Mini – Project

Report on Predictive Modeling

Report By

Chakradhar Varma

Table of Contents

1. Project Objective

The objective of the project is to build logistic regression model on the given data. The model is to be built on the data-set in order to predict if the customer is churned. Model Performance is to be measured on the Development sample and should be validated on the Holdout sample.

2. Solution

This Solution section will explain each part of the project in the following steps:

1. Preliminary and Exploratory Data Analysis.
2. Logistic Regression Model.

The Source code for the above steps is attached in the Appendix A Section.

2.1 Preliminary and Exploratory Data Analysis

The data provided is Cellphone customers churned data. In the given data set there are total of 3333 observations with 11 variables of interest. It is observed that data has features of numeric class. These features are basically providing the usage, consumption rate and customer service information of the customers. The data has been tested for any missing values and found out that there is no missing values in the sample.

As part of the Exploratory Data Analysis, each variable is summarized and descriptive statistics is performed to check the normality. Initially Filter Method is applied as a feature selection technique to see if any feature can be eliminated. Correlation between the variables is checked to see if there is any multicollinearity. It is observed that DataPlan and DataUsage features are highly correlated and MonthlyCharge variable is significantly correlated with DataPlan, DataUsage and DayMins.

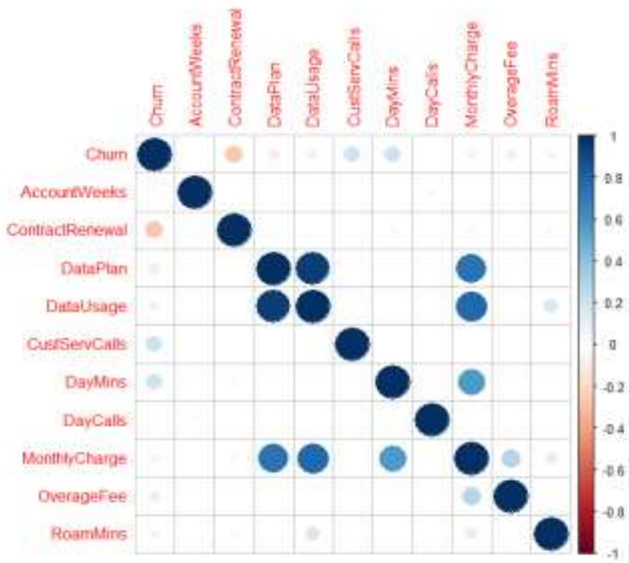


Figure 2.1 Correlation Map across all variables

The features are validated for selection using wrapper method i.e., Boruta Algorithm. The Boruta Algorithm confirms two variables to be unimportant and they are Accountweeks and Dailycalls. It is observed that the percentage of customers churned (**85.50**%) is way more than the customers not churned (**14.5**%). This implies that, data might need to be balanced in order to achieve good classification results.

2.2 Logistic Regression Model

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

Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. The sigmoid function returns values from 0 to 1. For the classification task, we need a discrete output of 0 or 1. To convert a continuous flow into discrete value, generally we set a decision bound at 0.5. All values above this threshold are classified as 1.

2.2.1 Data Partition

The given data set is divided into Training and Validation data set, with approximately 70:30 proportion using random variable. The distribution of Churned and Non Churned Class is verified in both the data sets, and ensured it's close to equal.

	No. of Observations	No. of Customers not Churned	No. of Customers Churned	% of Customers not Churned	% of Customers Churned
Dev Sample	2309	1972	337	85.40	14.6
Validation Sample	1024	878	146	85.74	14.26

2.2.2 Model Building

Initial logarithmic regression model is built on the training sample using all features. Later the insignificant variables are removed and built again. The variables MonthlyCharge, DayCalls, AccountWeeks and DataUsage are come out to be insignificant as it also observed from preliminary analysis. After the final model is built, it is used to predict the class and score the predicted values and are added to the new columns in the data set. The observations obtained are presented below.

```
Call:
glm(formula = CP_Logit.eq.final, family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0498  -0.5038  -0.3322  -0.1963   3.0438
```

```
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)     -5.693846   0.530694 -10.729  < 2e-16 ***
ContractRenewal -2.122021   0.177300 -11.969  < 2e-16 ***
DataPlan        -0.978886   0.172763  -5.666 1.46e-08 ***
CustServCalls    0.587100   0.047636  12.325  < 2e-16 ***
DayMins          0.012977   0.001321   9.820  < 2e-16 ***
OverageFee       0.150465   0.027605   5.451 5.02e-08 ***
RoamMins         0.081776   0.025140   3.253  0.00114 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1919.3  on 2308  degrees of freedom
Residual deviance: 1485.3  on 2302  degrees of freedom
AIC: 1499.3

Number of Fisher Scoring iterations: 6
```

Data 2.2.2 – 1: Logit Model Output

```
ContractRenewal   DataPlan   CustServCalls   DayMins   OverageFee   RoamMins
1.081388          1.031768   1.110680        1.055105  1.029217     1.012034
```

Data 2.2.2 – 2: Variance Inflation Factor

The VIF for all the predictors are around value of 1 which indicates that there is no multicollinearity and the variance is not inflated at all.

2.2.3 Model Performance

The following model performance measures are calculated on the development set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

To predict the scores, 0.5 p-value is used as the margin to convert the probabilities into binary values. Based on the predicted score on the data set, it is assigned with the deciles. Rank table is obtained by executing the ranking code upon the data set. The response rate in top decile is around **50%** and in top three deciles it is **77%**. The KS is around **55%**, indicating that the model is a good model with scope for improvement.

The Rank table, Performance plot and the confusion matrix obtained from the model are presented below.

