8.50
##  Median : 53.50   Median :10.07   Median :10.30
##  Mean   : 56.31   Mean   :10.05   Mean   :10.24
##  3rd Qu.: 66.20   3rd Qu.:11.77   3rd Qu.:12.10
##  Max.   :111.30   Max.   :18.19   Max.   :20.00

# Checking for any Null Values in the data

sum(is.na(cellphone))

## [1] 0

# Checking for Multicollinearity

mat<- cor(cellphone)
corrplot(mat,method = "number",type = "lower", number.cex = 0.7)
```



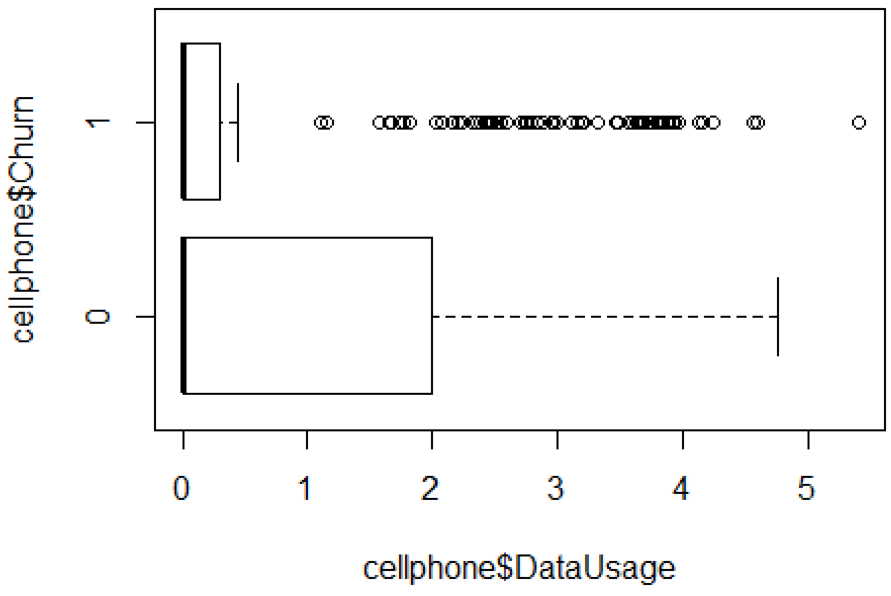
```
# Converting Binary variables to Factors

cellphone$Churn <- as.factor(cellphone$Churn)
cellphone$ContractRenewal <- as.factor(cellphone$ContractRenewal)
cellphone$DataPlan <- as.factor(cellphone$DataPlan)

str(cellphone)

## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 1 ...
##  $ DataPlan       : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 1 2 ...
##  $ 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 ...

# Performing Univariate Analysis

plot_histogram(cellphone)
```



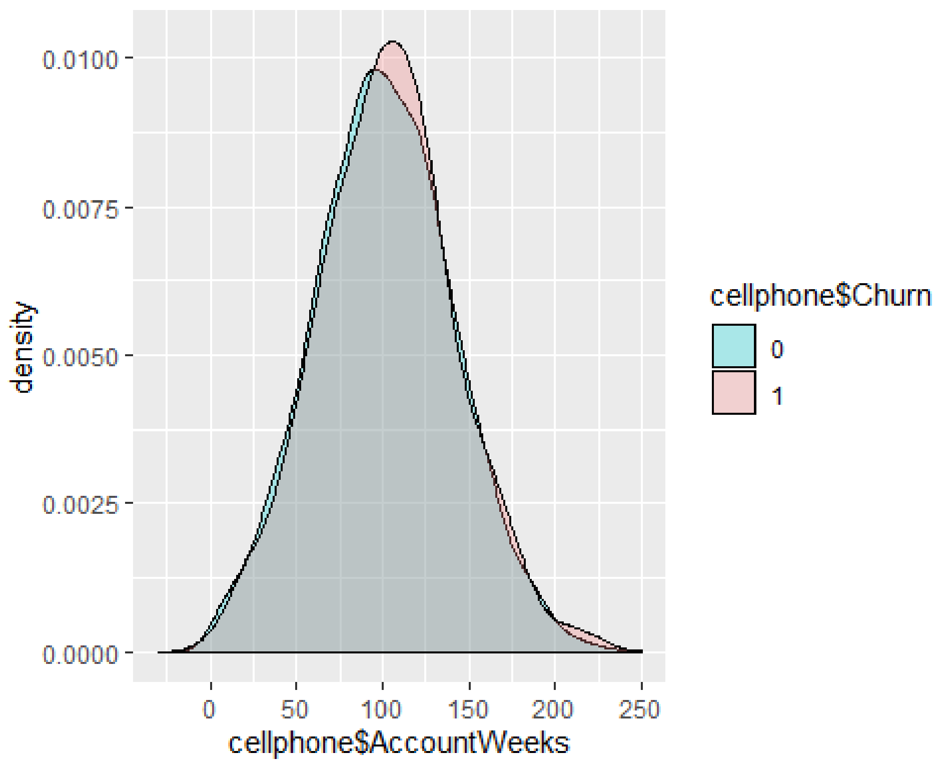
```
# Performing Bivariate Analysis on Continuous Variables


ggplot(cellphone, aes(x = cellphone$DataUsage)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_color_manual(values = c("#868686FF", "#EFC000FF")) +
  scale_fill_manual(values = c("darkturquoise", "lightcoral")) + xlim(-1,5)

