Customer Churn

Muthu Pandian G

November 24, 2019

Telecom Customer Churn Prediction Assessment

OBJECTIVE OF THE PROJECT:

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.

As an Analyst You are expected to do the following:

1. EDA

2. Build Models and compare them to get to the best one

3. Model Comparison using Model Performance metrics & Interpretation

4. Actionable Insights

5. Interpretation & Recommendations from the best model

Importing the Dataset

```
setwd("D:/Great Lakes/Projects/Predictive Modeling")
getwd()

## [1] "D:/Great Lakes/Projects/Predictive Modeling"

library(openxlsx)
churn <- read.xlsx("Cellphone.xlsx",2,startRow = 1,colNames = TRUE)
```

Understanding the data

Data Description

The dataset has details on 3333 customers with 11 Variables.The Following Table Explains the Variable Name and Its Meaning.

Variables	Meaning
Churn	1 if customer cancelled service, 0 if not
AccountWeeks	number of weeks customer has had active account
ContractRenewal	1 if customer recently renewed contract, 0 if not
DataPlan	1 if customer has data plan, 0 if not
DataUsage	gigabytes of monthly data usage
CustServCalls	number of calls into customer service
DayMins	average daytime minutes per month
DayCalls	average number of daytime calls
MonthlyCharge	average monthly bill
OverageFee	largest overage fee in last 12 months
RoamMins	average number of roaming minutes

Structure of Data

```
str(churn)

## '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 ...
```

We can see that all the Variables are taken as numerical Variable only, and we know that variables like **Churn, Contract Renewal, Data Plan and Customer service calls** are Categorical Variable.

So, we need to convert them to categorical variable. Let's Convert the class of the Numeric Variable which are supposed to be Categorical Variable to Factor

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

Now let's look at the Structure of our Dataset

```
str(churn)

## '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  : Factor w/ 10 levels "0","1","2","3",..: 2 2 1 3 4 1 4 1 2
1 ...
## $ 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

```r
summary(churn)
```

```
## 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
##
```

```r
summary(churn$Churn)
```

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

```r
483/(2850+483)
```

```
## [1] 0.1449145
```

Around 14 % of data has customers who have churned out.

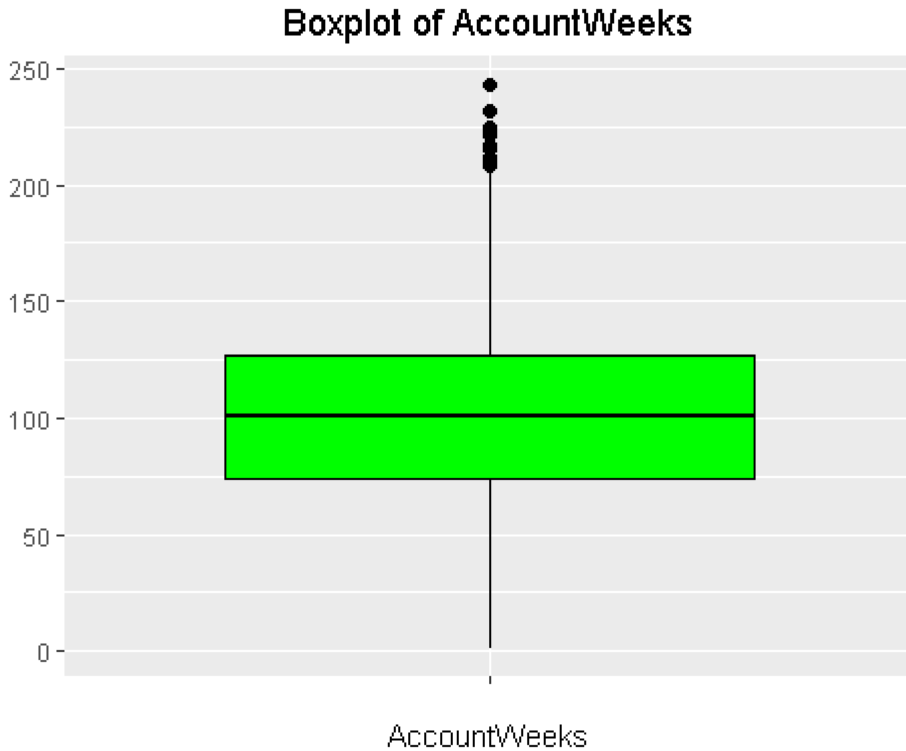
Checking NA Values/ Missing Values

```
colSums(is.na(churn))

##          Churn    AccountWeeks ContractRenewal        DataPlan
##              0               0               0               0
##      DataUsage   CustServCalls         DayMins        DayCalls
##              0               0               0               0
##   MonthlyCharge     OverageFee        RoamMins
##              0               0               0
```

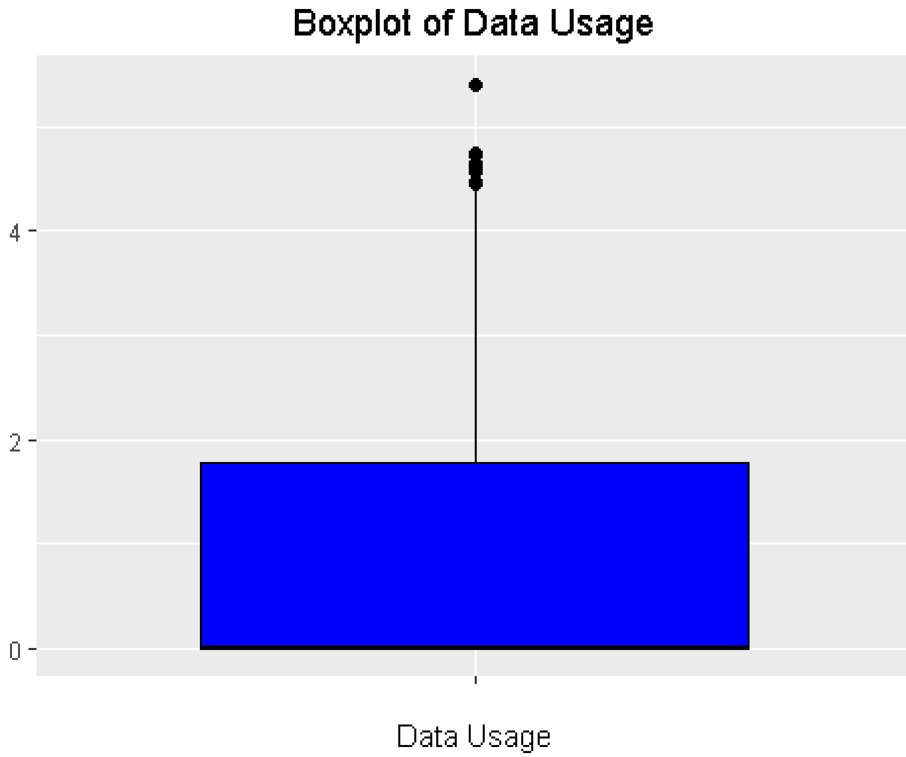
Luckily, we don't have any NA in our Dataset, then it's Time to check For Outlier.

Outlier Detection

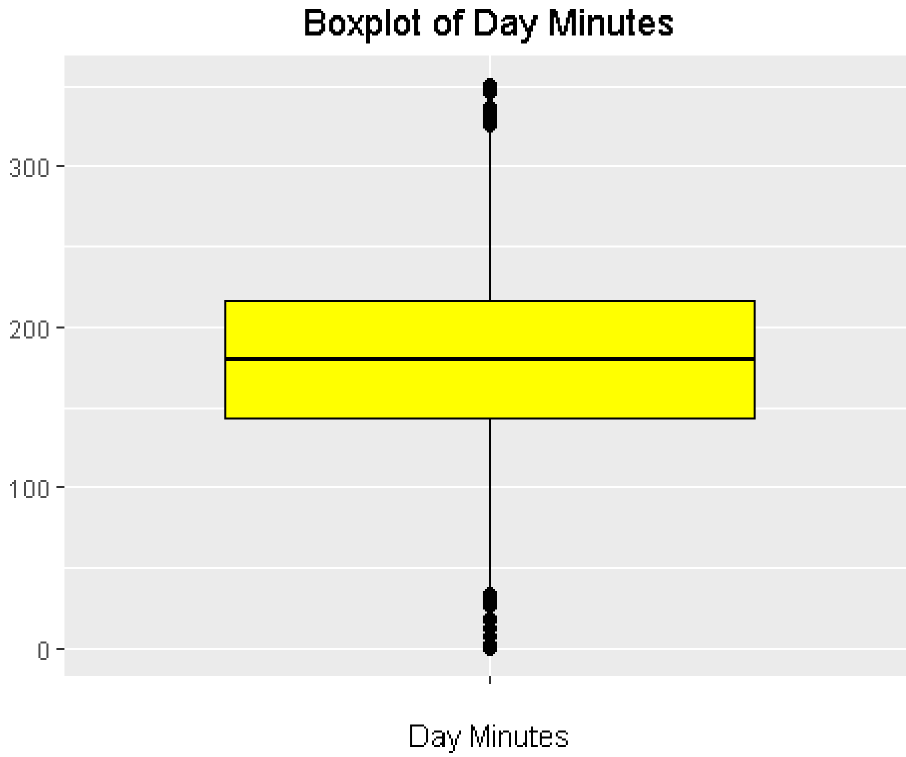
```
library(ggplot2)
ggplot(churn,aes(x="",churn$AccountWeeks))+
geom_boxplot(fill='green', color="black",outlier.colour="black", outlier.shape=16,outlier.size=2, notch=FALSE)+ labs(x = "AccountWeeks" , y = "") + theme(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of AccountWeeks")
```



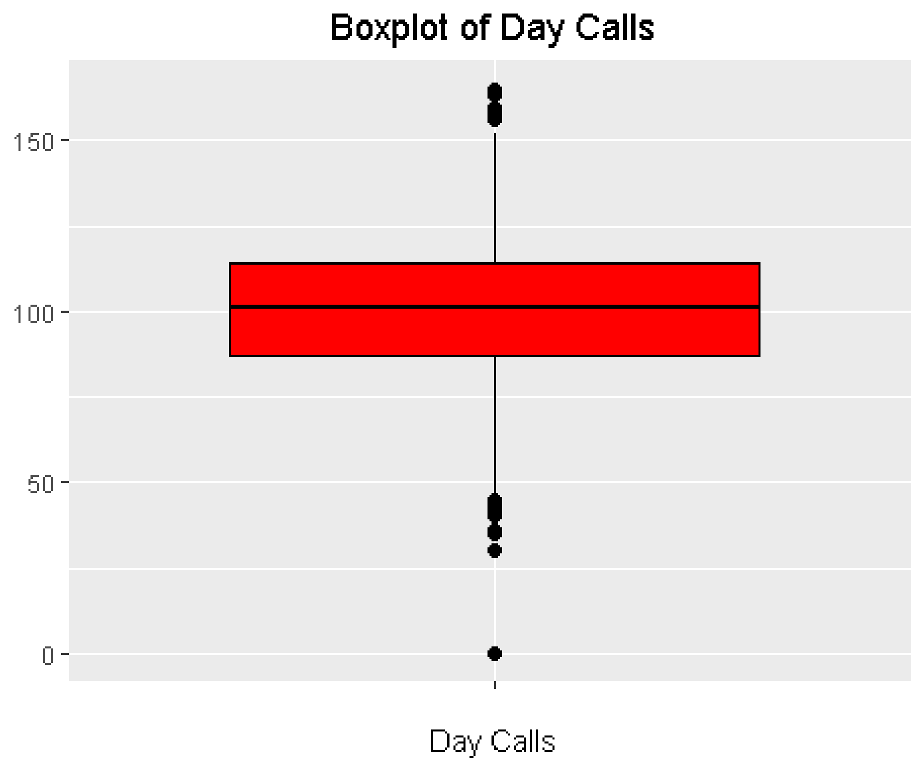