Deciles	cnt	cnt_resp	cnt_non_resp	rrate	cum_resp	cum_non_resp	cum_perct_resp	cum_perct_non_resp	ks
10	231	114	117	49.35	114	117	33.83	5.93	27.9
9	231	92	139	39.83	206	256	61.13	12.98	48.15
8	231	53	178	22.94	259	434	76.85	22.01	54.84
7	231	29	202	12.55	288	636	85.46	32.25	53.21
6	231	16	215	6.93	304	851	90.21	43.15	47.06
5	230	9	221	3.91	313	1072	92.88	54.36	38.52
4	231	5	226	2.16	318	1298	94.36	65.82	28.54
3	231	6	225	2.6	324	1523	96.14	77.23	18.91
2	231	10	221	4.33	334	1744	99.11	88.44	10.67
1	231	3	228	1.3	337	1972	100	100	0

Table 2.2.3-1: Rank Table of the development sample

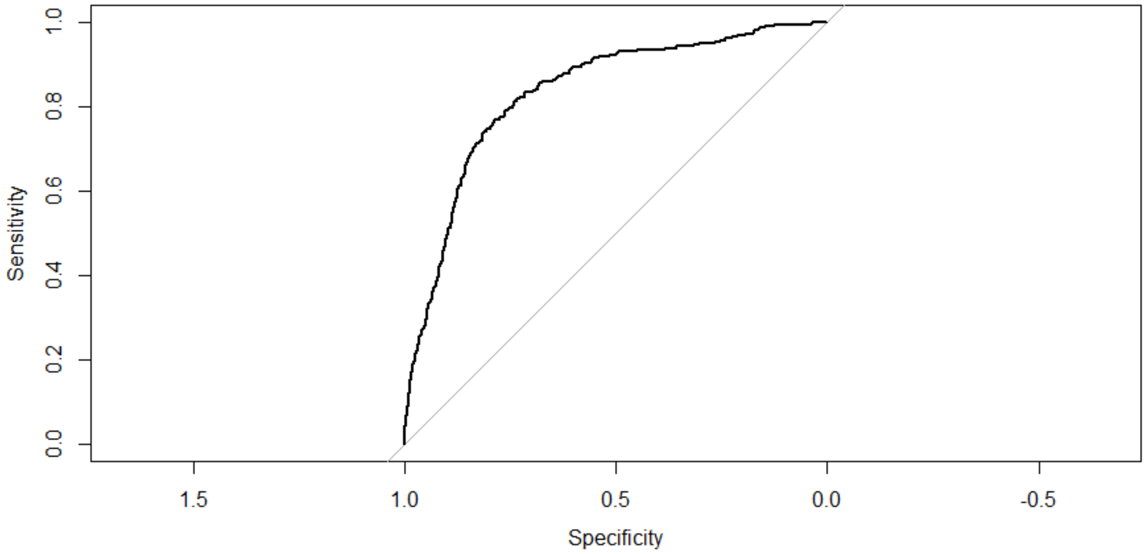


Figure 2.2.3.1: ROC Curve for the Model

The Area under curve (AUC) of **83.16 %** and Gini coefficient of **54.40 %** are obtained from the model on the development sample and they also indicate that, model is good. The Accuracy and Classification error rate are as below:

	predict.class	
TARGET	0	1
0	1922	50
1	264	73

Accuracy = (1922+73)/2309 = 0.8640

Classification Error Rate = 1 – Accuracy = 0.13598

```
Confusion Matrix and Statistics

      0    1
 0 1922   50
 1  264   73

               Accuracy : 0.864
                 95% CI : (0.8494, 0.8777)
    No Information Rate : 0.9467
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2596
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8792
            Specificity : 0.5935
         Pos Pred Value : 0.9746
         Neg Pred Value : 0.2166
             Prevalence : 0.9467
         Detection Rate : 0.8324
   Detection Prevalence : 0.8540
      Balanced Accuracy : 0.7364

       'Positive' Class : 0
```

Figure 2.2.3.2: Confusion Matrix of the Training Data

Now the model is used to predict on the validation sample and the observations are as follows:

deciles	cnt	cnt_resp	cnt_non_resp	rrate	cum_resp	cum_non_resp	cum_perct_resp	cum_perct_non_resp	ks
10	103	46	57	44.66	46	57	31.51	6.49	25.02
9	102	34	68	33.33	80	125	54.79	14.24	40.55
8	102	22	80	21.57	102	205	69.86	23.35	46.51
7	103	11	92	10.68	113	297	77.4	33.83	43.57
6	102	10	92	9.8	123	389	84.25	44.31	39.94
5	102	8	94	7.84	131	483	89.73	55.01	34.72
4	103	8	95	7.77	139	578	95.21	65.83	29.38
3	102	2	100	1.96	141	678	96.58	77.22	19.36
2	102	4	98	3.92	145	776	99.32	88.38	10.94
1	103	1	102	0.97	146	878	100	100	0

Table 2.2.3-2: Rank Table of the development sample

The response rate in top decile is around **45%** and in top three deciles it is **70%**. The KS is around **47%**, indicating that the model is a good model.

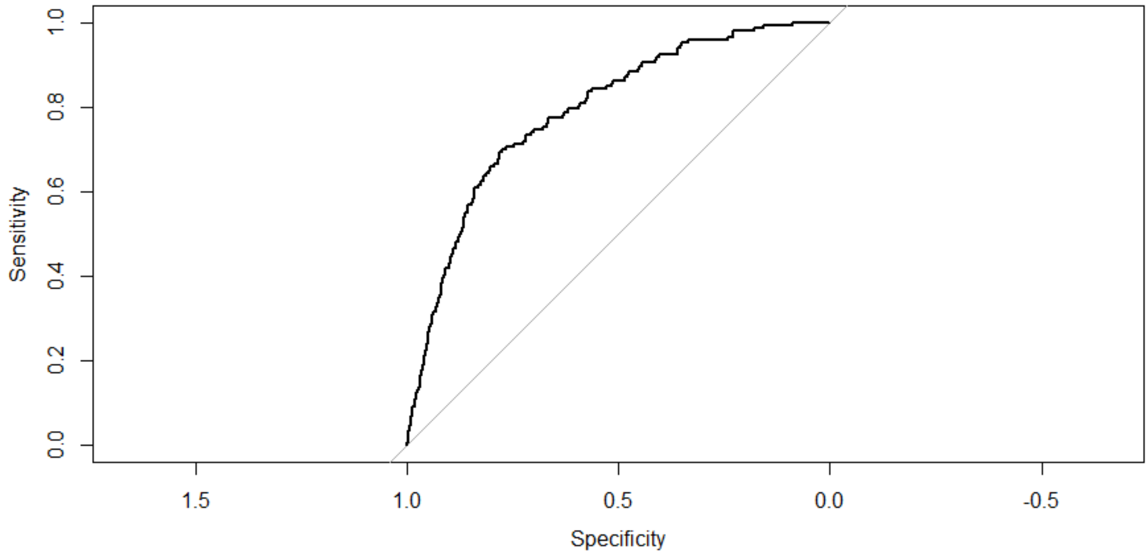


Figure 2.2.3.3: ROC Curve on the Testing Data

The Area under curve (AUC) of **79 %** and Gini coefficient of **54.34 %** are obtained from the model on the validation sample and they also indicate that, model is good. The Accuracy and Classification error rate are as below:

	predict.class	
TARGET	0	1
0	838	40
1	113	33

Accuracy = (838+40)/1024 = 0.8506

Classification Error Rate = 1 – Accuracy = 0.1494

```
Confusion Matrix and Statistics


      0   1
  0 838  40
  1 113  33

               Accuracy : 0.8506
                 95% CI : (0.8273, 0.8719)
    No Information Rate : 0.9287
    P-Value [Acc > NIR] : 1

                  Kappa : 0.228
 Mcnemar's Test P-Value : 5.855e-09

            Sensitivity : 0.8812
            Specificity : 0.4521
         Pos Pred Value : 0.9544
         Neg Pred Value : 0.2260
             Prevalence : 0.9287
         Detection Rate : 0.8184
   Detection Prevalence : 0.8574
      Balanced Accuracy : 0.6666

       'Positive' Class : 0
```

As the difference of the performance measure values of development sample and the validation sample are within the tolerance range of 5-10%, it is clear that model is not overfit model. As the model performance measures indicates the Logistic Regression model is good model, to improve the performance the data can be balanced and the model could be build.