## Warning: Removed 1 rows containing non-finite values (stat_density).
```



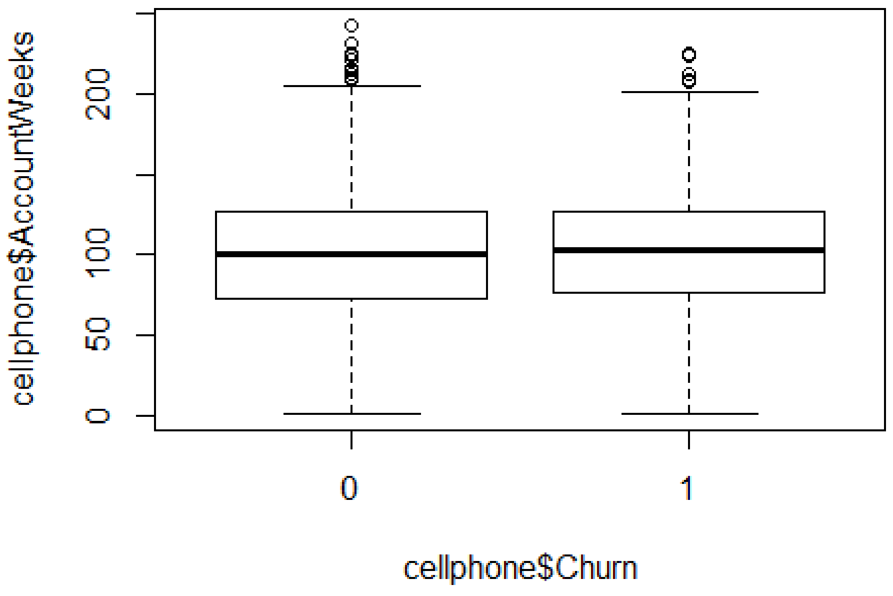
```
boxplot(cellphone$DataUsage~cellphone$Churn,horizontal = TRUE)
```



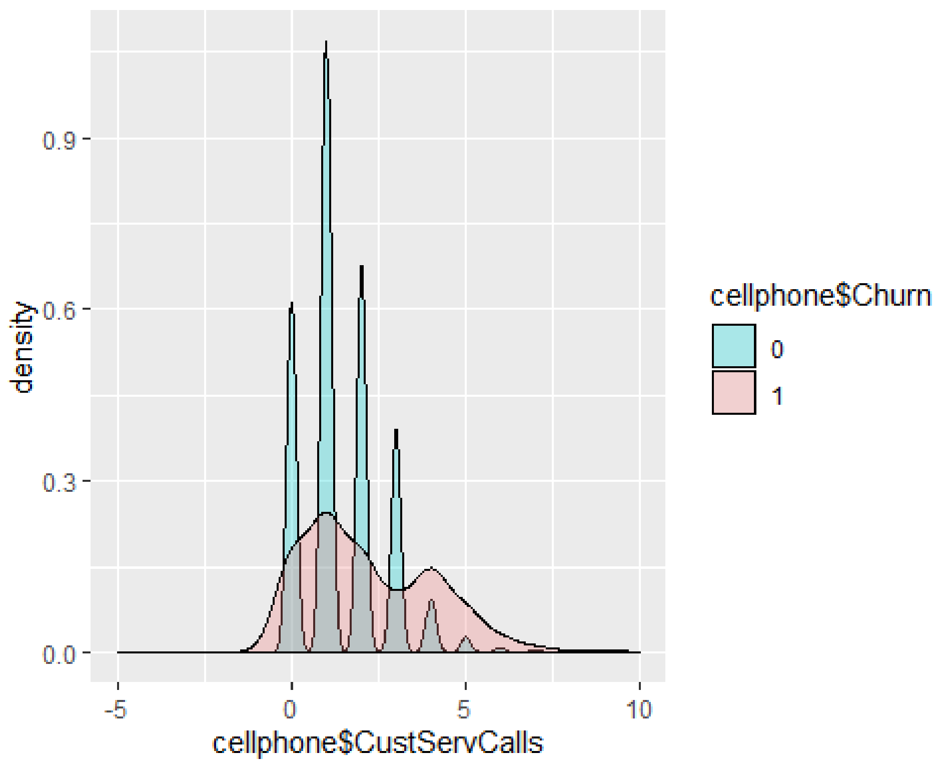
```
ggplot(cellphone, aes(x = cellphone$AccountWeeks)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) + xlim(-30,250)
```



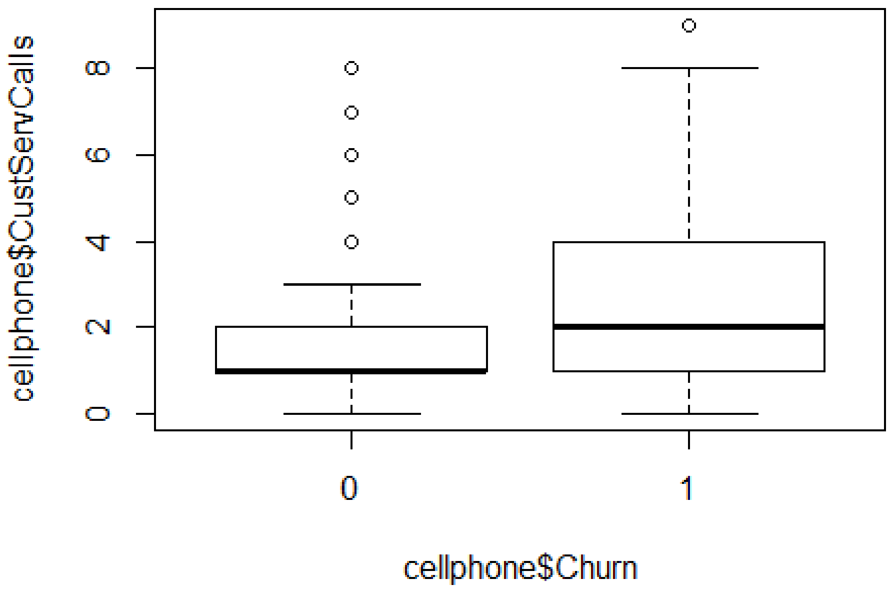
```
boxplot(cellphone$AccountWeeks~cellphone$Churn)
```



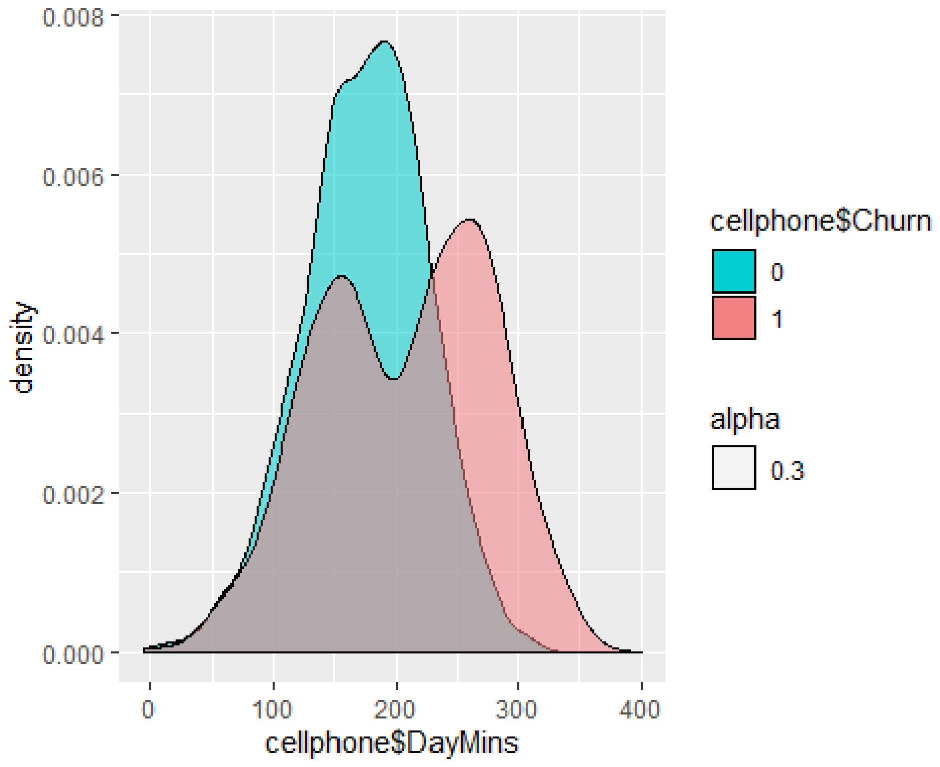
```
ggplot(cellphone, aes(x = cellphone$CustServCalls)) +