```
ggplot(churn,aes(x="",churn$DataUsage))+
geom_boxplot(fill='blue', color="black",outlier.colour="black", outlier.shape
=16,outlier.size=2, notch=FALSE)+ labs(x = "Data Usage" , y = "") + theme
(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Data Usage")
```



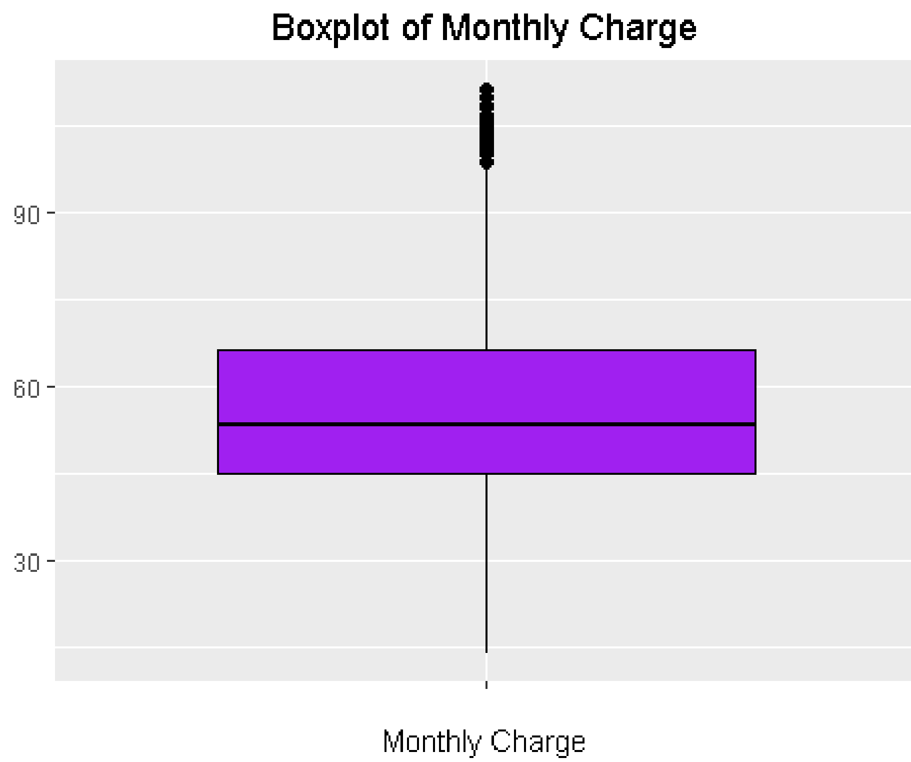

```
ggplot(churn,aes(x="",churn$DayMins))+
geom_boxplot(fill='yellow', color="black",outlier.colour="black", outlier.sha
pe=16,outlier.size=2, notch=FALSE)+ labs(x = "Day Minutes" , y = "") + the
me(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Day Minutes")
```



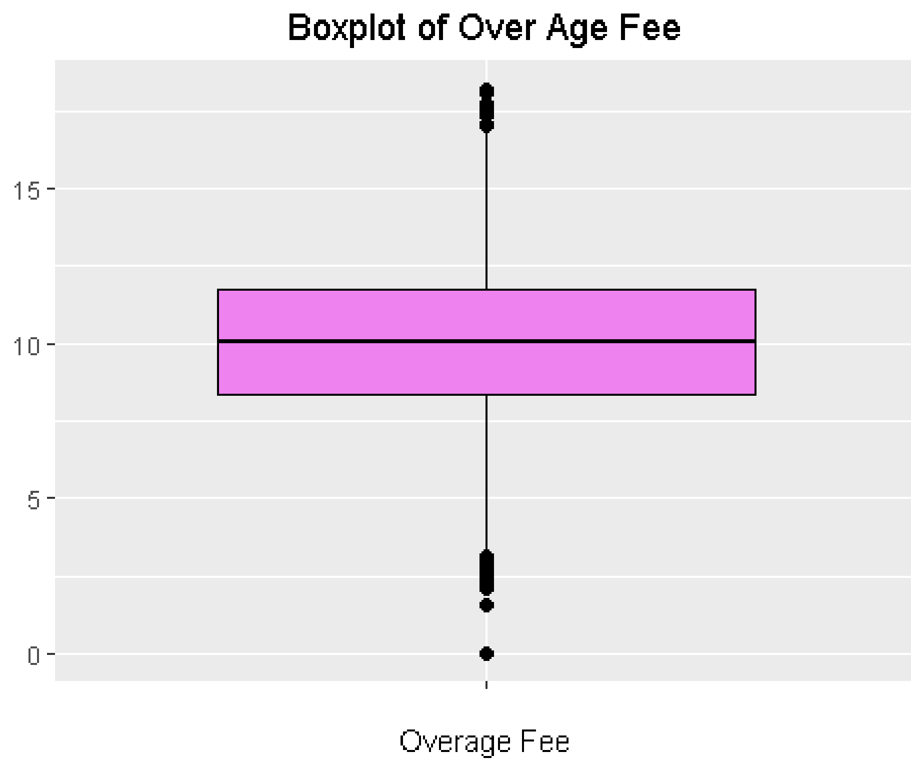

```
ggplot(churn,aes(x="",churn$DayCalls))+
geom_boxplot(fill='red', color="black",outlier.colour="black", outlier.shape=
16,outlier.size=2, notch=FALSE)+ labs(x = "Day Calls" , y = "") + theme(pl
ot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Day Calls")
```



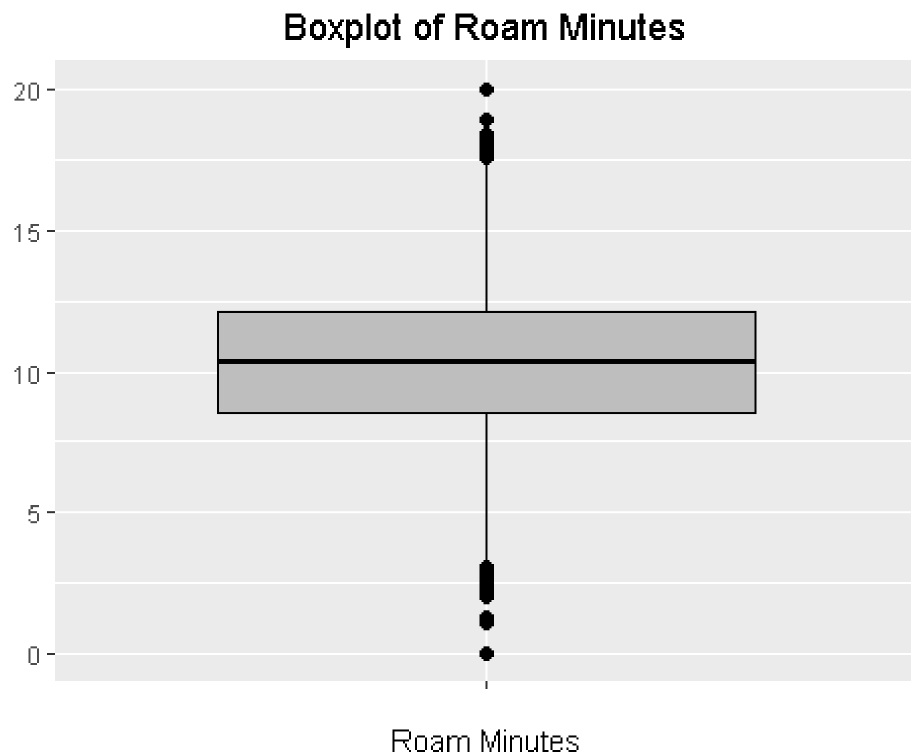

```
ggplot(churn,aes(x="",churn$MonthlyCharge))+
geom_boxplot(fill='Purple', color="black",outlier.colour="black", outlier.sha
pe=16,outlier.size=2, notch=FALSE)+ labs(x = "Monthly Charge" , y = "") +
theme(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Monthly C
harge")
```




```
ggplot(churn,aes(x="",churn$OverageFee))+
geom_boxplot(fill='Violet', color="black",outlier.colour="black", outlier.shape=16,outlier.size=2, notch=FALSE)+ labs(x = "Overage Fee" , y = "") + theme(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Over Age Fee")
```