2.2.4 Conclusion

The Logistic Regression Model built on the given data set has come out to be good model and also not a overfit model as the difference of performance measures on training and validation sample are under the tolerance limit. As stated model can be built on a balanced dataset to see if performance can be increased but the current model is sufficiently good to predict on any new data.

3. Conclusion

The main objective of the project was to develop a predictive model to predict if cellphone customers will cancel the service (churn) or not. In this context, the key parameter for model evaluation was 'Accuracy', i.e., the proportion of the total number of predictions that were correct (i.e. % of the customers that were correctly predicted).

The Logistic Regression Model technique is performed as a predictive model. The Performance measures are come out to be good with scope for improvement. Other predictive models like Random Forest may also perform better and also by **adjusting the cutoff p-value,** model performance can be monitored for improvement. The overview of the performance of the model on accuracy, over-fitting and other model performance measures as well as Odds Model output is provided below:

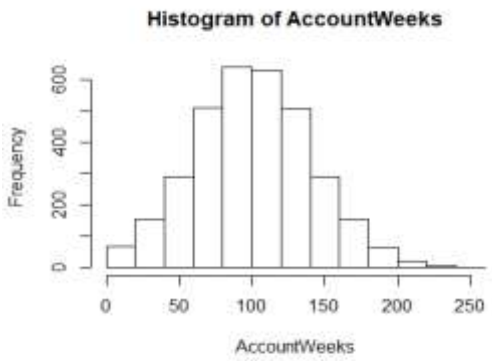
Measures	Train	Test	%Deviation
KS	55%	47%	8%
AUC	83.16%	79%	4.16%
Gini	54.40%	54.34%	0.14%
Accuracy	86.40%	85.06%	1.34%
CeR	13.6%	14.94%	1.34%

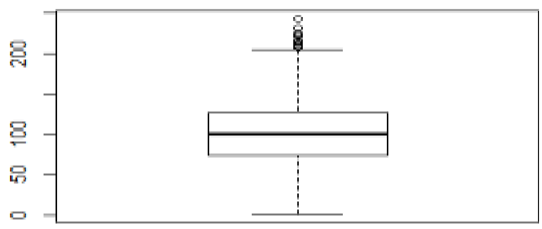
```
ODD Model: Odds for all the independent variables

ContractRenewal    DataPlan   CustServCalls  DayMins   OverageFee  RoamMins
   0.119789        0.375729      1.798765    1.013061   1.162374   1.085212
```

4. Appendix A – Source Code

```
#set up of working directory
setwd("D:/BACP Program/R Directory")

#Libraries

library(readxl)

library(car)

library(forecast)

library(corrplot)

library(ggplot2)

library(randomForest)

library(caret)

library(moments)

library(ROCR)

library(ineq)

#preliminary Analysis

##Importing the cellphone data
CP_data <- read_excel("Cellphone.xlsx",sheet="Data")

#View(CP_data)
attach(CP_data)

#dimension of the data set
dim(CP_data)

## [1] 3333   11

#Viewing the structure of the data (data types- class)
str(CP_data)

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

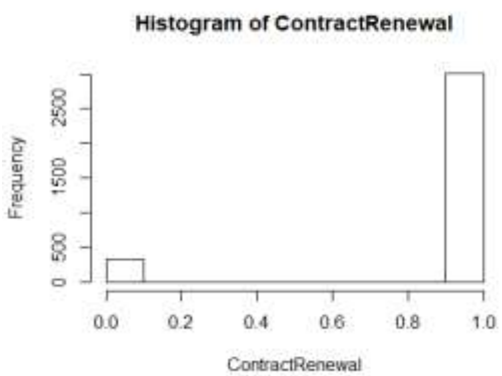
```
##  $ OverageFee      : num  9.87 9.78 6.06 3.1 7.42 ...
##  $ RoamMins        : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

#All columns are observed to be Numeric

#Summary of the data
##Total 0f 3333 observations with 11 variables

#check top 10 and bottom 10 observations
#head(CP_data)
#tail(CP_data)

#checking for missing values
colSums(is.na(CP_data))

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

#No missing values in the data set

#Exploratory Data Analysis - Checking individual columns

#Chrun variable
summary(factor(Churn))

##    0    1
## 2850  483

#2850 customers did not cancelled the service and 483 customers cancelled.

prop.table(table(Churn))

## Churn
##         0         1
## 0.8550855 0.1449145

#85.50 % customers are availing the service.
#14.5 % customers have cancelled service.

#AccountWeeks

summary(AccountWeeks)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    74.0   101.0   101.1   127.0   243.0

#The minimum usage is 1 week, while the maximum activation period is 243 w
eeks.
hist(AccountWeeks)
```





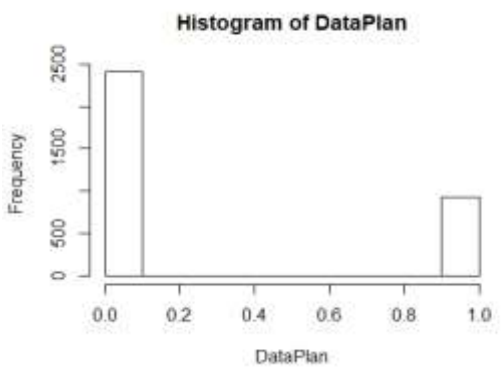

```
#Between these two values, the AccountWeeks looks to be uniformly spread.
boxplot(AccountWeeks)
```

```
#Normal with few outliers
BoxCox.lambda(AccountWeeks)

## [1] 0.7972743

#~0.8, No need of transformation

#ContractRenewal

summary(factor(ContractRenewal))

##    0    1
##  323 3010

#More number of customers have recently renewed around 3010.
prop.table(table(ContractRenewal))