  geom_density(aes(fill = cellphone$Churn), alpha = 0.3) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) + xlim(-5,10)
```



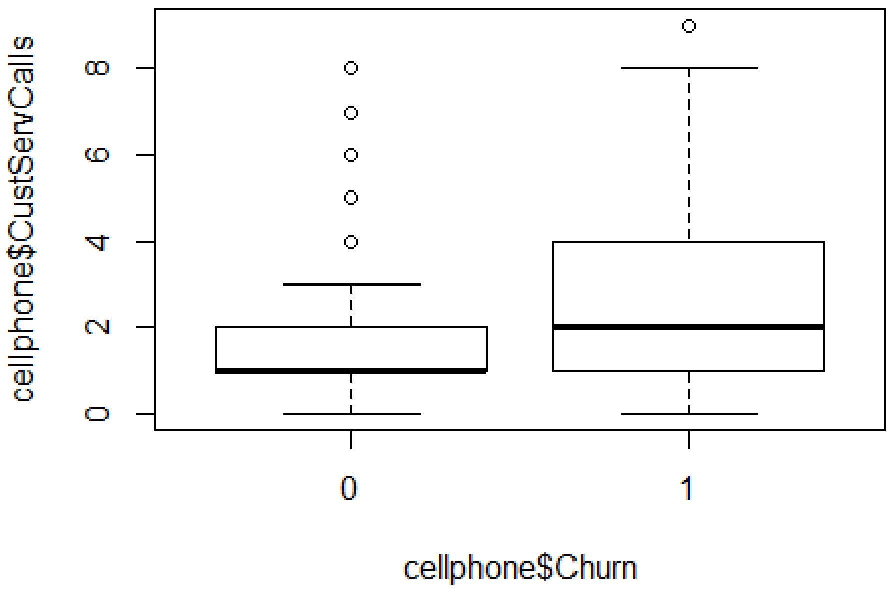
```
boxplot(cellphone$CustServCalls~cellphone$Churn)
```



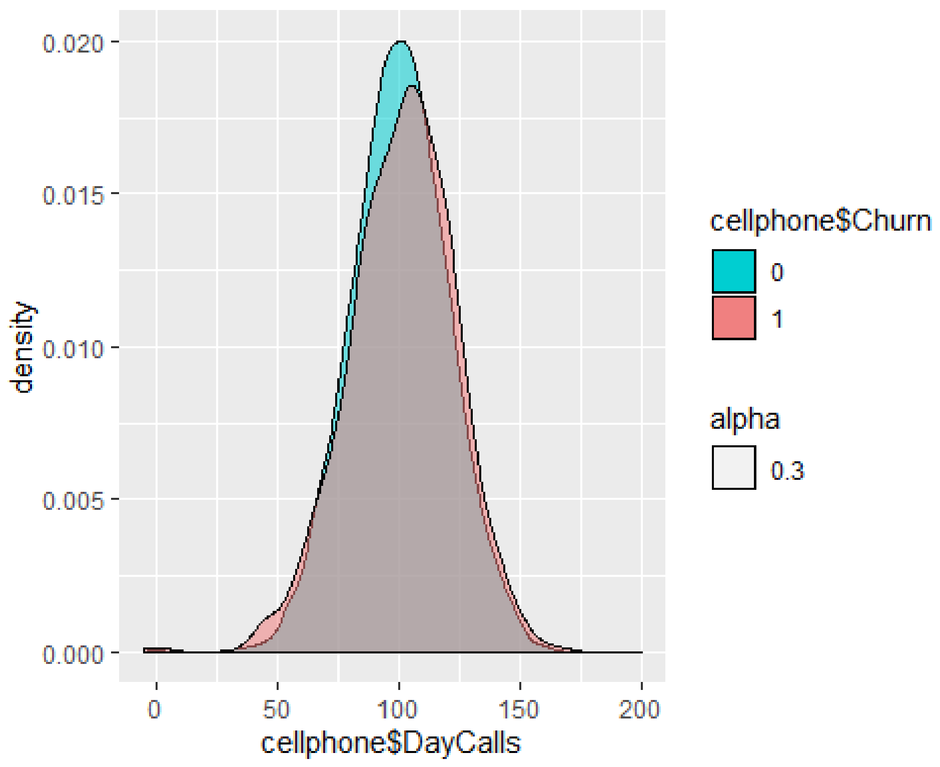
```
ggplot(cellphone,aes(x = cellphone$DayMins)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,400)
```



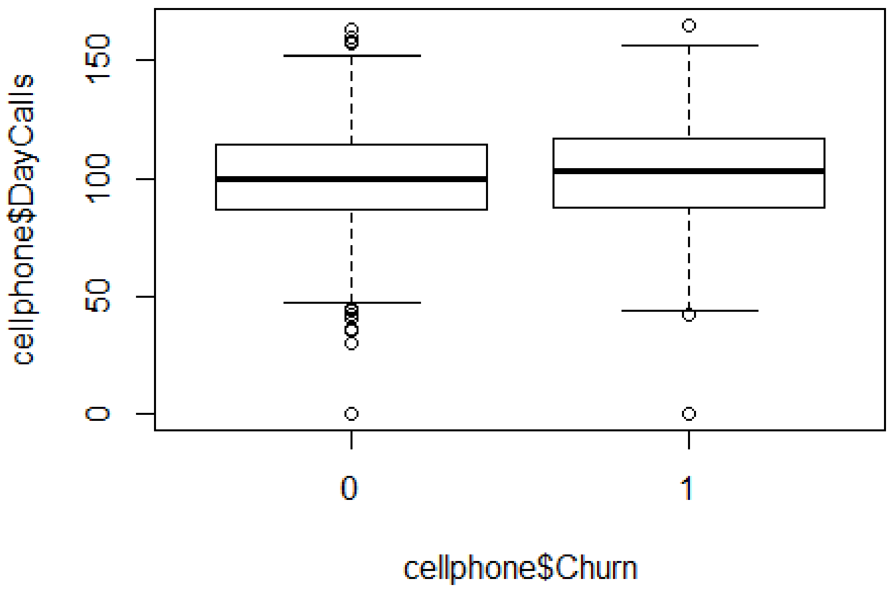
```
boxplot(cellphone$CustServCalls~cellphone$Churn)
```



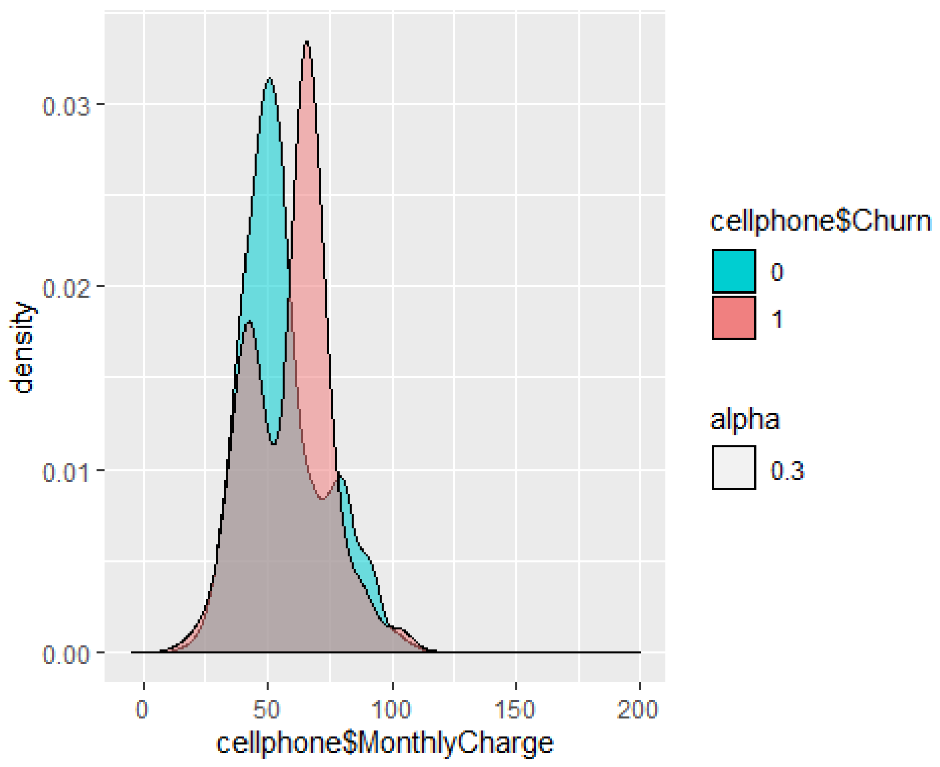
```