```
ggplot(churn,aes(x="",churn$RoamMins))+
geom_boxplot(fill='grey', color="black",outlier.colour="black", outlier.shape
=16,outlier.size=2, notch=FALSE)+ labs(x = "Roam Minutes" , y = "") + the
me(plot.title = element_text(hjust = 0.5))+ggtitle("Boxplot of Roam Minute
s")
```



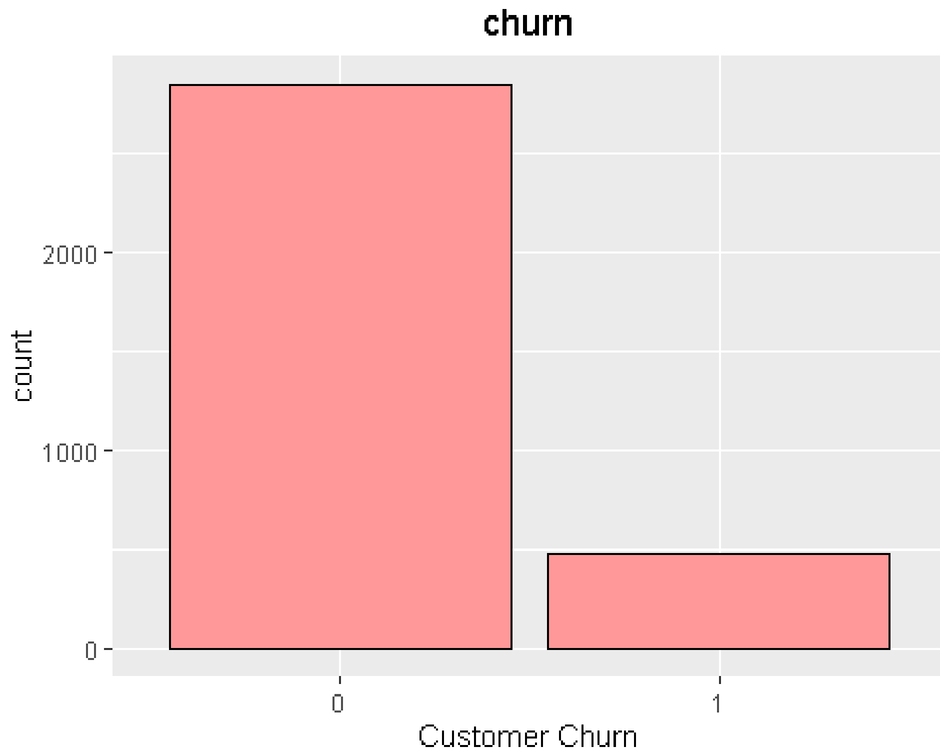
We Found out the existence of Outliers in the Following Variables: **Account Weeks, Data Usage, Day Minutes, Day Calls, Monthly Charge, OverAge Fee, Roam Minutes**

Exploratory Data Analysis

Univariate Analysis

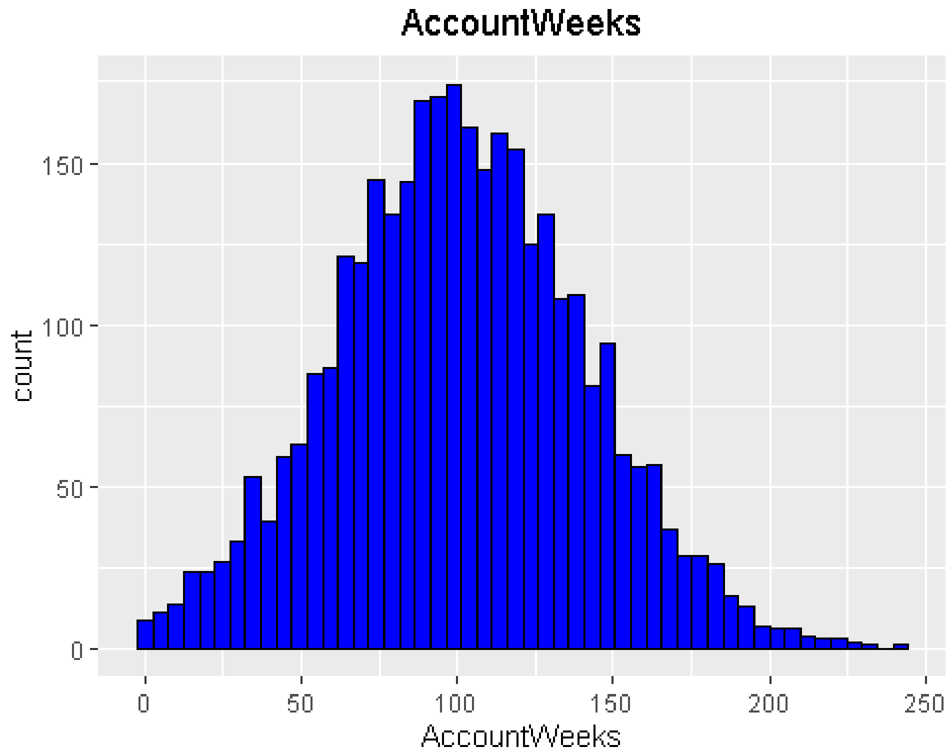
Frequency Distribution of each Independent numerical Variable

```
ggplot(churn,aes(x=Churn))+geom_bar(fill = "#FF9999",colour = "Black")+
ggtitle("churn")+theme(plot.title = element_text(hjust = 0.5))+xlab("Custo
mer Churn")
```



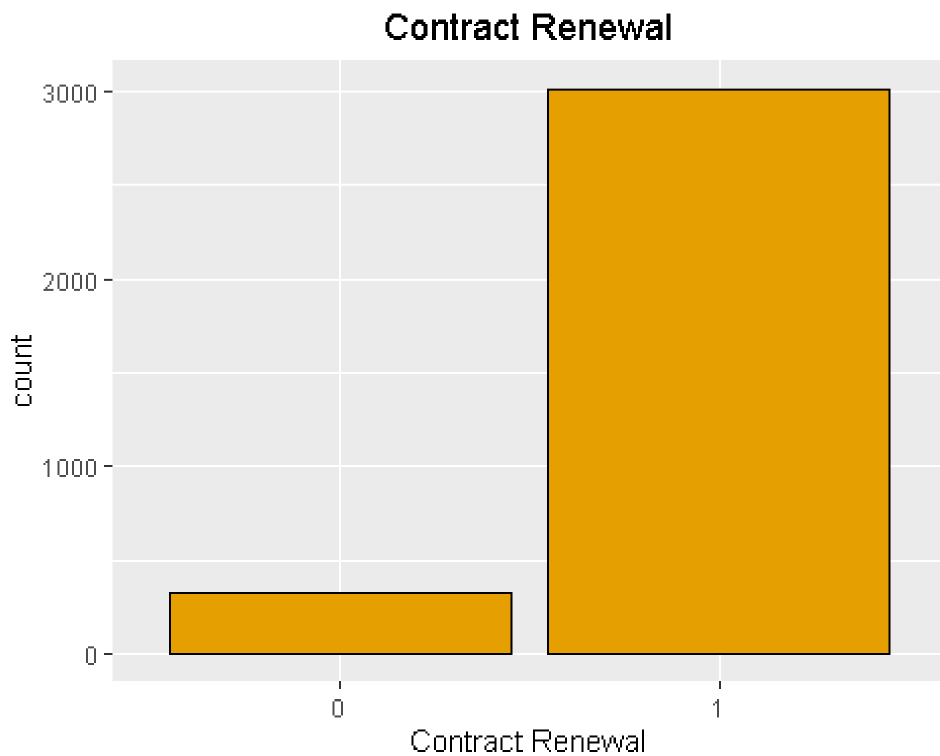
Churn - There are 2850 (86%) customers who haven't Churned Out and 483(14%) customers who have churned out

```
ggplot(churn,aes(x=AccountWeeks))+geom_histogram(bins = 50,fill = "Blue",colour = "Black")+ggtitle("AccountWeeks")+theme(plot.title = element_text(hjust = 0.5))+xlab("AccountWeeks")
```



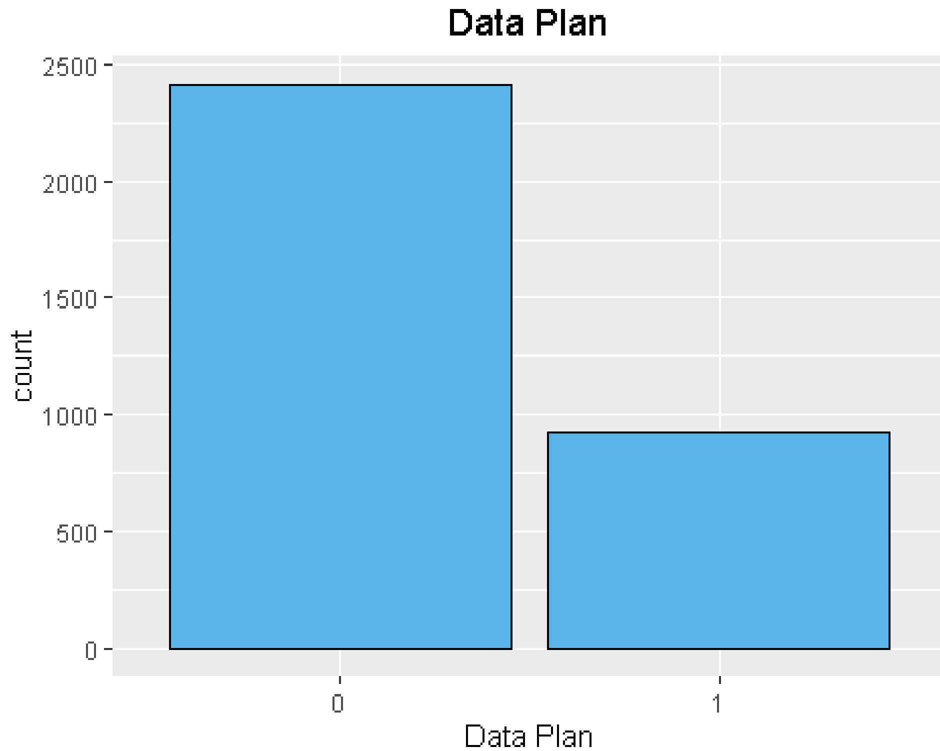
Account Weeks - The Number of Hours the customer had active accounts ranges from minimum of 1 week to maximum of 243 weeks. The Average Number of week lies around 101

```
ggplot(churn,aes(x=ContractRenewal))+geom_bar(fill = "#E69F00",colour = "Black")+ggtitle("Contract Renewal")+theme(plot.title = element_text(hjust = 0.5))+xlab("Contract Renewal")
```



ContractRenewal - There are 323(10%) customers who haven’t renewed thier Contract and 3010(90%) customers who have renewed thier Contract

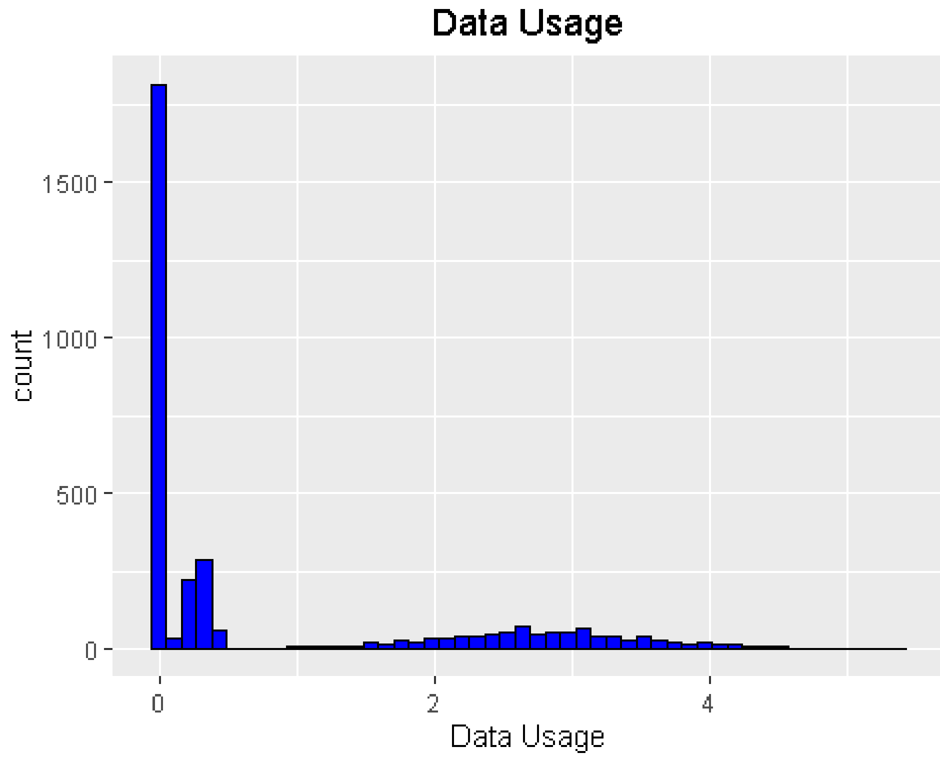
```
ggplot(churn,aes(x=DataPlan))+geom_bar(fill = "#56B4E9",colour = "Black")+ggtitle("Data Plan")+theme(plot.title = element_text(hjust = 0.5))+xlab("Data Plan")
```



Data Plan - There are 2411(72%) customers who didn't have data Plan and 922(18%) customers who have data plan.

It Shows that most of our customers don't use our Data Plans.