## ContractRenewal
##          0          1
## 0.09690969 0.90309031

#90.31% customers have renewed contract and 9.69% have not renewed.
hist(ContractRenewal)
```



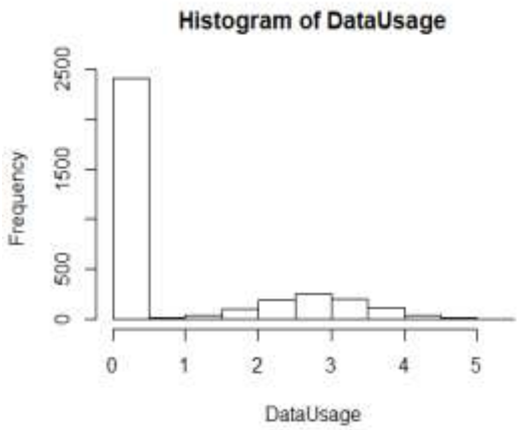

```
chisq.test(Churn, ContractRenewal)
```

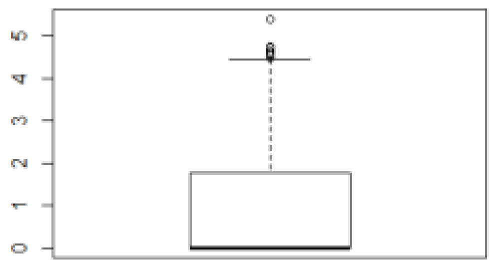
```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Churn and ContractRenewal
## X-squared = 222.57, df = 1, p-value < 2.2e-16

#Contract renewal is significant

#DataPlan

summary(factor(DataPlan))

##    0    1
## 2411  922

#More number of customers doesn't opt for data plan.
prop.table(table(DataPlan))

## DataPlan
##         0         1
## 0.7233723 0.2766277

#72.33% customers don't have data plan and 27.67% have data plan.
hist(DataPlan)
```



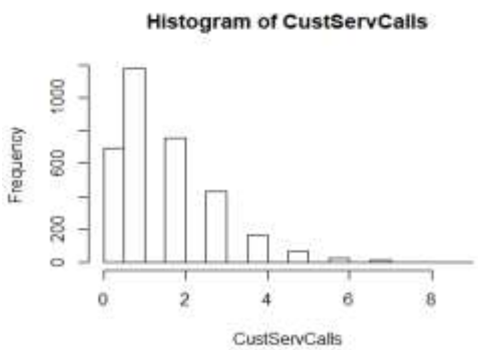

```
chisq.test(Churn, DataPlan)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Churn and DataPlan
## X-squared = 34.132, df = 1, p-value = 5.151e-09

#DataPlan is significant


#DataUsage

summary(DataUsage)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.8165  1.7800  5.4000
```

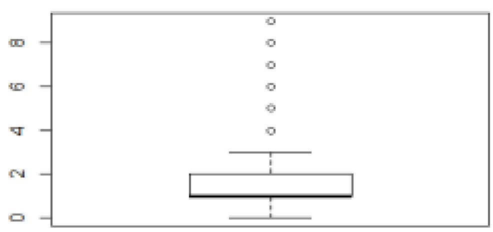
```
#The minimum usage is 0 gb per month, while the max usage is 5.4 gb.
hist(DataUsage)
```





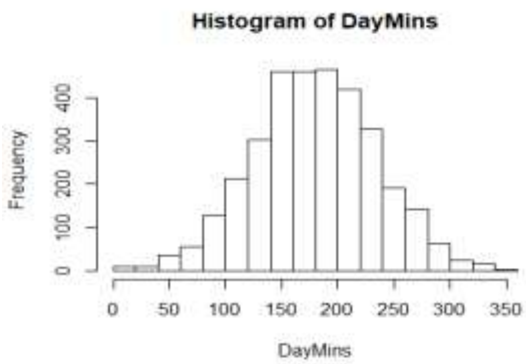

```
#It is evident, as 72.3% customers don't have data plan, there would be no
usage.
boxplot(DataUsage)

#Not Normal
BoxCox.lambda(DataUsage)

## [1] 0.03243403

#~0.03, log transformation can be applied
#CP_data$DataUsage <- ifelse(CP_data$DataUsage == 0, 0, log(CP_data$DataUs
age))


#CustServCalls

summary(CustServCalls)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   1.563   2.000   9.000

#The minimum calls to customer service is 0, while the maximum call count
9.
hist(CustServCalls)
```

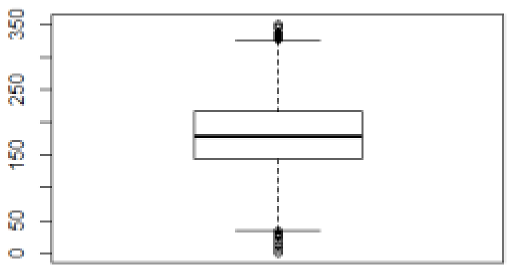




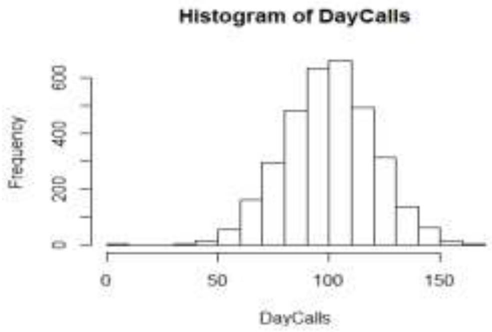

```
boxplot(CustServCalls)
```

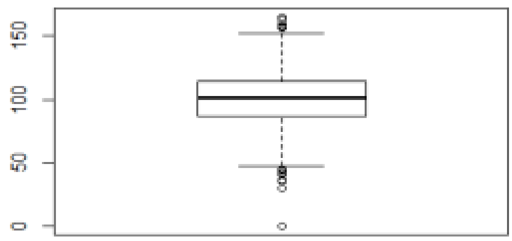
```
#Not Normal - skewed
BoxCox.lambda(CustServCalls)

## [1] 0.3569469

#~0.35, sqrt transformation can be applied
#CP_data$CustServCalls <- sqrt(CP_data$CustServCalls)

#DayMins - average daytime minutes per month

summary(DayMins)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   143.7   179.4   179.8   216.4   350.8

#The minimum avg daytime minutes is 0, while the maximum are 350.8.
hist(DayMins)
```





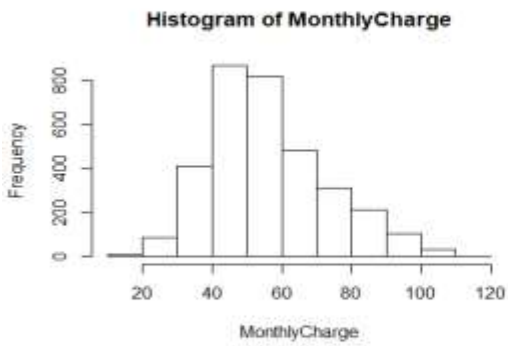