ggplot(cellphone,aes(x = cellphone$DayCalls)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,200)
```



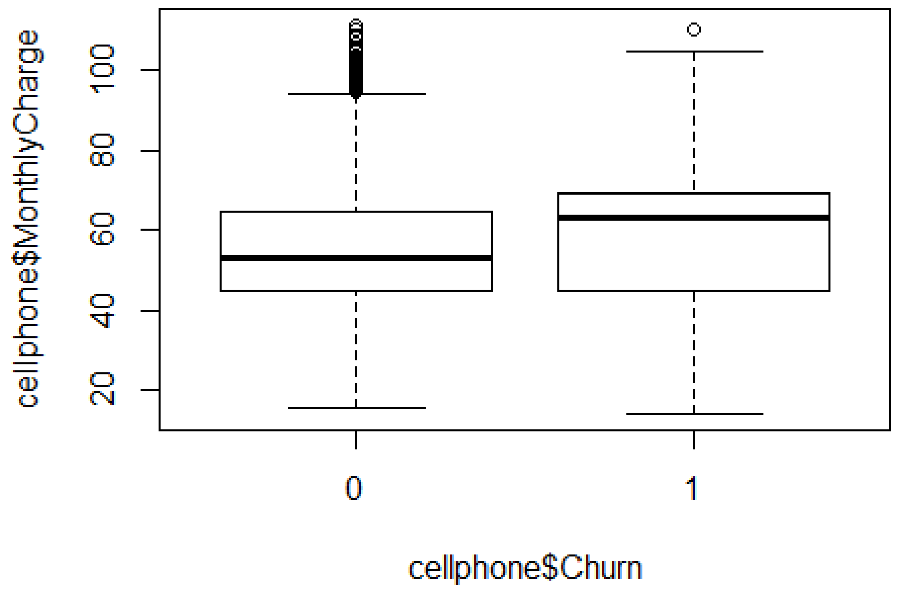
```
boxplot(cellphone$DayCalls~cellphone$Churn)
```



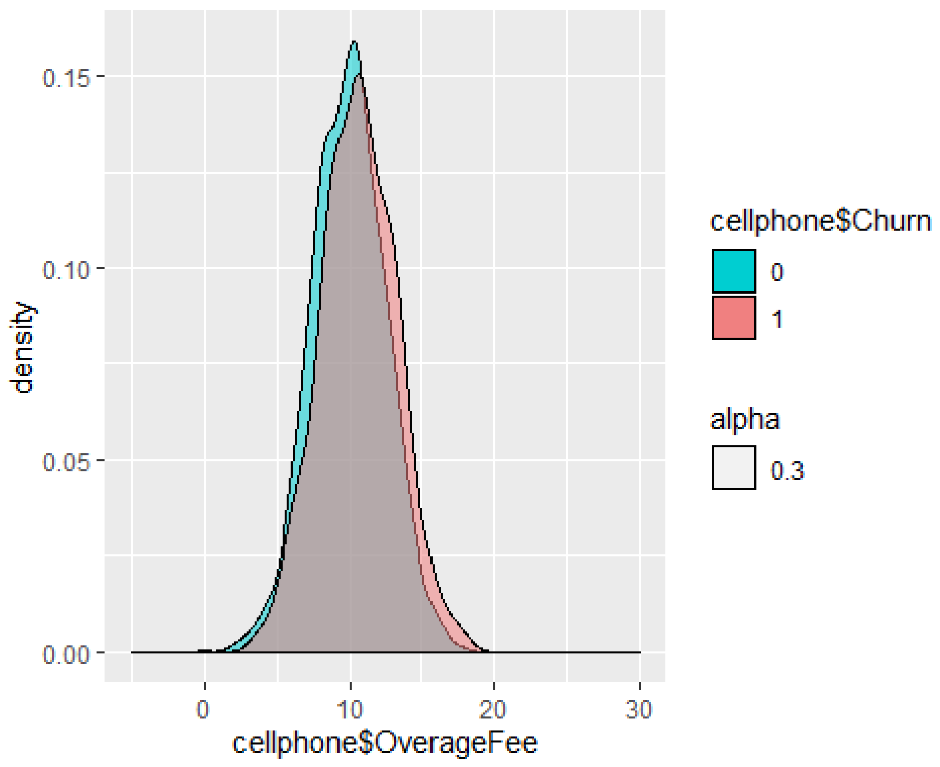
```
ggplot(cellphone,aes(x = cellphone$MonthlyCharge)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,200)
```



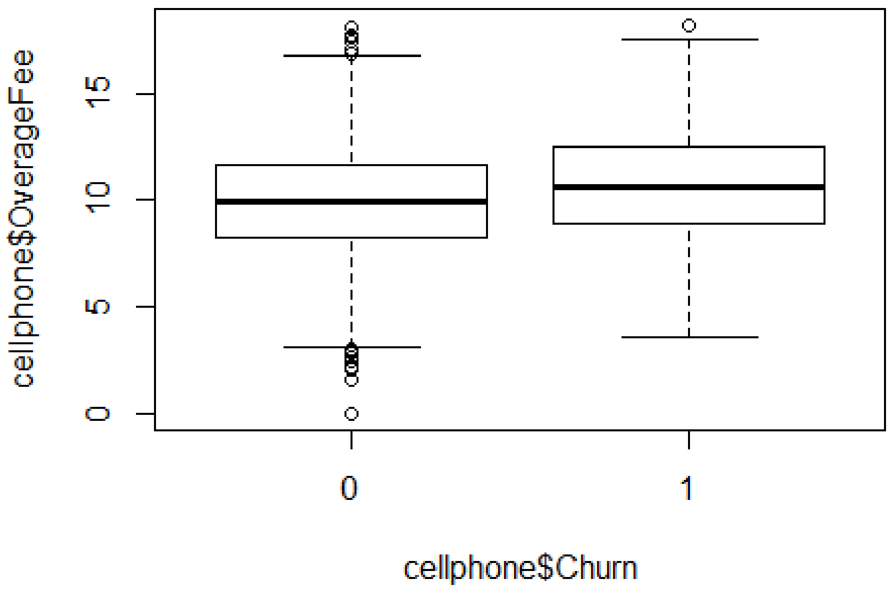
```
boxplot(cellphone$MonthlyCharge~cellphone$Churn)
```



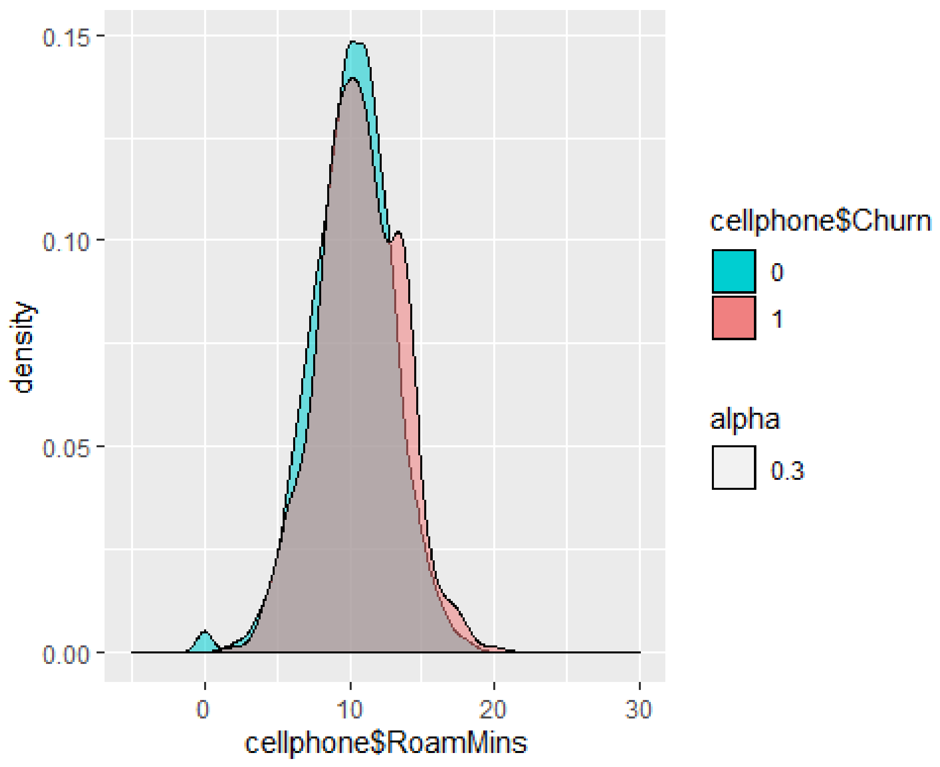
```
ggplot(cellphone,aes(x = cellphone$OverageFee)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,30)