```
ggplot(churn,aes(x=DataUsage))+geom_histogram(bins = 50,fill = "Blue",
colour = "Black")+ggtitle("Data Usage")+theme(plot.title = element_text(hj
ust = 0.5))+xlab("Data Usage")
```

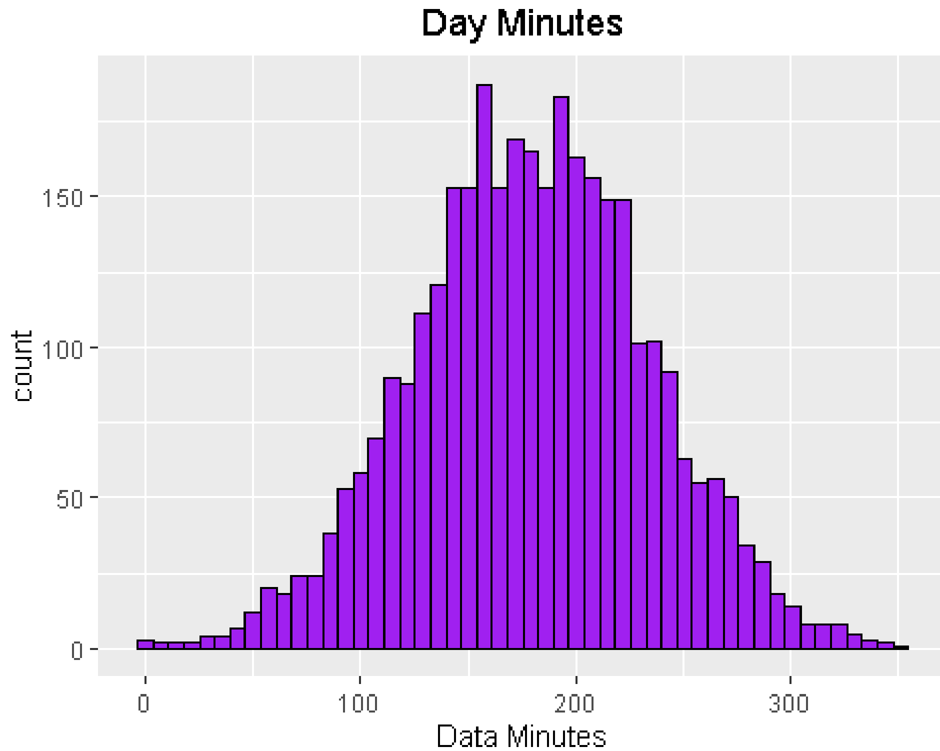


Data Usage - The Maximum Data Used by our customers is 5.4 Gb, on an average our customers use 0.8 gb.

```
ggplot(churn,aes(x = CustServCalls))+ geom_bar(fill = "#009E73",colour = "Black")+ ggtitle("Customer Service Calls") + theme(plot.title = element_text(hjust = 0.5))+xlab("Customer Service Calls")
```

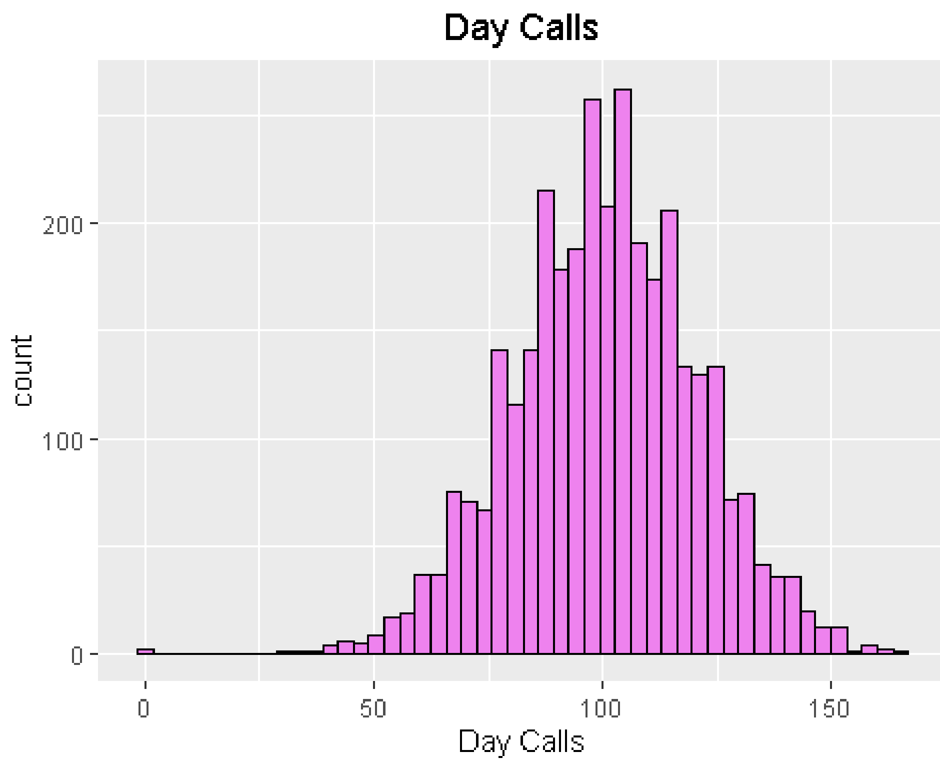
Customer Service Calls - A lot of our customers atleast have one Customer service calls

```
ggplot(churn,aes(x=DayMins))+geom_histogram(bins = 50,fill = "purple",
colour = "Black")+ggtitle("Day Minutes")+theme(plot.title = element_text(h
just = 0.5))+xlab("Data Minutes")
```



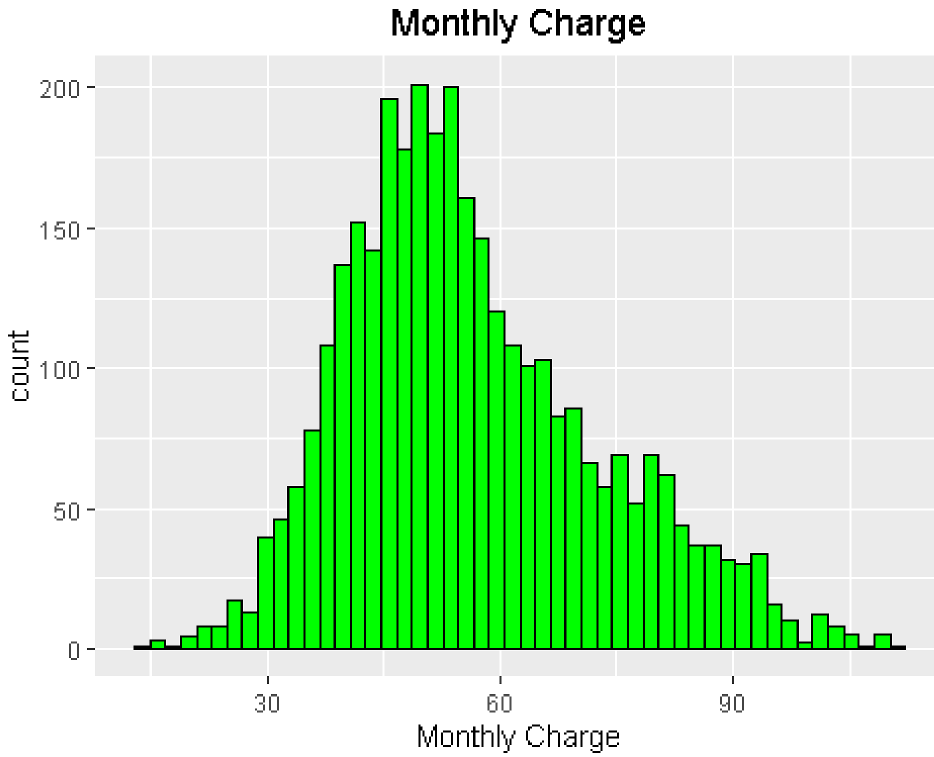
DayMins - On an average 180 Minutes per day is spent on Call by our customers, it reaches up to 3510 Minutes per day.

```
ggplot(churn,aes(x=DayCalls))+geom_histogram(bins = 50,fill = "violet",colour = "Black")+ggtitle("Day Calls")+theme(plot.title = element_text(hjust = 0.5))+xlab("Day Calls")
```



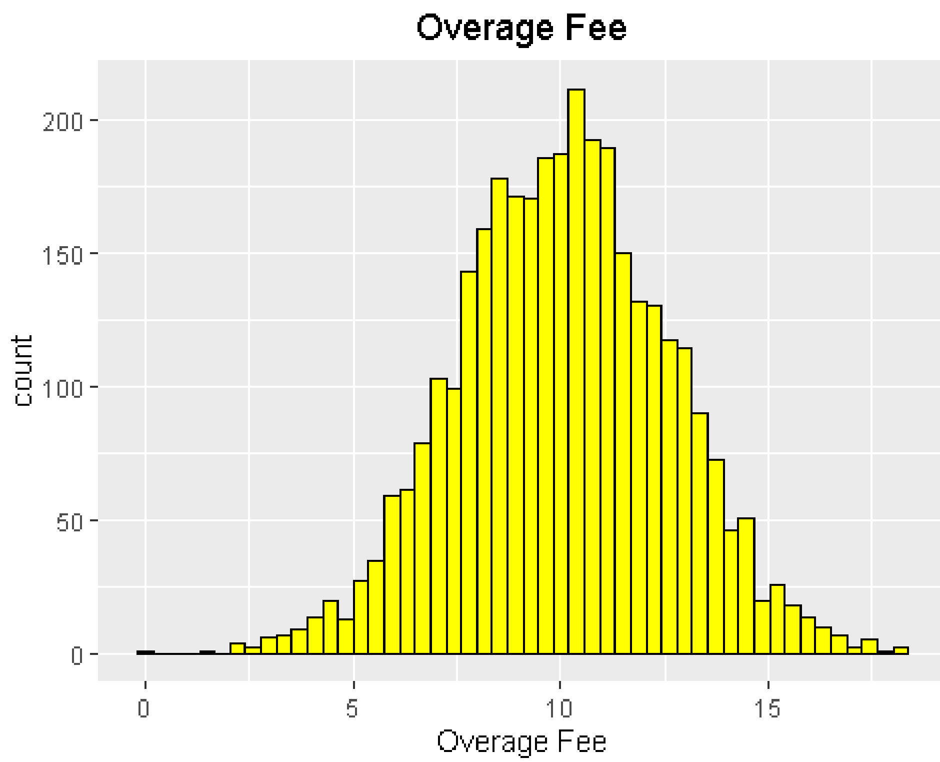
Daycalls - On an average around 100 Day Time calls are spoken by our customers, it reaches up to 165 Day Calls per day.