```
boxplot(DayMins)

#Looks Normal - outliers on either side
BoxCox.lambda(DayMins)

## [1] 1.052307

#~1.05, No transformation is required

#DayCalls - average number of daytime calls

summary(DayCalls)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    87.0   101.0   100.4   114.0   165.0

#The minimum avg daytime calls is 0, while the maximum are 165.
hist(DayCalls)
```

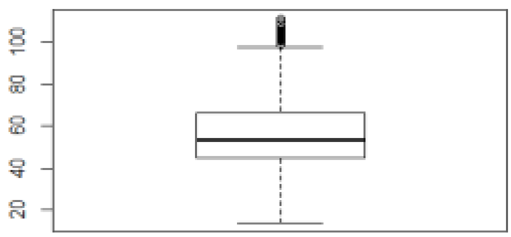





```r
boxplot(DayCalls)
```

```r
#looks Normal - outliers on either side
BoxCox.lambda(DayCalls)

## [1] 1.157052

#~1.15, No transformation is required

#MonthlyCharge - average monthly bill

summary(MonthlyCharge)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.00   45.00   53.50   56.31   66.20  111.30

#The minimum monthly charge is 14, while the maximum is 111.30.
hist(MonthlyCharge)
```

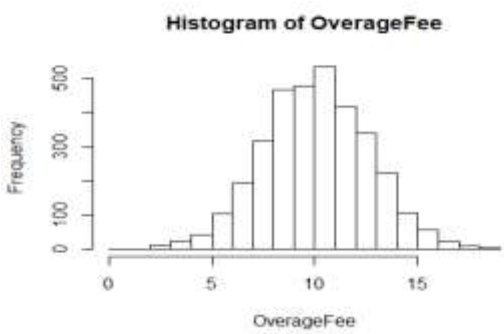


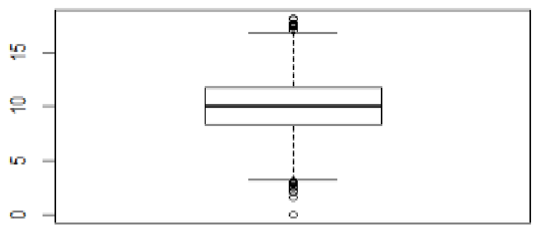



```r
boxplot(MonthlyCharge)
```

```r
#looks Normal with few outliers
BoxCox.lambda(MonthlyCharge)

## [1] -0.01038151
```

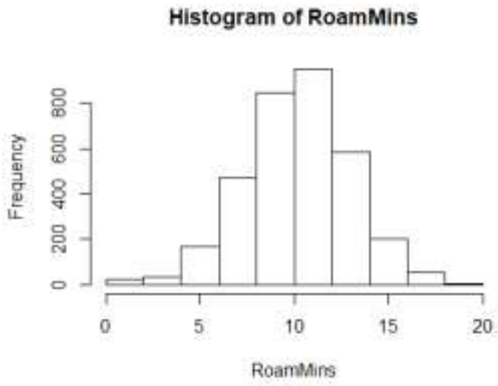
```r
#~-0.01, log transformation can be applied
#CP_data$MonthlyCharge <- ifelse(CP_data$MonthlyCharge == 0, 0, log(CP_dat
a$MonthlyCharge))

#OverageFee - largest overage fee in last 12 months

summary(OverageFee)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    8.33   10.07   10.05   11.77   18.19

#The minimum overagefee is 8.33, while the maximum is 18.19.
hist(OverageFee)
```

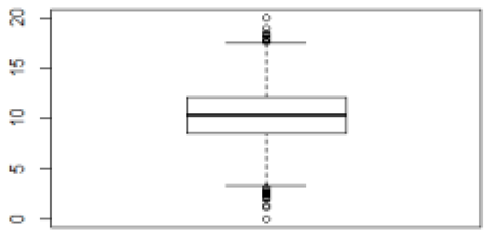





```r
boxplot(OverageFee)
```

```r
#Looks Normal with few outliers
BoxCox.lambda(OverageFee)

## [1] 1.072484

#~1.07, No transformation is required

#RoamMins - average number of roaming minutes

summary(RoamMins)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    8.50   10.30   10.24   12.10   20.00

#The minimum RoamMins is 8.50, while the maximum is 20.00.
hist(RoamMins)
```

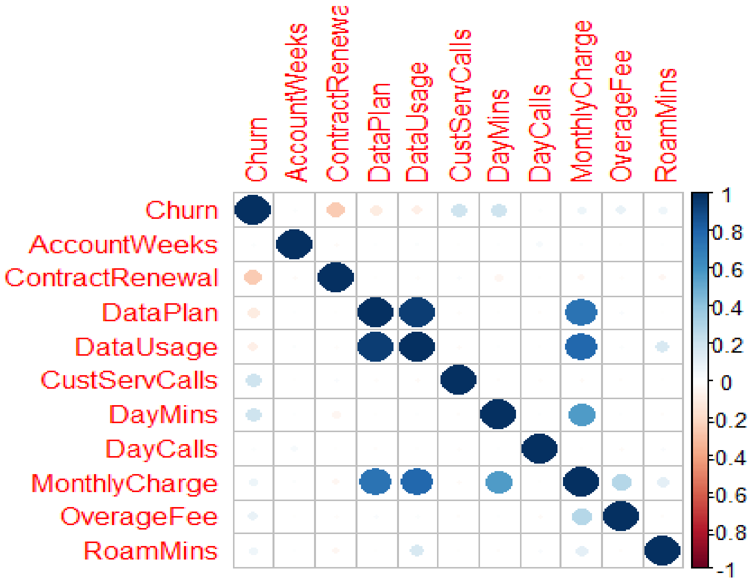





```
boxplot(RoamMins)
```

```
#looks Normal with few outliers
BoxCox.lambda(RoamMins)

## [1] 1.320654

#~1.32, No transformation is required


#Validating with wrapper method
library(Boruta)

## Warning: package 'Boruta' was built under R version 3.5.3

## Loading required package: ranger

## Warning: package 'ranger' was built under R version 3.5.3

##
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':
##
##     importance

#Feature Selection (Wrapper Method)
set.seed(123)
boruta.train <- Boruta(Churn~. ,data=CP_data, doTrace = 2)

##  1. run of importance source...

##  2. run of importance source...

##  3. run of importance source...

##  4. run of importance source...

##  5. run of importance source...

##  6. run of importance source...

##  7. run of importance source...
```

```
##  8. run of importance source...