```



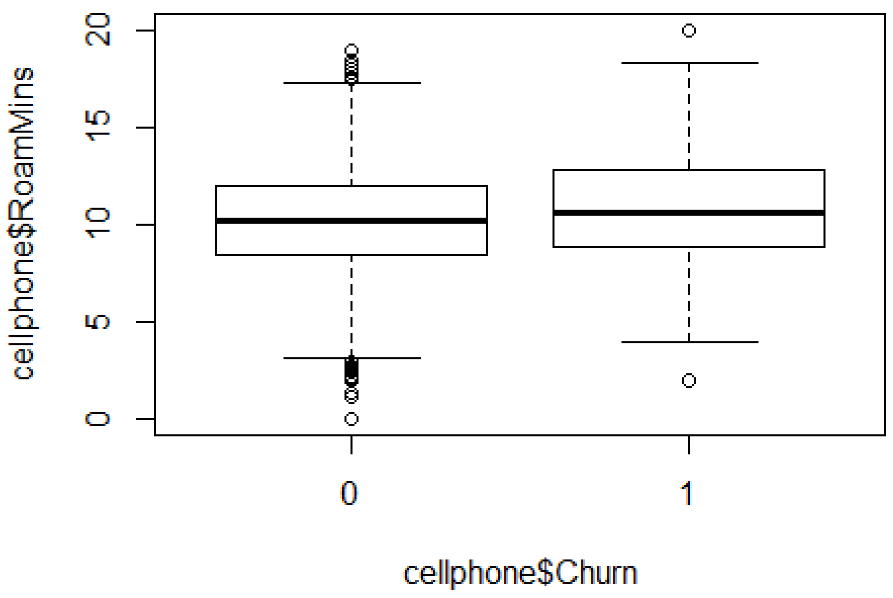
```
boxplot(cellphone$OverageFee~cellphone$Churn)
```



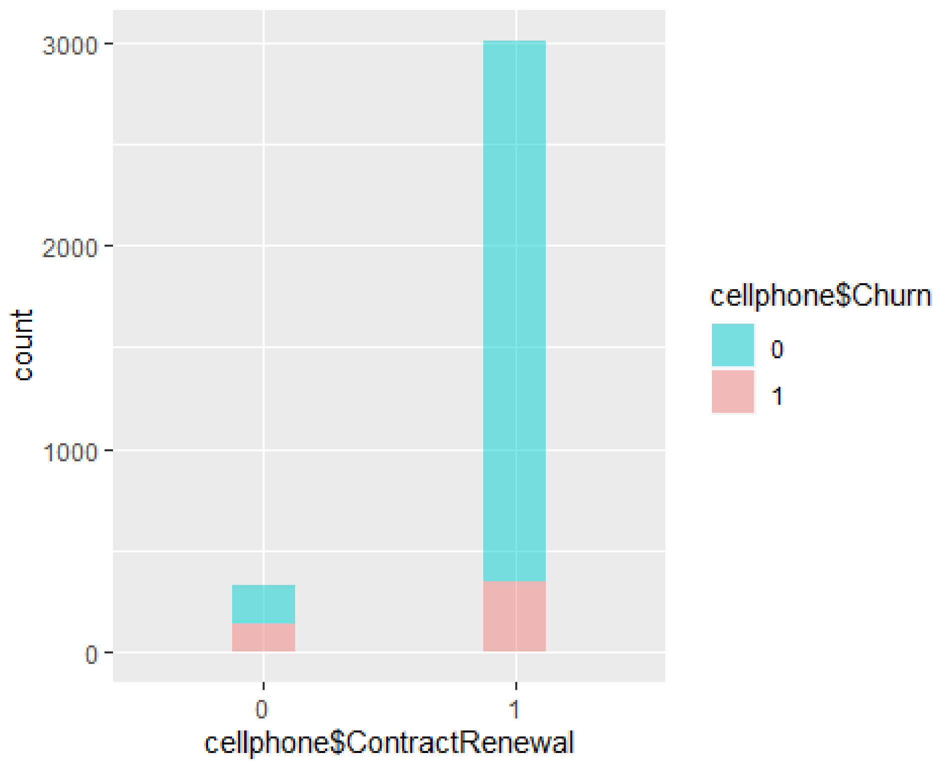
```
ggplot(cellphone,aes(x = cellphone$RoamMins)) +
  geom_density(aes(fill = cellphone$Churn,alpha = 0.3)) +
  scale_fill_manual(values = c("darkturquoise","lightcoral")) +xlim(-5,30)
```



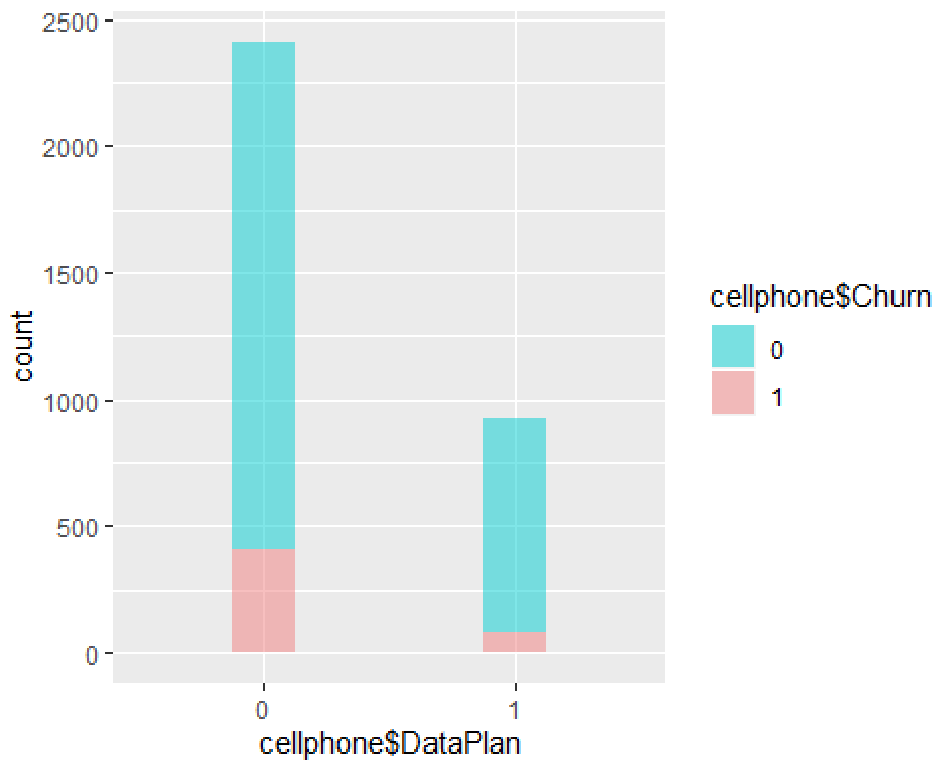
```
boxplot(cellphone$RoamMins~cellphone$Churn)
```



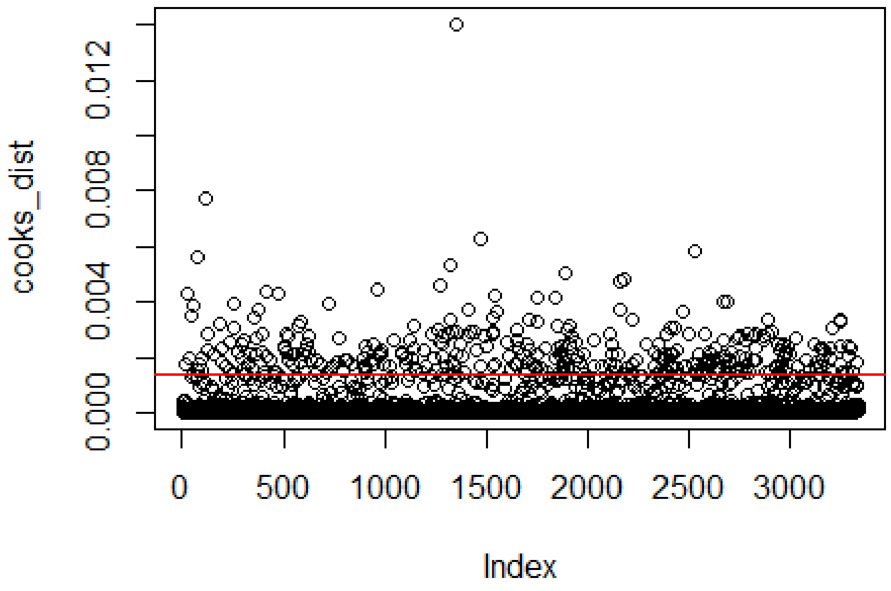
```
# Performing Bi-Variate analysis on Categorical Variables