```
ggplot(churn,aes(x=MonthlyCharge))+geom_histogram(bins = 50,fill = "green",colour = "Black")+ggtitle("Monthly Charge")+theme(plot.title = element_text(hjust = 0.5))+xlab("Monthly Charge")
```



MonthlyCharge - Average Monthly Bill comes around 57 rupees per month and it reaches upto 112 rupees per month.

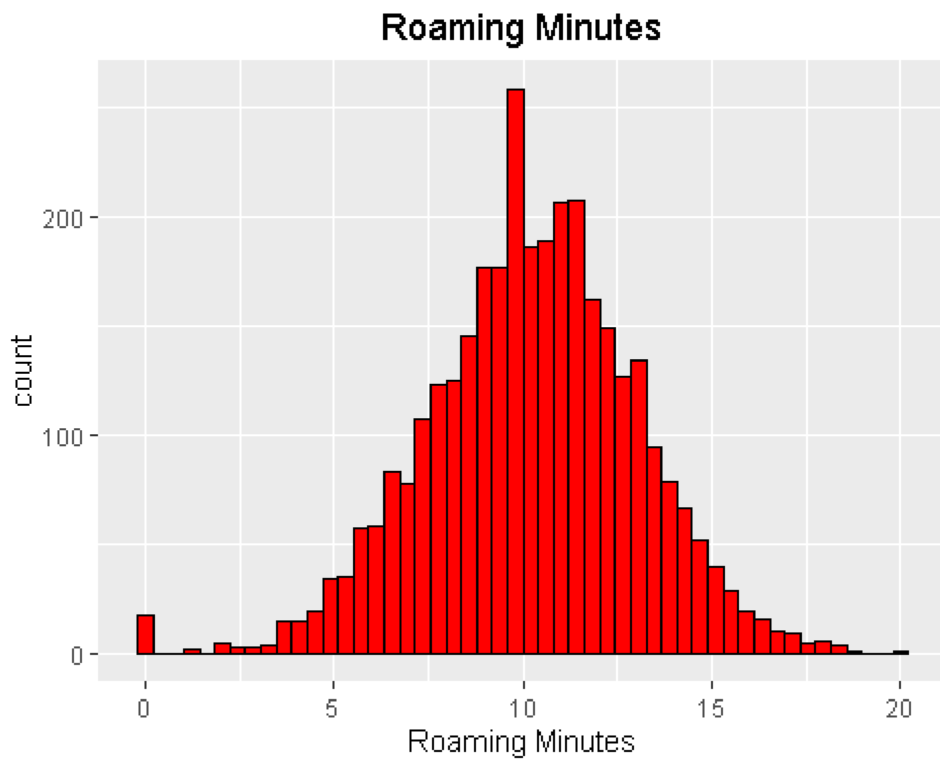
```
ggplot(churn,aes(x=OverageFee))+geom_histogram(bins = 50,fill = "yellow",colour = "Black")+ggtitle("Overage Fee")+theme(plot.title = element_text(hjust = 0.5))+xlab("Overage Fee")
```



Overage Fee - An Overage Fee is an extra amount of money that you have to pay for using more of something than was expected or agreed.

The Average Overage Fee in last 12 Months is 10rupees and it reaches up to 19 rupees.

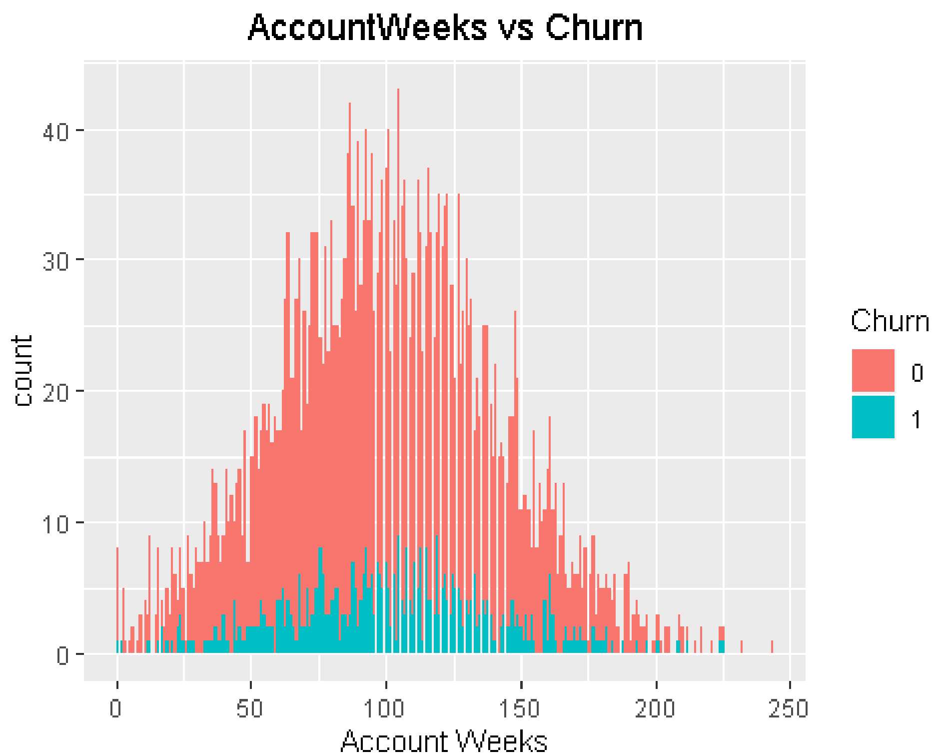
```
ggplot(churn,aes(x=RoamMins))+geom_histogram(bins = 50,fill = "Red",
colour = "Black")+ggtitle("Roaming Minutes")+theme(plot.title = element_t
ext(hjust = 0.5))+xlab("Roaming Minutes")
```



RoamMins - Average number of Roaming Minutes is 10 minutes and it reaches a maximum of 20 Minutes.

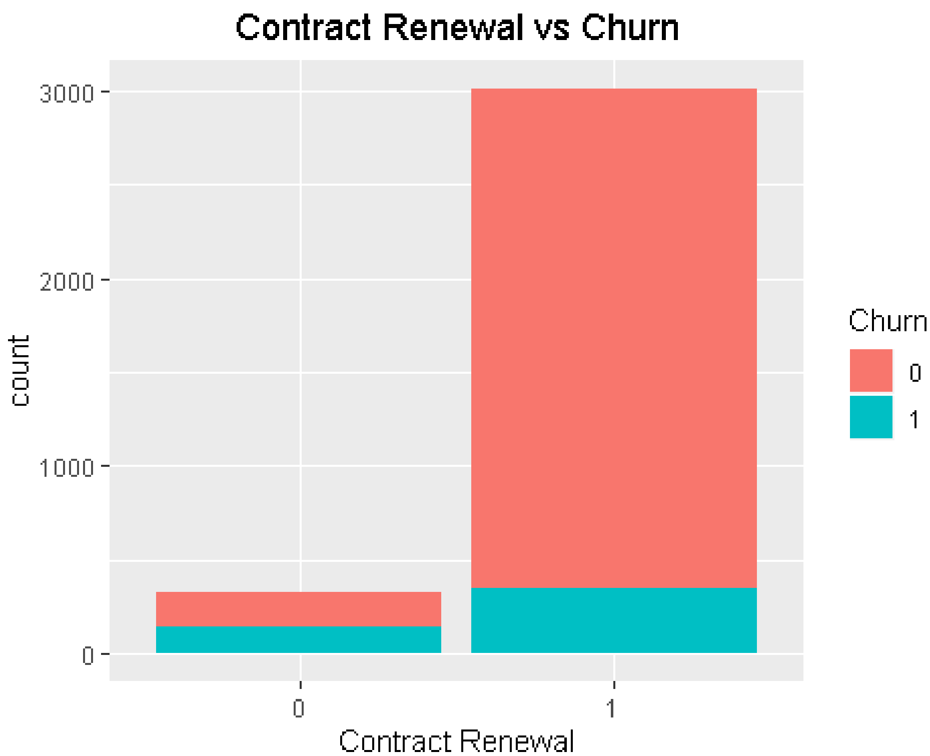
Bi Variate analysis

```
ggplot(churn,aes(AccountWeeks,fill = Churn))+geom_bar()+ggtitle("AccountWeeks vs Churn")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Account Weeks")
```



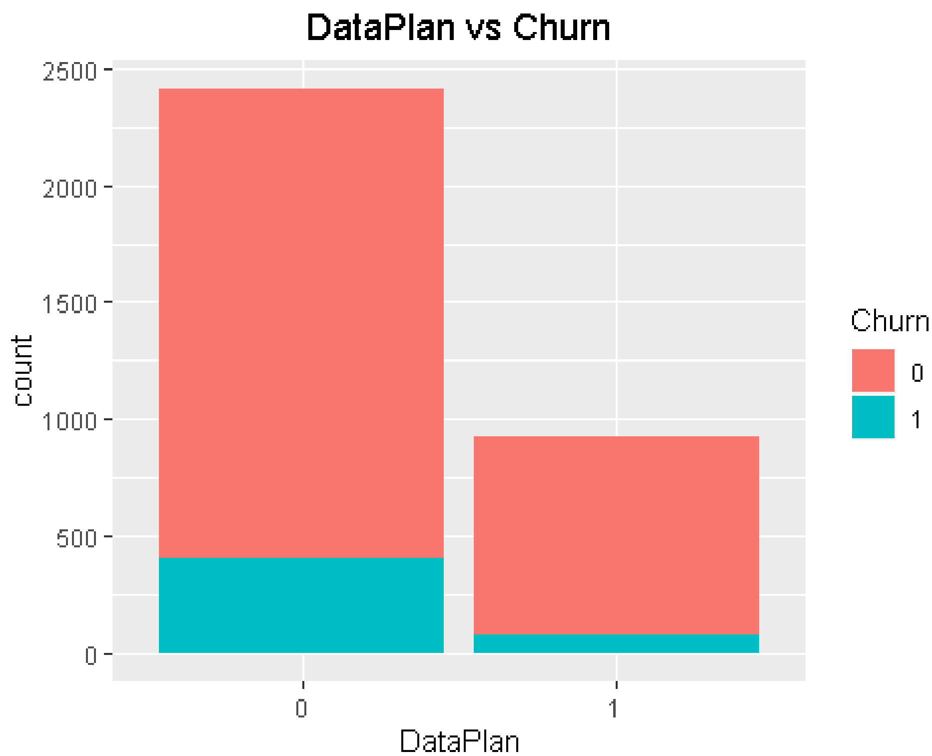
From the Graph, we can clearly see that Lesser the Customer stays higher the chance of Churning Out