##  9. run of importance source...

##  10. run of importance source...

## After 10 iterations, +5.3 secs:

##  confirmed 8 attributes: ContractRenewal, CustServCalls, DataPlan, Data
Usage, DayMins and 3 more;

##  rejected 1 attribute: AccountWeeks;

##  still have 1 attribute left.

##  11. run of importance source...

##  12. run of importance source...

##  13. run of importance source...

##  14. run of importance source...

##  15. run of importance source...

##  16. run of importance source...

##  17. run of importance source...

##  18. run of importance source...

## After 18 iterations, +10 secs:

##  rejected 1 attribute: DayCalls;

##  no more attributes left.

print(boruta.train)

## Boruta performed 18 iterations in 10.11096 secs.
##  8 attributes confirmed important: ContractRenewal, CustServCalls,
## DataPlan, DataUsage, DayMins and 3 more;
##  2 attributes confirmed unimportant: AccountWeeks, DayCalls;

#Boruta method has confirmed 8 variables to be important and Accountweeks
& DayCalls are unimportant


#correlations plot

correlations<- cor(CP_data)
corrplot(correlations, method="circle")
```




```
#MonthlyCharge is significantly correlated with DataPlan, DataUsage, and D
ayMins
#DataPlan and DataUsage are also correlated

#so, we can remove MonthlyCharge and DataUsage.

#Partitioning Data Set
CP <- CP_data

CP$random <- runif(nrow(CP_data),0,1)
#Adding these randomly generated numbers to the data as a new column
CP <- CP[order(CP$random),]
#Splitting the data into dev and testing sample based on the random number
train <- CP[which(CP$random <= 0.7),]
val <- CP[which(CP$random > 0.7),]


sum(train$Churn)

## [1] 336

sum(val$Churn)

## [1] 147

dim(train)

## [1] 2327   12

summary(as.factor(train$Churn))

##    0    1
## 1991  336

dim(val)

## [1] 1006   12

summary(as.factor(val$Churn))
```

```
##   0   1
## 859 147

prop.table(table(train$Churn))

##
##         0         1
## 0.8556081 0.1443919

prop.table(table(val$Churn))

##
##         0         1
## 0.8538767 0.1461233

train <- train[-12]
val <- val[-12]

#Logistic Regression

#install.packages(c("SDMTools","pROC", "Hmisc"))
library(SDMTools)

## Warning: package 'SDMTools' was built under R version 3.5.3

##
## Attaching package: 'SDMTools'

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

## The following object is masked from 'package:forecast':
##
##     accuracy

library(pROC)

## Warning: package 'pROC' was built under R version 3.5.3

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

##
## Attaching package: 'pROC'

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

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

library(Hmisc)

## Warning: package 'Hmisc' was built under R version 3.5.3

## Loading required package: survival
```

```
##
## Attaching package: 'survival'

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

## Loading required package: Formula

## Warning: package 'Formula' was built under R version 3.5.2

##
## Attaching package: 'Hmisc'

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

```
# Fit the Sigmoid function

#All Variables
CP_Logit.eq<-as.factor(Churn) ~ .

CP_Logit <- glm(formula = as.factor(train$Churn) ~ .  , train, family = bi
nomial)
summary(CP_Logit)
```

```
##
## Call:
## glm(formula = as.factor(train$Churn) ~ ., family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0393  -0.5017  -0.3380  -0.2053   2.9826
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.1893655  0.6495915  -7.989 1.36e-15 ***
## AccountWeeks     0.0006182  0.0016458   0.376  0.70718
## ContractRenewal -2.0903609  0.1772127 -11.796  < 2e-16 ***
## DataPlan        -0.7457900  0.6424158  -1.161  0.24568
## DataUsage        2.1090037  2.3258704   0.907  0.36453
## CustServCalls    0.5382132  0.0470809  11.432  < 2e-16 ***
## DayMins          0.0496097  0.0392072   1.265  0.20576
## DayCalls        -0.0026040  0.0032866  -0.792  0.42818
## MonthlyCharge   -0.2130119  0.2304097  -0.924  0.35523
## OverageFee       0.4814800  0.3931611   1.225  0.22071
## RoamMins         0.0861267  0.0267712   3.217  0.00129 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1921.4  on 2326  degrees of freedom
## Residual deviance: 1503.6  on 2316  degrees of freedom
```

```
## AIC: 1525.6
##
## Number of Fisher Scoring iterations: 6
```

```
vif(CP_Logit)
```

```
##    AccountWeeks ContractRenewal        DataPlan       DataUsage
##        1.003704        1.064849       14.302948     1647.227865
##    CustServCalls          DayMins        DayCalls   MonthlyCharge
##        1.094337      923.587110        1.005759     2819.601870
##      OverageFee        RoamMins
##      209.469593        1.203319
```

```
#removed MonthlyCharge, DayCalls, AccountWeeks and DataUsage
#####Retain only significant ones
CP_Logit.eq.final<-as.factor(train$Churn)  ~ ContractRenewal+DataPlan+Cust
ServCalls+DayMins+OverageFee+RoamMins
CP_Logit.final <- glm(CP_Logit.eq.final   , train, family = binomial)
summary(CP_Logit.final)
```