ggplot(cellphone,aes(x = cellphone$ContractRenewal, fill = cellphone$Churn)) +
  geom_bar(width = 0.25,alpha = 0.5) +
  scale_fill_manual(values = c("darkturquoise","lightcoral"))
```



```r
ggplot(cellphone, aes(x = cellphone$DataPlan, fill = cellphone$Churn)) +
  geom_bar(width = 0.25, alpha=0.5) +
  scale_fill_manual(values = c('darkturquoise', 'lightcoral'))
```



```
str(cellphone)

## Classes 'tbl_df', 'tbl' and 'data.frame':    3333 obs. of  11 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AccountWeeks   : num  128 107 137 84 75 118 121 147 117 141 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 1 ...
##  $ DataPlan       : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 2 1 1 2 ...
##  $ 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 ...

# Performing Cooks Distance to check the Outliers.
cellphone$Churn<- as.numeric(cellphone$Churn)
cook_lm <- lm(cellphone$Churn~. -ContractRenewal -DataPlan, data = cellphone,)

cooks_dist<- cooks.distance(cook_lm)

plot(cooks_dist)

abline(h=4*mean(cooks_dist,na.rm = TRUE),col="red")
```



```
influential <- as.numeric(names(cooks_dist)[(cooks_dist > 4*mean(cooks_dist, na.rm=T))])

out <- (cellphone[influential, ])
dim(out)

## [1] 341  11
```

```
View(out)

cellphone$Churn<- as.factor(cellphone$Churn)
```

```
#Splitting the Data to Test and Train.

set.seed(1234)

index<- sample.split(cellphone$Churn,SplitRatio = .70)

trainData <- subset(cellphone, index == TRUE)
testData <- subset(cellphone,index == FALSE)

str(trainData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    2333 obs. of  11 variables:
##  $ Churn          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ AccountWeeks   : num  128 107 137 84 118 121 147 117 141 65 ...
##  $ ContractRenewal: Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 2 1 2 ...
##  $ DataPlan       : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 1 1 2 1 ...
##  $ DataUsage      : num  2.7 3.7 0 0 0 2.03 0 0.19 3.02 0.29 ...
##  $ CustServCalls  : num  1 1 0 2 0 3 0 1 0 4 ...
##  $ DayMins        : num  265 162 243 299 223 ...
##  $ DayCalls       : num  110 123 114 71 98 88 79 97 84 137 ...
##  $ MonthlyCharge  : num  89 82 52 57 57 87.3 36 63.9 93.2 44.9 ...
##  $ OverageFee     : num  9.87 9.78 6.06 3.1 11.03 ...
##  $ RoamMins       : num  10 13.7 12.2 6.6 6.3 7.5 7.1 8.7 11.2 12.7 ...

# Creating the Logistic Regression Model on Train data.

logModel <- glm(trainData$Churn~., data = trainData,family = "binomial")

summary(logModel)

##
## Call:
## glm(formula = trainData$Churn ~ ., family = "binomial", data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0417  -0.5097  -0.3504  -0.2088   3.0459
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -6.1567684  0.6674321  -9.225  < 2e-16 ***
## AccountWeeks     -0.0001507  0.0016498  -0.091  0.92722
## ContractRenewal1 -1.9669347  0.1762725 -11.158  < 2e-16 ***
## DataPlan1        -0.6793909  0.6385085  -1.064  0.28732
## DataUsage         1.3441189  2.3006497   0.584  0.55906
## CustServCalls     0.5529267  0.0468420  11.804  < 2e-16 ***
## DayMins           0.0360104  0.0387869   0.928  0.35319
## DayCalls          0.0054809  0.0032975   1.662  0.09648 .
## MonthlyCharge    -0.1399872  0.2280180  -0.614  0.53926
## OverageFee        0.4001170  0.3896952   1.027  0.30454
## RoamMins          0.0715722  0.0265797   2.693  0.00709 **
## ---
## 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: 1533.6  on 2322  degrees of freedom
## AIC: 1555.6
##
## Number of Fisher Scoring iterations: 5
```

```
logModel$coefficients
```

```
##      (Intercept)     AccountWeeks ContractRenewal1        DataPlan1
##    -6.1567683649    -0.0001506931    -1.9669347188    -0.6793909335
##        DataUsage    CustServCalls          DayMins         DayCalls
##     1.3441188926     0.5529266860     0.0360104406     0.0054808964
##    MonthlyCharge       OverageFee         RoamMins
##    -0.1399871773     0.4001170448     0.0715721615
```

```
# Checking for Multicollinearity using VIF

vif(logModel)
```

```
##    AccountWeeks ContractRenewal        DataPlan       DataUsage
##        1.002719        1.060203       14.134368     1581.374480
##   CustServCalls         DayMins        DayCalls   MonthlyCharge
##        1.096619      938.800975        1.005125     2719.647020
##      OverageFee        RoamMins
##      207.886910        1.197360
```

```
# Model refining - Logistic Model

# Performing Step Wise AIC on Logistic Model
blr_step_aic_both(logModel)
```

```
## Stepwise Selection Method
## -------------------------
##
## Candidate Terms:
##
## 1 . AccountWeeks
## 2 . ContractRenewal
## 3 . DataPlan
## 4 . DataUsage
## 5 . CustServCalls
## 6 . DayMins
## 7 . DayCalls
## 8 . MonthlyCharge
## 9 . OverageFee
## 10 . RoamMins
##
##
## Variables Entered/Removed:
##
```

```
## - CustServCalls added
## - ContractRenewal added
## - DayMins added
## - OverageFee added
## - DataPlan added
## - RoamMins added
## - DayCalls added
##
## No more variables to be added or removed.

##
##
##                          Stepwise Summary
## -----------------------------------------------------------------
## Variable             Method       AIC          BIC        Deviance
## -----------------------------------------------------------------
## CustServCalls      addition    1821.620    1833.130    1817.620
## ContractRenewal    addition    1700.531    1717.795    1694.531
## DayMins            addition    1609.925    1632.945    1601.925
## OverageFee         addition    1580.904    1609.678    1570.904
## DataPlan           addition    1557.154    1591.684    1545.154
## RoamMins           addition    1550.903    1591.187    1536.903
## DayCalls           addition    1550.107    1596.146    1534.107
## -----------------------------------------------------------------

logModel2 <- glm(trainData$Churn ~ trainData$ContractRenewal + trainData$DataPlan +train
Data$CustServCalls+trainData$DayMins+ trainData$DayCalls+ trainData$OverageFee+ trainDat
a$RoamMins,
                 data= trainData,
                 family = "binomial")
summary(logModel2)

##
## Call:
## glm(formula = trainData$Churn ~ trainData$ContractRenewal + trainData$DataPlan +
##     trainData$CustServCalls + trainData$DayMins + trainData$DayCalls +
##     trainData$OverageFee + trainData$RoamMins, family = "binomial",
##     data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0392  -0.5088  -0.3515  -0.2083   3.0190
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -6.212947   0.636348  -9.763  < 2e-16 ***
## trainData$ContractRenewal1 -1.966915   0.175849 -11.185  < 2e-16 ***
## trainData$DataPlan1       -0.834454   0.170899  -4.883 1.05e-06 ***
## trainData$CustServCalls     0.552820   0.046818  11.808  < 2e-16 ***
## trainData$DayMins           0.012211   0.001291   9.457  < 2e-16 ***
## trainData$DayCalls          0.005498   0.003294   1.669   0.0952 .
## trainData$OverageFee        0.161485   0.027597   5.851 4.87e-09 ***
## trainData$RoamMins          0.069105   0.024446   2.827   0.0047 **
## ---
## 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: 1534.1  on 2325  degrees of freedom
## AIC: 1550.1
##
## Number of Fisher Scoring iterations: 5
```

```
logModel2$coefficients
```

```
##                 (Intercept) trainData$ContractRenewal1
##                -6.212947493               -1.966915470
##          trainData$DataPlan1    trainData$CustServCalls
##                -0.834453703                0.552820065
##            trainData$DayMins          trainData$DayCalls
##                 0.012210756                0.005497873
##         trainData$OverageFee          trainData$RoamMins
##                 0.161485386                0.069105211
```

```
#Likelihood Ratio

lrtest(logModel2)
```

```
## Likelihood ratio test
##
## Model 1: trainData$Churn ~ trainData$ContractRenewal + trainData$DataPlan +
##     trainData$CustServCalls + trainData$DayMins + trainData$DayCalls +
##     trainData$OverageFee + trainData$RoamMins
## Model 2: trainData$Churn ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   8 -767.05
## 2   1 -965.21 -7 396.31  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