```
ggplot(churn,aes(ContractRenewal,fill = Churn))+geom_bar()+ggtitle("Contract Renewal vs Churn")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Contract Renewal")
```



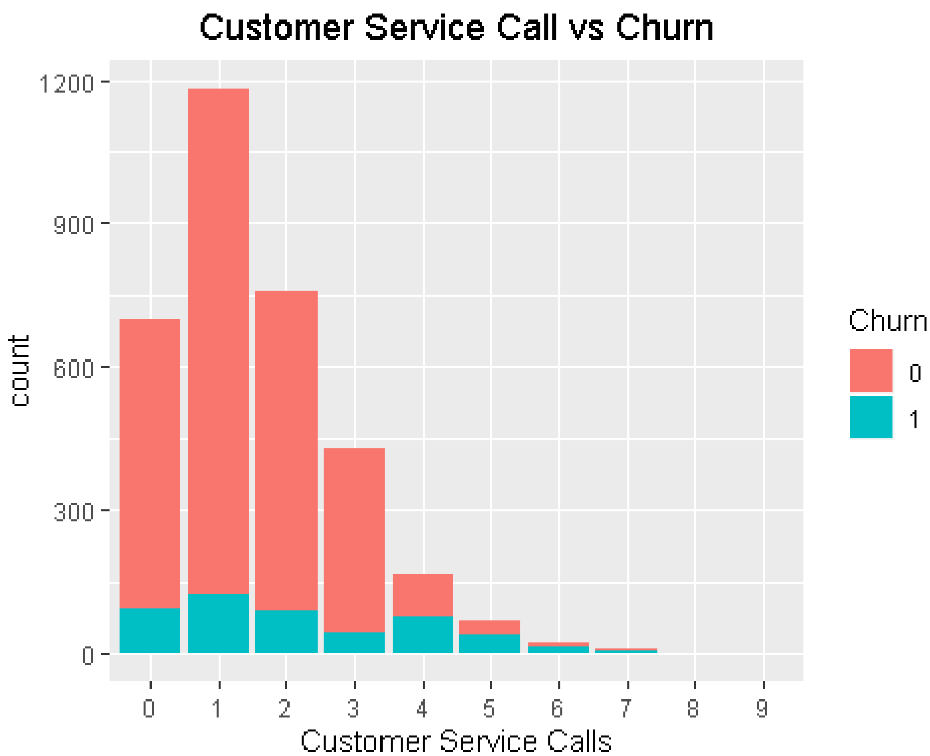
As expected, people who have renewed the contract has less possibility of churning out

```
ggplot(churn,aes(DataPlan,fill = Churn))+geom_bar()+ggtitle("DataPlan vs Churn")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("DataPlan")
```



Here, the customer who doesn't have data plan have churned out lot more than the customers who have data plan.

```
ggplot(churn,aes(CustServCalls,fill = Churn))+geom_bar()+ggtitle("Customer Service Call vs Churn")+ theme(plot.title = element_text(hjust = 0.5))+
xlab("Customer Service Calls")
```

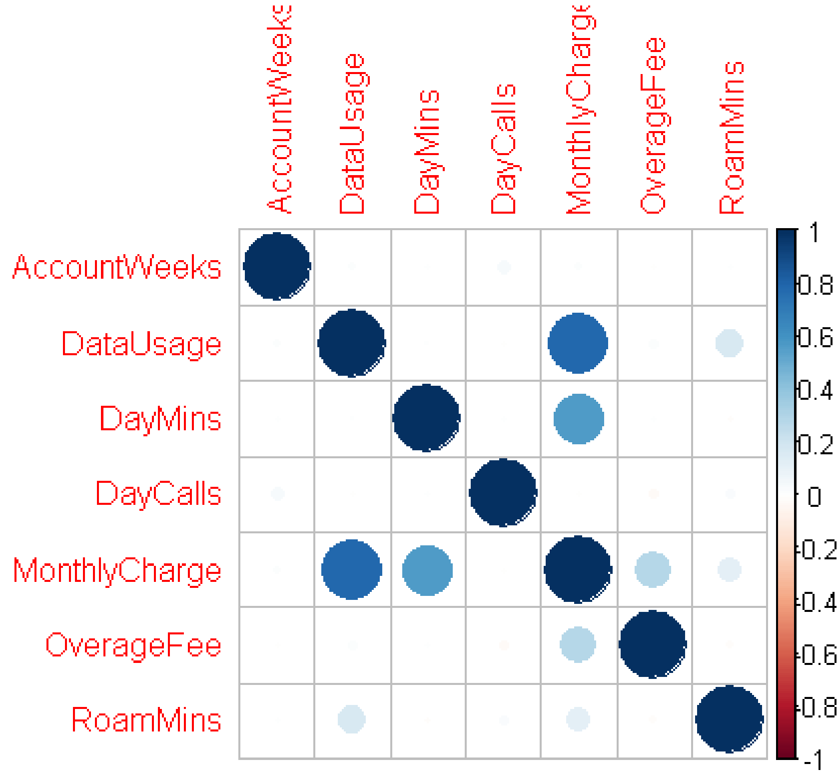


Higher the Customer Service Calls made by the Customers Higher the Possibility of churning out

Multi-Collinearity

Let's checkout the existence of Multi-collinearity between the Independent variables

```
library(corrgram)

## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus

library(corrplot)

## corrplot 0.84 loaded

library(car)

## Loading required package: carData

corrplot::corrplot(corrgram(churn[,-c(1,3,4,6)]))
```



From the above plot, it’s clear that the Variable “Monthly Charge” is positively correlated with Data usage (0.78), Day minutes (0.567) and slightly correlated with Overage Fee (0.28), Roam Minutes (0.11).

Roam Minutes has a 0.16 correlation with Data Usage.

```
cor(churn[,-c(1,3,4,6)])

##               AccountWeeks   DataUsage     DayMins     DayCalls
## AccountWeeks   1.000000000  0.014390757  0.006216021  0.03846988
2
## DataUsage      0.014390757  1.000000000  0.003175951 -0.007962079
## DayMins        0.006216021  0.003175951  1.000000000  0.006750414
## DayCalls       0.038469882 -0.007962079  0.006750414  1.000000000
## MonthlyCharge  0.012580670  0.781660429  0.567967924 -0.00796321
8
## OverageFee    -0.006749462  0.019637372  0.007038214 -0.02144860
2
## RoamMins       0.009513902  0.162745576 -0.010154586  0.021564794
```

```
##              MonthlyCharge   OverageFee     RoamMins
## AccountWeeks    0.012580670 -0.006749462  0.009513902
## DataUsage       0.781660429  0.019637372  0.162745576
## DayMins         0.567967924  0.007038214 -0.010154586
## DayCalls       -0.007963218 -0.021448602  0.021564794
## MonthlyCharge   1.000000000  0.281766048  0.117432607
## OverageFee      0.281766048  1.000000000 -0.011023336
## RoamMins        0.117432607 -0.011023336  1.000000000
```

It's Evident that Multicollinearity is exist in the dataset.

Now, let's calculate the VIF value and decide how to treat Multi-Collinearity

```
model <- glm(Churn~.,churn,family = "binomial")
summary(model)
```

```
## 
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = churn)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.3675  -0.4740  -0.3278  -0.1978   3.0712  
## 
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)    
## (Intercept)     -5.499e+00  5.703e-01  -9.642  < 2e-16 ***
## AccountWeeks     9.247e-04  1.453e-03   0.636 0.524534    
## ContractRenewal1 -2.009e+00  1.463e-01 -13.733  < 2e-16 ***
## DataPlan1       -1.316e+00  5.594e-01  -2.353 0.018604 *  
## DataUsage        4.170e-01  2.002e+00   0.208 0.834973    
## CustServCalls1  -1.975e-01  1.633e-01  -1.210 0.226451    
## CustServCalls2   2.623e-02  1.764e-01   0.149 0.881768    
## CustServCalls3  -1.962e-01  2.116e-01  -0.927 0.353817    
## CustServCalls4   2.113e+00  2.200e-01   9.603  < 2e-16 ***
## CustServCalls5   3.066e+00  3.096e-01   9.904  < 2e-16 ***
## CustServCalls6   3.928e+00  5.038e-01   7.796 6.37e-15 ***
## CustServCalls7   3.084e+00  6.967e-01   4.427 9.58e-06 ***
## CustServCalls8   3.063e+00  1.434e+00   2.135 0.032751 *  
## CustServCalls9   1.495e+01  3.402e+02   0.044 0.964960    
```

```
## DayMins          1.936e-02  3.380e-02   0.573 0.566836
## DayCalls         2.969e-03  2.838e-03   1.046 0.295598
## MonthlyCharge    -3.287e-02  1.986e-01  -0.166 0.868549
## OverageFee       2.019e-01  3.388e-01   0.596 0.551179
## RoamMins         8.057e-02  2.298e-02   3.505 0.000456 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2758.3  on 3332  degrees of freedom
## Residual deviance: 2061.9  on 3314  degrees of freedom
## AIC: 2099.9
##
## Number of Fisher Scoring iterations: 12
```

```
vif(model)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## AccountWeeks       1.008202  1        1.004092
## ContractRenewal    1.055737  1        1.027491
## DataPlan          14.041482  1        3.747197
## DataUsage       1605.799054  1       40.072423
## CustServCalls      1.176779  9        1.009084
## DayMins          968.646585  1       31.123088
## DayCalls           1.010592  1        1.005282
## MonthlyCharge   2842.190923  1       53.312202
## OverageFee       215.462663  1       14.678647
## RoamMins           1.198184  1        1.094616
```

As hinted in the Correlation Matrix plot, we can clearly see that Monthly Charge has high VIF.

Let’s remove the Monthly charge, data plan, data Usage and check the VIF for other predictors

```
churn1 <- churn[,-c(4,5,9)]
model1 <- glm(Churn~.,churn1,family = "binomial")
vif(model1)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## AccountWeeks    1.005583  1        1.002788
```

```
## ContractRenewal 1.044303  1       1.021911
## CustServCalls   1.134694  9       1.007045
## DayMins         1.072077  1       1.035412
## DayCalls        1.009964  1       1.004970
## OverageFee      1.024308  1       1.012081
## RoamMins        1.011195  1       1.005582
```

The Vif of all the other variables are around 1, i.e. they are less correlated with each other.