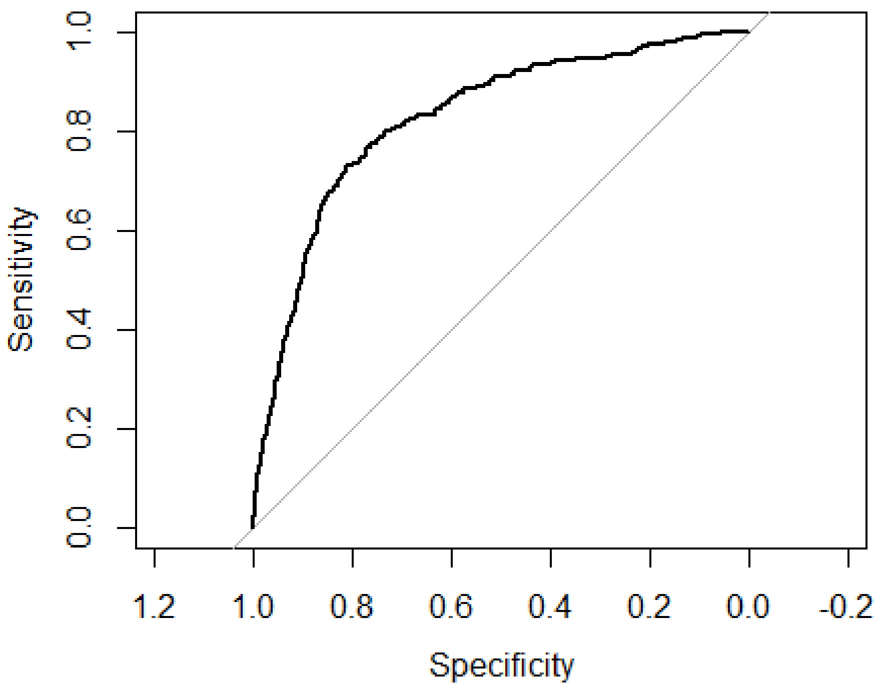
```
##
## Call:
## glm(formula = CP_Logit.eq.final, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0565  -0.5013  -0.3387  -0.2050   2.9607
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -5.465976   0.522786 -10.455  < 2e-16 ***
## ContractRenewal -2.097363   0.176441 -11.887  < 2e-16 ***
## DataPlan        -0.805363   0.170970  -4.711 2.47e-06 ***
## CustServCalls    0.538507   0.046936  11.473  < 2e-16 ***
## DayMins          0.013385   0.001317  10.163  < 2e-16 ***
## OverageFee       0.119304   0.027439   4.348 1.37e-05 ***
## RoamMins         0.084528   0.024510   3.449 0.000563 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1921.4  on 2326  degrees of freedom
## Residual deviance: 1505.3  on 2320  degrees of freedom
## AIC: 1519.3
##
## Number of Fisher Scoring iterations: 5
```

```
vif(CP_Logit.final)
```

```
## ContractRenewal        DataPlan   CustServCalls         DayMins
##        1.058714        1.012337        1.089263        1.043739
##      OverageFee        RoamMins
##        1.023101        1.009443
```

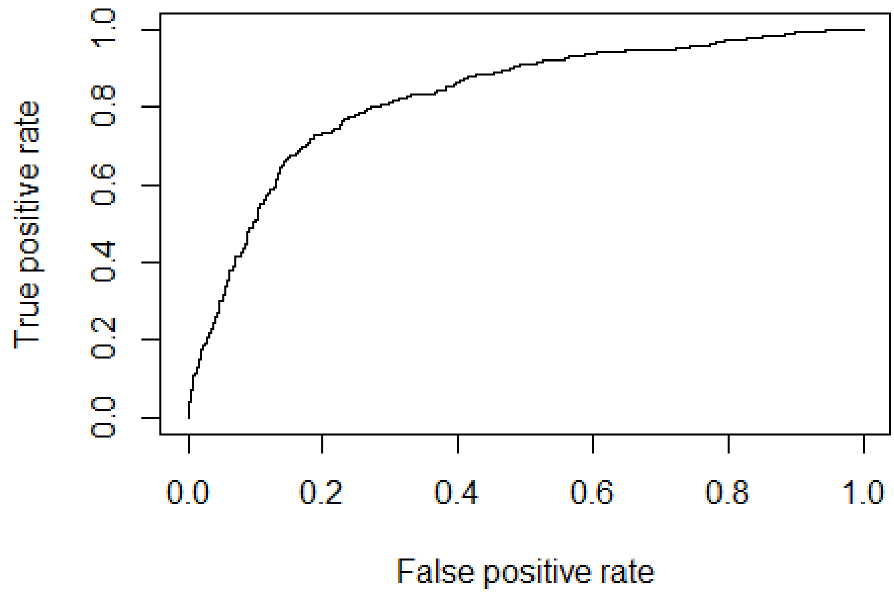
```r
#Classification

#Train Data
pred.logit.final <- predict.glm(CP_Logit.final, newdata=train, type="respo
nse")

tab.LR.imp = data.frame(Target = train$Churn, Prediction = pred.logit.fina
l )
tab.LR.imp$Classification = ifelse(tab.LR.imp$Prediction>0.5,1,0)
with(tab.LR.imp, table(Target, Classification))

##       Classification
## Target    0    1
##      0 1935   56
##      1  267   69

confusionMatrix(table(tab.LR.imp$Target, tab.LR.imp$Classification))

## Confusion Matrix and Statistics
##
##
##         0    1
##    0 1935   56
##    1  267   69
##
##                Accuracy : 0.8612
##                  95% CI : (0.8465, 0.875)
##     No Information Rate : 0.9463
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2398
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8787
##             Specificity : 0.5520
##          Pos Pred Value : 0.9719
##          Neg Pred Value : 0.2054
##              Prevalence : 0.9463
##          Detection Rate : 0.8315
##    Detection Prevalence : 0.8556
##       Balanced Accuracy : 0.7154
##
##        'Positive' Class : 0
##

accuracy.logit<- roc.logit<-roc(train$Churn,pred.logit.final )
roc.logit

##
## Call:
## roc.default(response = train$Churn, predictor = pred.logit.final)
##
## Data: pred.logit.final in 1991 controls (train$Churn 0) < 336 cases (tr
ain$Churn 1).
## Area under the curve: 0.8248
```

```
plot(roc.logit)
```