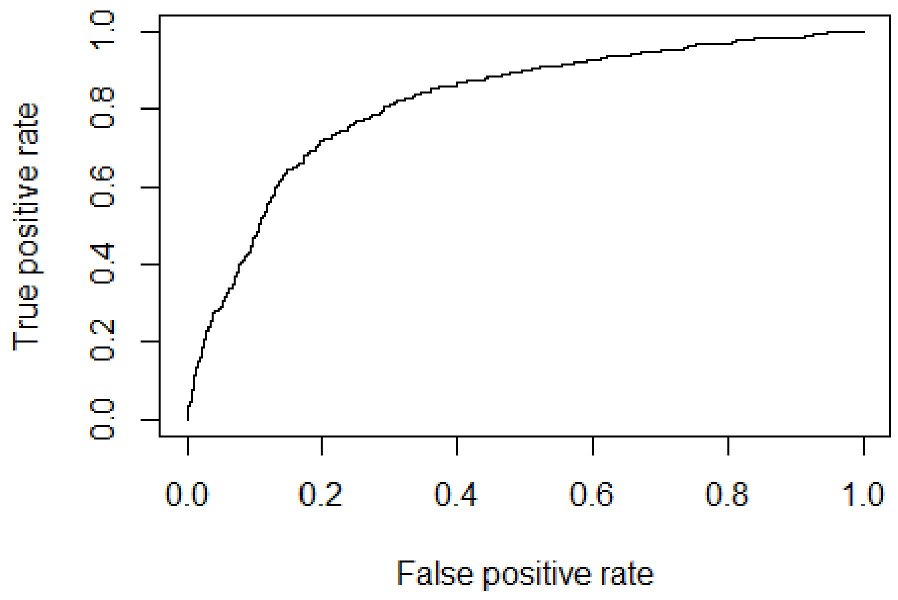
```
# Calculating Psudo R Sq Value

pR2(logModel)
```

```
##          llh       llhNull            G2     McFadden         r2ML
## -766.8221376 -965.2094900  396.7747047    0.2055381    0.1563947
##         r2CU
##    0.2778705
```

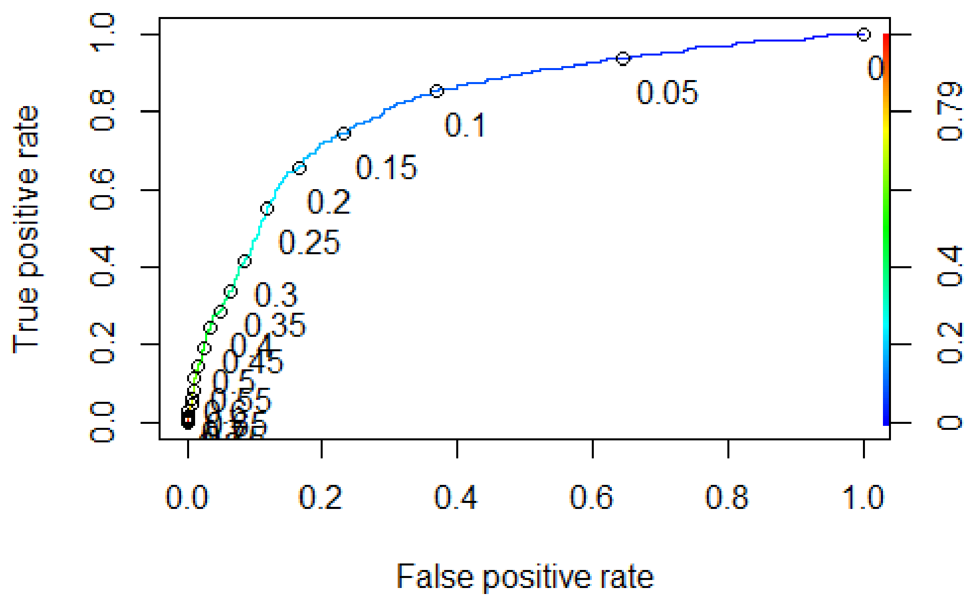
```
#Odds Ratio

exp(logModel2$coefficients)
```

```
##                 (Intercept) trainData$ContractRenewal1
##                 0.002003324                0.139887679
##          trainData$DataPlan1    trainData$CustServCalls
##                 0.434111570                1.738147802
##            trainData$DayMins          trainData$DayCalls
##                 1.012285612                1.005513014
##       trainData$OverageFee         trainData$RoamMins
##                1.175255283                1.071548942

# Predicting the Model on Train Set.
log_pred_train <- predict(logModel2,type = "response",data = trainData)

table_pred_train <- table(trainData$Churn, log_pred_train>.5)
table_pred_train

##
##     FALSE TRUE
##   0  1946   49
##   1   273   65

pred_train_num <- ifelse(log_pred_train >.5,1,0)
pred_train_num <- as.factor(pred_train_num)
class(pred_train_num)

## [1] "factor"

actual_train_num <- trainData$Churn
class(actual_train_num)

## [1] "factor"

confusionMatrix(pred_train_num,actual_train_num)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1946  273
##          1   49   65
##
##                Accuracy : 0.862
##                  95% CI : (0.8473, 0.8757)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.1813
##
##                   Kappa : 0.2314
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9754
##             Specificity : 0.1923
##          Pos Pred Value : 0.8770
##          Neg Pred Value : 0.5702
##              Prevalence : 0.8551
##          Detection Rate : 0.8341
##    Detection Prevalence : 0.9511
##       Balanced Accuracy : 0.5839
##
##        'Positive' Class : 0
##
```

```r
# Baseline Accuracy
nrow(trainData[trainData$Churn ==0,])/nrow(trainData)

## [1] 0.8551222

# Plotting ROC Curve

perfObj_train <- prediction(log_pred_train,trainData$Churn)
perf_train <- performance(perfObj_train,"tpr","fpr")
plot(perf_train)
```



```r
plot(perf_train, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05),text.adj = c(-0.2,1.7
))
```



```
auc <- performance(perfObj_train,"auc")
auc@y.values

## [[1]]
## [1] 0.8180318

# Calculating KS Value

KS_Train <- max(perf_train@y.values[[1]] - perf_train@x.values[[1]])

# Calculating Gini Value
gini_train <- ineq(log_pred_train,type = "Gini")

# Calculating the Concordance and Discordance Values.

Concordance(actuals = trainData$Churn, predictedScores = log_pred_train)

## $Concordance
## [1] 0.8180318
##
## $Discordance
## [1] 0.1819682
##
## $Tied
## [1] 5.551115e-17
##
## $Pairs
## [1] 674310

# Predicting values on Test Dataset.
names(testData)
```

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

```
log_pred_test <- predict(logModel, newdata = testData, type = "response")


pred_test_num <- ifelse(log_pred_test >.5,1,0)
pred_test_num <- as.factor(pred_test_num)
str(pred_test_num)
```

```
##  Factor w/ 2 levels "0","1": 2 1 1 2 1 1 1 1 1 1 ...
##  - attr(*, "names")= chr [1:1000] "1" "2" "3" "4" ...
```

```
actual_test_num <- (testData$Churn)
confusionMatrix(pred_test_num,actual_test_num)
```

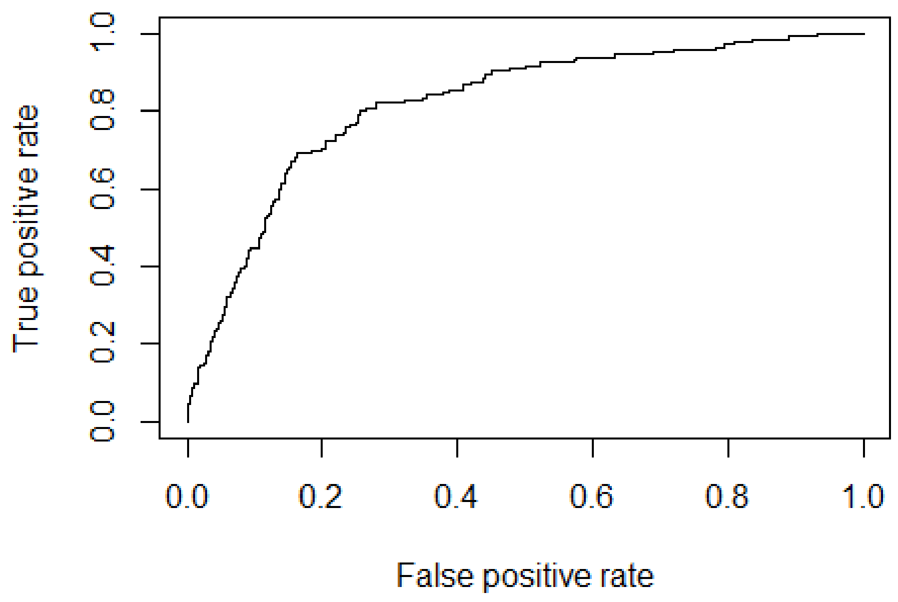
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 826 116
##          1  29  29
##
##                Accuracy : 0.855
##                  95% CI : (0.8316, 0.8763)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.5221
##
##                   Kappa : 0.2212
##
##  Mcnemar's Test P-Value : 9.204e-13
##
##             Sensitivity : 0.9661
##             Specificity : 0.2000
##          Pos Pred Value : 0.8769
##          Neg Pred Value : 0.5000
##              Prevalence : 0.8550
##          Detection Rate : 0.8260
##    Detection Prevalence : 0.9420
##       Balanced Accuracy : 0.5830
##
##        'Positive' Class : 0
##
```