```
summary(model1)

##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = churn1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2649  -0.4734  -0.3451  -0.2288   2.9979
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -5.580e+00  5.513e-01 -10.122  < 2e-16 ***
## AccountWeeks      7.591e-04  1.439e-03   0.528 0.597755
## ContractRenewal1 -1.946e+00  1.430e-01 -13.604  < 2e-16 ***
## CustServCalls1   -2.121e-01  1.613e-01  -1.315 0.188382
## CustServCalls2    4.646e-02  1.738e-01   0.267 0.789256
## CustServCalls3   -1.571e-01  2.096e-01  -0.749 0.453558
## CustServCalls4    2.073e+00  2.158e-01   9.605  < 2e-16 ***
## CustServCalls5    2.991e+00  3.081e-01   9.705  < 2e-16 ***
## CustServCalls6    3.634e+00  4.899e-01   7.419 1.18e-13 ***
## CustServCalls7    3.082e+00  7.087e-01   4.349 1.37e-05 ***
## CustServCalls8    2.677e+00  1.452e+00   1.843 0.065328 .
## CustServCalls9    1.495e+01  3.219e+02   0.046 0.962963
## DayMins           1.348e-02  1.115e-03  12.092  < 2e-16 ***
## DayCalls          2.989e-03  2.804e-03   1.066 0.286511
## OverageFee        1.345e-01  2.305e-02   5.837 5.33e-09 ***
## RoamMins          7.898e-02  2.092e-02   3.776 0.000159 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2758.3  on 3332  degrees of freedom
## Residual deviance: 2118.8  on 3317  degrees of freedom
## AIC: 2150.8
##
## Number of Fisher Scoring iterations: 12
```

From the model summary, it's clear that Account weeks, Day Calls are least significant variables, so let's remove those variables also.

Key-Insights From EDA

Uni-Variate Analysis:

- We have 14% of customers who have churned out and 86% of customers who haven't churned out in our dataset
- Majority of our customers (72%) don't have a Data Plan.
- It shows our customers primary need is Phone calls which is confirmed by Day Minutes Variable.
- On an Average Our Customers spend 3 hours per day on Phone Calls

Bi-Variate Analysis:

- Higher the Customer Service Calls made by the Customers Higher the Possibility of churning out
- The customer who doesn't have data plan have churned out lot more than the customers who have data plan.

Multi-collinearity and Outliers:

- We Found out the existence of Outliers in the Following Variables: Account Weeks, Data Usage, Day Minutes, Day Calls, Monthly Charge, OverAge Fee, Roam Minutes

- Then, we figured out that the Monthly Charge, Data Plan, Data Usage are highly correlated with the other variables and causing Misinterpretation.
- So, we removed them from our Data.

Let's Build the Model with the variables which have Low VIF, High Significance and check how it performs on the training and testing dataset.

Logistic Regression

```
library(caTools)# Used for Spliting the Data
set.seed(1234)
churn2 <- churn[,c(1,3,6,7,10)]
split <- sample.split(churn2$Churn, SplitRatio = 0.7)
train <- subset(churn2,split== TRUE)
test <- subset(churn2,split == FALSE)
LogTrainModel <- glm(Churn~.,train,family = "binomial")
summary(LogTrainModel)
```

```
##
## Call:
## glm(formula = Churn ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2235  -0.4668  -0.3463  -0.2338   2.9240
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -4.585977   0.457839 -10.017  < 2e-16 ***
## ContractRenewal1  -1.961813   0.175343 -11.188  < 2e-16 ***
## CustServCalls1    -0.230201   0.198865  -1.158    0.247
## CustServCalls2     0.189764   0.205370   0.924    0.355
## CustServCalls3    -0.124998   0.249690  -0.501    0.617
## CustServCalls4     2.221489   0.255584   8.692  < 2e-16 ***
## CustServCalls5     3.321722   0.378163   8.784  < 2e-16 ***
```

```
## CustServCalls6     3.150863   0.523637   6.017 1.77e-09 ***
## CustServCalls7     3.591036   0.911085   3.941 8.10e-05 ***
## CustServCalls8    17.799541 882.743396   0.020    0.984
## CustServCalls9    16.129784 539.290370   0.030    0.976
## DayMins            0.013081   0.001348   9.704  < 2e-16 ***
## OverageFee         0.158908   0.027836   5.709 1.14e-08 ***
## ---
## 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: 1483.0  on 2320  degrees of freedom
## AIC: 1509
##
## Number of Fisher Scoring iterations: 13
```

From the output above, the coefficients table shows the beta coefficient estimates and their significance levels.

The Columns are:

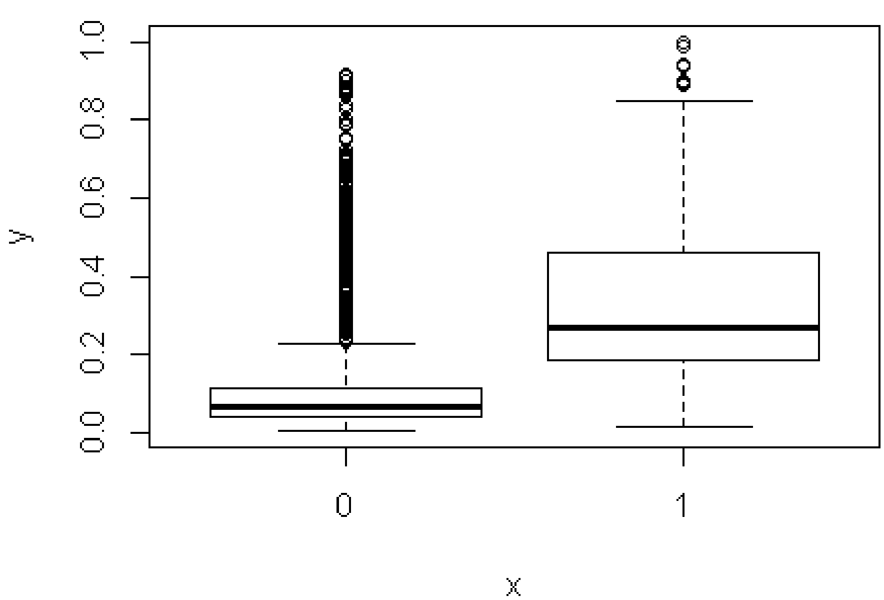
- **Estimate:** Estimate column gives the intercept (b0: -4.585977) and the beta coefficient estimates associated to each predictor variable
- **Std.Error:** The standard error of the coefficient estimates. This represents the accuracy of the coefficients. The larger the standard error, the less confident we are about the estimate.
- **z value:** the z-statistic, which is the coefficient estimate (column 2) divided by the standard error of the estimate (column 3)
- **Pr(>|z|):** The p-value corresponding to the z-statistic. The smaller the p-value, the more significant the estimate is.

It's Evident That ContractRenewal, CustServCalls (4,5,6,7), DayMins and Overage Fee are Highly Significant and have High Impact in predicting whether the customer churn out or not.

Now, Let's see how our Logistic model performs on both Training and Test Dataset.

- ✓ Logistic regression does not return directly the class of observations.
- ✓ It allows us to estimate the probability (p) of class membership.
- ✓ The probability will range between 0 and 1. We need to decide the threshold probability at which the category flips from one to the other.

```
Log_Prediction_Train <- predict(LogTrainModel,data = "train",type = "response")

plot(train$Churn,Log_Prediction_Train)
```



From the above Plot, we can clearly see that the customers who haven't churned out lies within 0-0.3.

So, let's take the **threshold** of **0.3**. The Probabilty predicted by our Model above 0.3 will be taken as 1 (Customers has high chance of Churn Out)

Model Perfomance on Training Data

Confusion Matrix

```
Log_model.predicted <- ifelse(Log_Prediction_Train<0.3,0,1)
Logmodel <- table(train$Churn,Log_model.predicted)
print(Logmodel)

##    Log_model.predicted
##        0    1
##   0 1825  170
##   1  195  143

# Accuracy
accuracy <- round(sum(diag(Logmodel))/sum(Logmodel),2)
print(accuracy)

## [1] 0.84

# Sensitivity
sensitivity <- round(143/(143+170),2)
print(sensitivity)

## [1] 0.46

# Specificity
specificity <-round(1825/(1825 + 195),2)
print(specificity)

## [1] 0.9
```

Confusion Matrix Inference on Training Dataset:

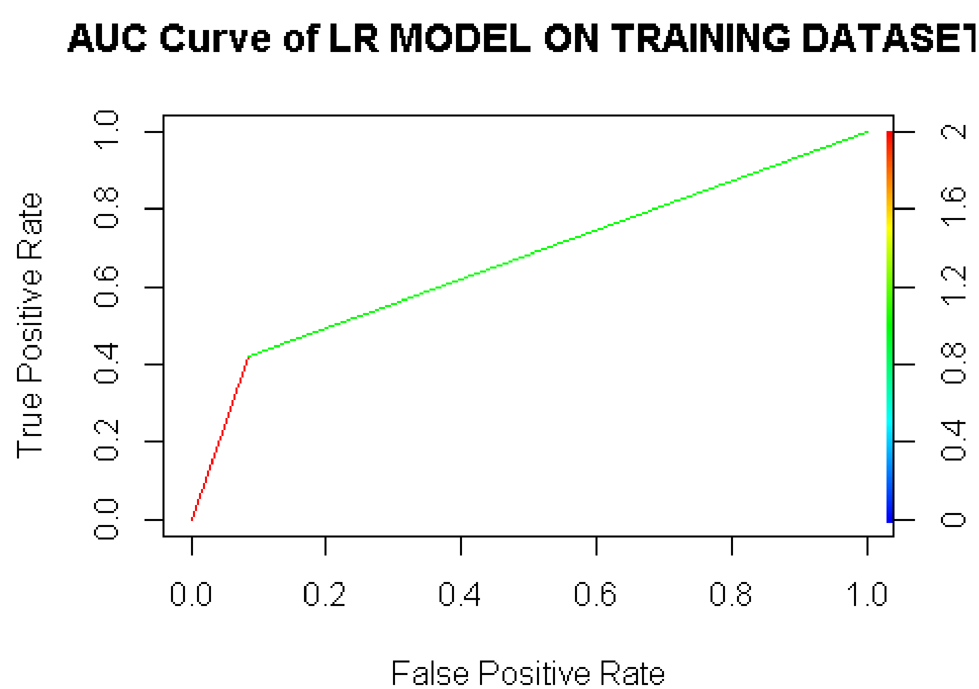
Based on Confusion Matrix,

- ✓ With 84% accuracy on the Training Dataset Our Model Done well (90%) in predicting the 0 (Customers who haven't churn out) than in Predicting (46%) the 1 (Customers who have Churn out).

Now, Let's Check Our Model with other Model Perfomance measures like AUC, Gini, KS

AUC & Gini

```
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

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

ROCRpred <- prediction(Log_model.predicted, train$Churn)
ROCRperf <- performance(ROCRpred, 'tpr','fpr')
plot(ROCRperf,colorize = TRUE, text.adj = c(-0.2,1.7),main="AUC Curve o
f LR MODEL ON TRAINING DATASET",xlab="False Positive Rate",ylab="
True Positive Rate")
```



```r
auc = performance(ROCRpred,"auc");
auc = as.numeric(auc@y.values)
print(auc)

## [1] 0.6689319

library(ineq)
gini = ineq(Log_model.predicted, type="Gini")
print(gini)

## [1] 0.865838
```

Thumb Rule - Larger the auc and gini coefficient better the model is.

We have an auc of 67% and gini coefficient of 87% which conveys the message that our model has done a Ok Job in training dataset.

KS

- ✓ KS Statistic or Kolmogorov-Smirnov statistic is the maximum difference between the cumulative true positive and cumulative false positive rate.
- ✓ It is often used as the deciding metric to judge the efficacy of models in credit scoring. The higher the ks_stat, the more efficient is the model at capturing the Ones.
- ✓ This should not be confused with the ks.test function.