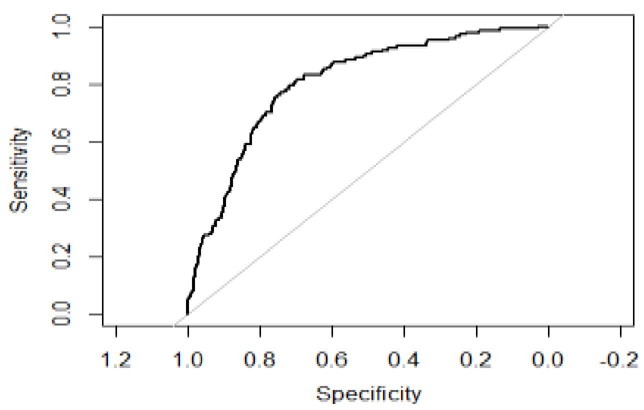

```
tab.LR.imp$Target<- as.character(tab.LR.imp$Target)
tab.LR.imp$Target[tab.LR.imp$Target == "0"] <- 0
tab.LR.imp$Target[tab.LR.imp$Target== "1"] <- 1

tab.LR.imp$Target <- as.numeric(tab.LR.imp$Target)
#Deciling
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8],
8,
                                                            ifelse(x<decil
es[9], 9, 10
                                                            ))))))))))
}

#Assigning deciles to the data
tab.LR.imp$deciles <- decile(tab.LR.imp$Prediction)
```

```r
##Ranking the data
library(data.table)

## Warning: package 'data.table' was built under R version 3.5.2

#Creating rank table
tmp_DT = data.table(tab.LR.imp)
rank <- tmp_DT[, list(
  cnt = length(Target),
  cnt_resp = sum(Target),
  cnt_non_resp = sum(Target == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_resp * 100 / rank$cnt,2);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_perct_resp <- round(rank$cum_resp * 100 / sum(rank$cnt_resp),2);
rank$cum_perct_non_resp <- round(rank$cum_non_resp * 100 / sum(rank$cnt_no
n_resp),2);
rank$ks <- abs(rank$cum_perct_resp - rank$cum_perct_non_resp);
rank

##     deciles cnt cnt_resp cnt_non_resp rrate cum_resp cum_non_resp
##  1:      10 233      116          117 49.79      116          117
##  2:       9 233       90          143 38.63      206          260
##  3:       8 232       45          187 19.40      251          447
##  4:       7 233       27          206 11.59      278          653
##  5:       6 233       19          214  8.15      297          867
##  6:       5 232       13          219  5.60      310         1086
##  7:       4 233        8          225  3.43      318         1311
##  8:       3 232        5          227  2.16      323         1538
##  9:       2 233        9          224  3.86      332         1762
## 10:       1 233        4          229  1.72      336         1991
##     cum_perct_resp cum_perct_non_resp    ks
##  1:          34.52               5.88 28.64
##  2:          61.31              13.06 48.25
##  3:          74.70              22.45 52.25
##  4:          82.74              32.80 49.94
##  5:          88.39              43.55 44.84
##  6:          92.26              54.55 37.71
##  7:          94.64              65.85 28.79
##  8:          96.13              77.25 18.88
##  9:          98.81              88.50 10.31
## 10:         100.00             100.00  0.00

#ks ===> 49.76

pred <- prediction(tab.LR.imp$Prediction, tab.LR.imp$Target)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```




```
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(tab.LR.imp$Prediction, type="Gini")

auc

## [1] 0.82478

KS

## [1] 0.5428281

gini

## [1] 0.5346071

#Testing Data

pred.logit.final <- predict.glm(CP_Logit.final, newdata=val, type="respons
e")

tab.LR.imp = data.frame(Target = val$Churn, Prediction = pred.logit.final
)
tab.LR.imp$Classification = ifelse(tab.LR.imp$Prediction>0.5,1,0)
with(tab.LR.imp, table(Target, Classification))

##         Classification
## Target    0   1
##        0 831  28
##        1 114  33
```

```r
confusionMatrix(table(tab.LR.imp$Target, tab.LR.imp$Classification))

## Confusion Matrix and Statistics
##
##
##        0   1
##    0 831  28
##    1 114  33
##
##                Accuracy : 0.8588
##                  95% CI : (0.8358, 0.8798)
##     No Information Rate : 0.9394
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2533
##  Mcnemar's Test P-Value : 9.817e-13
##
##             Sensitivity : 0.8794
##             Specificity : 0.5410
##          Pos Pred Value : 0.9674
##          Neg Pred Value : 0.2245
##              Prevalence : 0.9394
##          Detection Rate : 0.8260
##    Detection Prevalence : 0.8539
##       Balanced Accuracy : 0.7102
##
##        'Positive' Class : 0
##

accuracy.logit<- roc.logit<-roc(val$Churn,pred.logit.final )
roc.logit

##
## Call:
## roc.default(response = val$Churn, predictor = pred.logit.final)
##
## Data: pred.logit.final in 859 controls (val$Churn 0) < 147 cases (val$C
hurn 1).
## Area under the curve: 0.8056

plot(roc.logit)
```



```
tab.LR.imp$Target<- as.character(tab.LR.imp$Target)
tab.LR.imp$Target[tab.LR.imp$Target == "0"] <- 0
tab.LR.imp$Target[tab.LR.imp$Target== "1"] <- 1

tab.LR.imp$Target <- as.numeric(tab.LR.imp$Target)
#Deciling
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8],
8,
                                                            ifelse(x<decil
es[9], 9, 10
                                                            ))))))))))
}

#Assigning deciles to the data
tab.LR.imp$deciles <- decile(tab.LR.imp$Prediction)

#Creating rank table
tmp_DT = data.table(tab.LR.imp)
rank <- tmp_DT[, list(
  cnt = length(Target),
  cnt_resp = sum(Target),
  cnt_non_resp = sum(Target == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_resp * 100 / rank$cnt,2);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_perct_resp <- round(rank$cum_resp * 100 / sum(rank$cnt_resp),2);
rank$cum_perct_non_resp <- round(rank$cum_non_resp * 100 / sum(rank$cnt_no
n_resp),2);
rank$ks <- abs(rank$cum_perct_resp - rank$cum_perct_non_resp);
rank

##     deciles cnt cnt_resp cnt_non_resp rrate cum_resp cum_non_resp
##  1:      10 101       42           59 41.58       42           59
##  2:       9 101       37           64 36.63       79          123
##  3:       8 100       24           76 24.00      103          199
##  4:       7 101       19           82 18.81      122          281
##  5:       6 100        7           93  7.00      129          374
##  6:       5 101        6           95  5.94      135          469
##  7:       4 100        2           98  2.00      137          567
##  8:       3 101        6           95  5.94      143          662
```

```
## 9:        2 100        3           97  3.00       146           759
## 10:       1 101        1          100  0.99       147           859
##     cum_perct_resp cum_perct_non_resp    ks
## 1:           28.57               6.87 21.70
## 2:           53.74              14.32 39.42
## 3:           70.07              23.17 46.90
## 4:           82.99              32.71 50.28
## 5:           87.76              43.54 44.22
## 6:           91.84              54.60 37.24
## 7:           93.20              66.01 27.19
## 8:           97.28              77.07 20.21
## 9:           99.32              88.36 10.96
## 10:         100.00             100.00  0.00
```

```
pred <- prediction(tab.LR.imp$Prediction, tab.LR.imp$Target)
perf <- performance(pred, "tpr", "fpr")
#plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(tab.LR.imp$Prediction, type="Gini")

auc
```

```
## [1] 0.8056117
```

```
KS
```

```
## [1] 0.5159773
```

```
gini
```

```
## [1] 0.5262393
```

```
#Odds for all independent variables
oddModel<-exp(coef(CP_Logit.final))
print(oddModel)
```

```
##     (Intercept) ContractRenewal        DataPlan   CustServCalls
##      0.00422821      0.12277982      0.44692585      1.71344639
##         DayMins      OverageFee        RoamMins
##      1.01347473      1.12671250      1.08820345
```