```
View(pred_test_num)

# Baseline Accuracy
nrow(testData[testData$Churn ==0,])/nrow(testData)
```

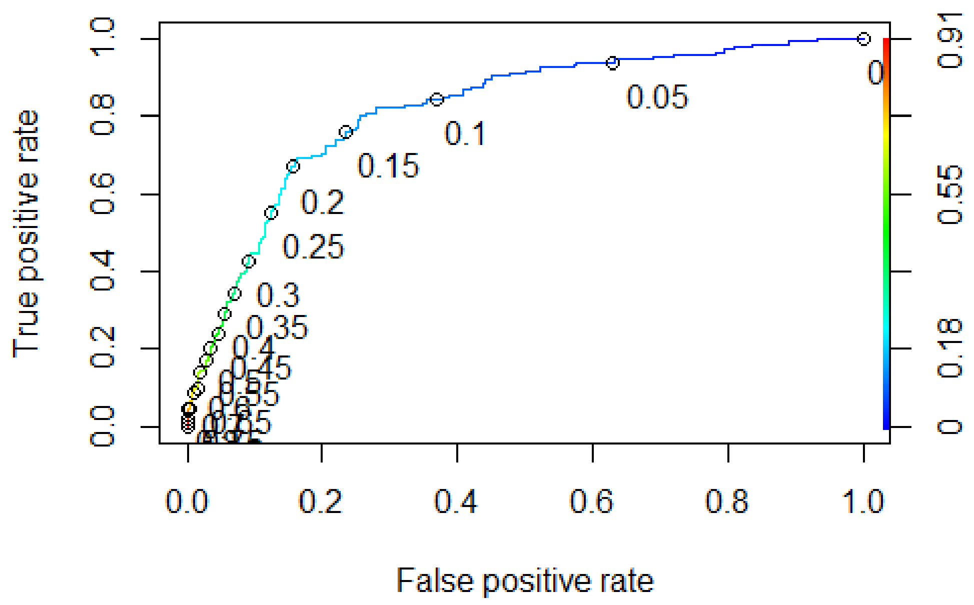
```
## [1] 0.855
```

```
# Plotting ROC Curve
length(log_pred_test)
```

```
## [1] 1000

length(testData$Churn)

## [1] 1000

perfObj_test <- prediction(log_pred_test,testData$Churn)
perf_test <- performance(perfObj_test,"tpr","fpr")
plot(perf_test)
```



```
plot(perf_test, colorize = TRUE, print.cutoffs.at = seq(0,1,0.05),text.adj = c(-0.2,1.7)
)
```



```
auc <- performance(perfObj_test,"auc")
auc@y.values

## [[1]]
## [1] 0.8188264

# Calculating KS Value

KS_Test <- max(perf_test@y.values[[1]] - perf_test@x.values[[1]])

# Calculating Gini Value
gini_test <- ineq(log_pred_test,type = "Gini")

# Calculating the Concordance and Discordance Values.

Concordance(actuals = testData$Churn, predictedScores = log_pred_test)

## $Concordance
## [1] 0.8188264
##
## $Discordance
## [1] 0.1811736
##
## $Tied
## [1] 0
##
## $Pairs
## [1] 123975
```

```
# Performing KNN Model


trainData_scale <- scale(trainData[,-c(1,3,4)])
trainData_scale1 <- cbind(trainData[,c(1,3,4)], trainData_scale)

testData_scale <- scale(testData[,-c(1,3,4)])
testData_scale1 <- cbind(testData[,c(1,3,4)], testData_scale)
trctrl <- trainControl(method = "cv", number = 3)
knn_fit = train(Churn ~., data = trainData_scale1, method = "knn",
                trControl = trctrl,
                tuneLength = 10)

knn_fit$bestTune$k

## [1] 5

#Performance Matrix on Train data

pred_knn_train <- predict(knn_fit, data = trainData_scale1, type = "raw")
confusionMatrix(pred_knn_train, trainData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1973  174
##          1   22  164
##
##                Accuracy : 0.916
##                  95% CI : (0.904, 0.9269)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5831
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9890
##             Specificity : 0.4852
##          Pos Pred Value : 0.9190
##          Neg Pred Value : 0.8817
##              Prevalence : 0.8551
##          Detection Rate : 0.8457
##    Detection Prevalence : 0.9203
##       Balanced Accuracy : 0.7371
##
##        'Positive' Class : 0
##

#Performance Matrix on Test data

pred_knn_test <- predict(knn_fit, newdata = testData_scale1, type = "raw")
confusionMatrix(pred_knn_test, testData_scale1$Churn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 842  96
##          1  13  49
##
##                Accuracy : 0.891
##                  95% CI : (0.87, 0.9096)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.0004856
##
##                   Kappa : 0.4233
##
##  Mcnemar's Test P-Value : 4.024e-15
##
##             Sensitivity : 0.9848
##             Specificity : 0.3379
##          Pos Pred Value : 0.8977
##          Neg Pred Value : 0.7903
##              Prevalence : 0.8550
##          Detection Rate : 0.8420
##    Detection Prevalence : 0.9380
##       Balanced Accuracy : 0.6614
##
##        'Positive' Class : 0
##

# Performing Naive Bayes

NB <- naiveBayes(trainData_scale1$Churn~., data = trainData_scale1)
summary(NB)

##           Length Class  Mode
## apriori    2     table  numeric
## tables    10     -none- list
## levels     2     -none- character
## isnumeric 10     -none- logical
## call       4     -none- call

#Performance Matrix on Train Data

pred_nb_train <- predict(NB, newdata = trainData_scale1)
confusionMatrix(pred_nb_train, trainData_scale1$Churn)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1933  238
##          1   62  100
##
##                Accuracy : 0.8714
##                  95% CI : (0.8571, 0.8847)
##     No Information Rate : 0.8551
##     P-Value [Acc > NIR] : 0.01273
##
##                   Kappa : 0.3378
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9689
##             Specificity : 0.2959
##          Pos Pred Value : 0.8904
##          Neg Pred Value : 0.6173
##              Prevalence : 0.8551
##          Detection Rate : 0.8285
##    Detection Prevalence : 0.9306
##       Balanced Accuracy : 0.6324
##
##        'Positive' Class : 0
##
```

```
#Performance Matrix on Test Data

pred_nb_test <- predict(NB, newdata = testData_scale1)
confusionMatrix(pred_nb_test, testData_scale1$Churn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 818 101
##          1  37  44
##
##                Accuracy : 0.862
##                  95% CI : (0.8391, 0.8828)
##     No Information Rate : 0.855
##     P-Value [Acc > NIR] : 0.2821
##
##                   Kappa : 0.3186
##
##  Mcnemar's Test P-Value : 8.189e-08
##
##             Sensitivity : 0.9567
##             Specificity : 0.3034
##          Pos Pred Value : 0.8901
##          Neg Pred Value : 0.5432
##              Prevalence : 0.8550
##          Detection Rate : 0.8180
##    Detection Prevalence : 0.9190
##       Balanced Accuracy : 0.6301
##
##        'Positive' Class : 0
##
```