```r
KS = max(ROCRperf@y.values[[1]]-ROCRperf@x.values[[1]]) # The Maximum the Better
print(KS)

## [1] 0.3378639
```

Here, In Training Dataset our Logistic Model done Poorly (0.34) in Predicting the customers who will cancel our services.

Model Performance on Test Data

Confusion Matrix

```
Log_Prediction_Test <- predict(LogTrainModel,test,type = "response")
Log_model.predicted1 <- ifelse(Log_Prediction_Test <0.3,0,1)
Logmodel1 <- table(test$Churn,Log_model.predicted1)
print(Logmodel1)

##    Log_model.predicted1
##       0   1
##   0 784  71
##   1  85  60

# Accuracy
Test_accuracy <- round(sum(diag(Logmodel1))/sum(Logmodel1),2)
print(Test_accuracy)

## [1] 0.84

# Sensitivity
sensitivity <- round(60/(60+71),2)
print(sensitivity)

## [1] 0.46

# Specificity
specificity <-round(784/(784 + 85),2)
print(specificity)

## [1] 0.9
```

Confusion Matrix Inference on Test Dataset:

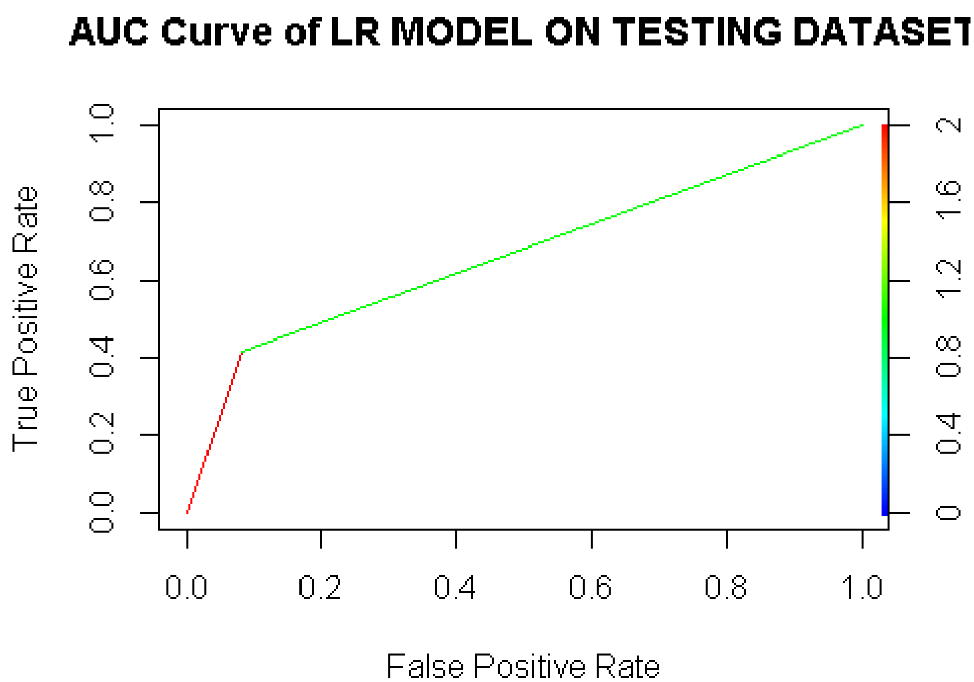
Based on Confusion Matrix,

- With 84% accuracy on the Training Dataset Our Logistic Model Done well (90%) in predicting the 0 (Customers who haven’t churn out) than in Predicting (46%) the 1 (Customers who have Churn out) as exactly the same Prediction in the Training Dataset.

Now Let’s Check Our Logistic Model with other Model Perfomance measures like AUC, Gini, KS

AUC & Gini

```
TestROCRpred <- prediction(Log_model.predicted1, test$Churn)
TestROCRperf <- performance(TestROCRpred, 'tpr','fpr')
plot(TestROCRperf,colorize = TRUE, text.adj = c(-0.2,1.7),main="AUC Cur
ve of LR MODEL ON TESTING DATASET",xlab="False Positive Rate",ylab
="True Positive Rate")
```




```
Testauc = performance(TestROCRpred,"auc");
Testauc = as.numeric(Testauc@y.values)
print(Testauc)

## [1] 0.6653761

# Gini on Test dataset
Testgini = ineq(Log_model.predicted1, type="Gini")
print(Testgini)

## [1] 0.869
```

Thumb Rule - Larger the auc and gini coefficient better the model is.

We have an auc of 67% and gini coefficient of 87% which conveys the message that our model has done a Ok Job in the test datset.

KS

The higher the ks-stat, the more efficient is the model at capturing the Ones.

```
TestKS = max(TestROCRperf@y.values[[1]]-TestROCRperf@x.values[[1]])
# The Maximum the Better
print(TestKS)

## [1] 0.3307522
```

CART MODEL

Here, In Test Dataset our Logistic Model done Poorly (0.33) in Predicting the customers who will cancel our services.

Our Model Has Performed exactly the Sameway in both the train and Test dataset.

Now, Let's Build a KNN Model and Measure its Performance

KNN

```
library(class)
knnmodel <- knn(train,test,train$Churn,k=5)
summary(knnmodel)

##   0   1
## 935  65
```

Interpretation:

- After Trail and Error Method @ k = 5 the Model performs well in predicting Both 0 (Customer who will not cancel) and 1 (Customer who will cancel) when compared to Logistic Regression model.
- Our Model Predicted 935 ‘0’ and 65 ‘1’.

Now, Let’s Check how well it have performed by using Confusion Matrix

Confusion Matrix

```
knntable <- table(test$Churn,knnmodel)
print(knntable)

##    knnmodel
##       0   1
##   0 845  10
##   1  90  55

# Accuracy
knnaccuracy <- round(sum(diag(knntable))/sum(knntable),2)
print(knnaccuracy)

## [1] 0.9

# Sensitivity
sensitivity <- round(55/(55+10),2)
print(sensitivity)

## [1] 0.85

# Specificity
specificity <-round(845/(845 + 90),2)
print(specificity)

## [1] 0.9
```

With 90% accuracy our KNN-Model has Done well in predicting both the 0 (90%) (Customers who haven’t churn out) 1 (85%) (Customers who have Churn out).

Let’s Look, How Naive Bayes Model works on this Dataset.

Naive Bayes

- ❖ The Naive Bayes is a classification algorithm that is suitable for binary and multiclass classification.
- ❖ Generally, Naive Bayes performs well in cases of categorical input variables compared to numerical variables.
- ❖ So, a version of Naive Bayes algorithm has been created where the predicted variable do not have to be descrete they can be continuous also.
- ❖ However, it is useful for making predictions and forecasting data based on historical results.
- ❖ Therefore, we can use Naive Bayes Algorithm for this use case.

Let's Build the model and see its performance on the Train and Test data.

```
library(e1071)
NBModel <- naiveBayes(Churn~.,data = train)
print(NBModel)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.8551222 0.1448778
##
## Conditional probabilities:
##    ContractRenewal
## Y            0          1
##   0 0.06065163 0.93934837
##   1 0.26331361 0.73668639
```

```
##
##    CustServCalls
## Y            0           1           2           3           4
##   0 0.208020050 0.367418546 0.240601504 0.138345865 0.033082707
##   1 0.180473373 0.224852071 0.198224852 0.094674556 0.162721893
##    CustServCalls
## Y            5           6           7           8           9
##   0 0.007518797 0.004010025 0.001002506 0.000000000 0.000000000
##   1 0.088757396 0.029585799 0.011834320 0.002958580 0.005917160
##
##    DayMins
## Y       [,1]     [,2]
##   0 174.9678 49.79137
##   1 204.1873 67.49795
##
##    OverageFee
## Y         [,1]     [,2]
##   0  9.985935 2.517011
##   1 10.677692 2.590921
```

```
NBPredictTrain <- predict(NBModel,newdata = train)
```

The model creates the conditional probability for each feature separately.

We also have the a-priori probabilities which indicates the distribution of our data.

Let’s calculate how we perform on the Training data.

Confusion Matrix on Train Dataset

```
NBTrainTable <- table(train$Churn,NBPredictTrain)
print(NBTrainTable)
```

```
##    NBPredictTrain
##       0    1
##   0 1957   38
##   1  281   57
```

```
# Accuracy
NBTrainaccuracy <- round(sum(diag(NBTrainTable))/sum(NBTrainTable),
```

```
2)
print(NBTrainaccuracy)

## [1] 0.86

# Sensitivity
NBTrainsensitivity <- round(57/(57+38),2)
print(NBTrainsensitivity)

## [1] 0.6

# Specificity
NBTrainspecificity <-round(1957/(1957 + 281),2)
print(NBTrainspecificity)

## [1] 0.87
```

Based on Confusion Matrix,

With 86% accuracy on the Training Dataset Our Naive Bayes Model Done well in predicting the 0 (87%) (Customers who haven't churn out) than in Predicting the 1 (60%) (Customers who have Churn out)

Let's Look, how the Naive Bayes Model performs on the Test Data set

```
NBTestPredict <- predict(NBModel,newdata = test)
```

Confusion Matrix on Test Dataset

```
NBTestTable <- table(test$Churn,NBTestPredict)
print(NBTestTable)

##    NBTestPredict
##       0   1
##   0 826  29
##   1 117  28

# Accuracy
NBTestaccuracy <- round(sum(diag(NBTestTable))/sum(NBTestTable),2)
print(NBTestaccuracy)

## [1] 0.85
```

```
# Sensitivity
NBTestsensitivity <- round(28/(28+29),2)
print(NBTestsensitivity)

## [1] 0.49# Specificity
NBTestspecificity <-round(826/(826 + 117),2)
print(NBTestspecificity)## [1] 0.88
```

Based on Confusion Matrix,

With 85% accuracy on the Test Dataset Our Naive Bayes Model Done well in predicting the 0 (88%) (Customers who haven't churn out) than in Predicting the 1 (49%) (Customers who have Churn out)

LOGISTIC REGRESSION vs NAÏVE BAYES vs KNN

PERFORMANCE MEASURES		Model Evaluation				
		Logistic Regression		Naïve Bayes		KNN
		TRAIN	TEST	TRAIN	TEST	TEST
CONFUSION MATRIX	Accuracy	84	84	86	85	**90**
	Sensitivity (1)	46	46	60	49	**85**
	Specificity (0)	90	90	87	88	**90**
AUC		66	66	-	-	-
KS		34	33	-	-	-
GINI		87	87	-	-	-

- The above table clearly shows that the Logistic Model Ranks the Lowest when Compared to Naïve Bayes and KNN.
- Though Logistic Regression did a Great Job in predicting the customers who won't churn out. It did a Pretty Bad Job in Predicting the customers who will churn out.
- On the Other Hand, Naïve Bayes was performed mediocre in predicting the Customers who will churn out.
- In the End, the one model which performed well in predicting both (Customers who will cancel / who won't cancel our service) is **KNN**

CONCLUSION

Customer Churn is a burning problem for Telecom companies. In this project, we had analyzed one such case of customer churn where we worked on a data of postpaid customers with a contract.

We did analysis on the customer usage behavior, contract details and the payment details. Then Based on this past data, we build a Multiple Classification models which can predict whether a customer will cancel their service in the future or not.

In the end based on the Performance, we decided that **KNN** algorithm did well in predicting whether a customer will cancel their service in the Future or not.

I would Recommend using KNN Model to Predict whether a customer will cancel their service in the Future or not.