CUSTOMER CHURN PREDICTION

Done by,

Ranjith P

Contents: **Page.no:**

Chapter 1

Introduction

1.1 Problem Description:

Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving. This problem statement is targeted at enabling churn reduction using analytics concepts.

1.2 Problem Statement:

A Telecom company wants to predict the customer behaviour towards churn. The company would like to predict the customer churn and make them to retain in their company. In order to take the proactive actions to retain the customers, the company need the reasons for customer churn.

1.3 Business Understanding:

Being an Analyst, I understood that the telecom company would like to predict the likelihood of customer churn. And most importantly the company need the major reasons for customer churn in their company. So, my duty is to predict the likelihood of churn for each customer and the major features impacting the churn.

The result of this project will help the telecom company to take proactive actions in order to retain the customer who has high likelihood of churning.

1.4 Data Description:

The telecom sector has provided their customers usage data in order to predict the customer behaviour.

The data is given in two datasets such as Train data & Test data.

Train data have to be used to train the model.

Test data have to be used for prediction and evaluation.

Data Dictionary:

The following are the features in the given datasets and their descriptions:

Features	**Description**
Predictors	
account length	Account length or duration of the customer in this company
international plan	Whether the customer has chosen international plan or not
voicemail plan	Whether the customer has chosen voice mail plan or not
number voicemail messages	No. of voice mail messages sent
total day minutes	Total minutes of usage during the day time by the customer
total day calls	Total calls made during the day time by the customer
total day charge	Total charge for usage during the day time by the customer
total evening minutes	Total minutes of usage during the evening time by the customer
total evening calls	Total calls made during the evening time by the customer
total evening charge	Total charge for usage during the evening time by the customer
Total night minutes	Total minutes of usage during the night time by the customer
Total night calls	Total calls made during the night time by the customer
Total night charge	Total charge for usage during the night time by the customer
Total intl minutes	Total international minutes used
Total intl calls	Total international calls made
Total intl charge	Total charge on international calls
Target	
Churn	Whether the customer has churned or not. (Target Variable) True. – Churn False. – No Churn

Chapter 2

Methodology:

Exploratory Data Analysis

Data Pre-processing:

Exploratory Data Analysis helps us to understand the data better. It will also help us to do necessary data cleaning and data preparations in the pre-processing stage. Most probably the exploratory data analysis and data pre-processing are done together in order to sanitize the data for modelling. The nature of data will help us to decide the methodology of dealing with the data and perform the necessary algorithms.

To start this process, we look at the summary, structure and dimension of data to have a basic understanding on the data. We visualize the data to know the distribution of each features to check the normality of the data. Also, multi-variate visualizations can be done with the features to know the relationship of features. Generally, Data Explorations and Pre-processing includes understanding the data, cleaning the data and visualizing the data as well.

2.1 Missing Value Analysis:

The first step in cleaning the data is detecting the missing values in the data and removing or imputing it. The problem of missing value is common. Missing values in the data can complicate our analysis by creating bias or reducing statistical efficiency. So, we detect the missing values and treat it. Either we remove the missing values or we will impute it through various techniques. In our data, there is no missing value.

2.2 Outlier Detection & Treatment:

Outliers are the extreme values which may skew the data and creates bias in our analysis. Outliers will affect the assumptions and results of our analysis. So, it's better to remove or impute the outliers. We can visualize the outliers using the box plot. Generally, outliers are considered as the values above q75+(1.5*iqr) or values below q25-(1.5*iqr).

I thought of imputing the outliers instead if removing it. Because, already there are only 3333 observations in my train data. If I remove outliers, then the observations will be reduced. So, I converted the outliers to NA's and imputed using K Nearest Neighbours.

2.3 Feature Selection:

Feature Selection is a process used to select the important features among the predictors for the model building. The general rule is that the target should be dependent on predictors and the predictors (either numeric or categorical) should be independent to each other. If two or more predictors are dependent on each other, then there exists the problem of **multicollinearity**. Then we remove the multicollinear features to get rid of multicollinearity issue. It is selecting relevant features from dataset to use in model. It is also called as Dimensionality reduction. For numerical features we perform correlation and for categorical features we perform Chi-Square test.

In our data, most of the features are numerical and the few categorical variables are converted to numeric by converting levels to numbers. Then Correlation plot is found to know the multicollinearity affected features.

2.4 Feature Scaling:

Features will be in different ranges. Feature Scaling is a technique used to limit the range of features.

First, I'm plotting the data to see the shape of each feature. I'm performing normalization on the skewed features. Then I'm performing standardization on the normally (symmetry) distributed features. Thus, the values in the features are scaled down.

Now the train & test data are cleaned properly and kept ready for modelling.

Chapter 3

Modelling

Since the target variable is binary (categorical), I'm applying the classification algorithms. I planned to use Logistic Regression and Random Forest algorithms to predict the Churn.

3.1 Logistic Regression:

Working of Logistic Regression:

Logistic Regression is a classification algorithm which uses regression techniques and gives results in the form of probabilities. It finds the relationship between predictors and target to predict the target and classifies the predicted target. **Logistic Regression uses logit transformation and maximum likelihood estimation to calculate the likelihood or probability**. The probability values are restricted from 0 to 1. But the regression values will be unbounded (- infinity to + infinity). To solve this, we do logit transformation (I.e. log of odds ratio). This pushes the value to +ve and -ve infinity. Now we project the values to the line and assign the appropriate log odds value. Then transform the log odds to probabilities. The Coefficients of logistic regression are estimated using a technique called Maximum Likelihood Estimation (MLE).

Also, Logistic Regression needs the data to satisfy the following assumptions;

- The target variable should be categorical
- There should not be outliers in the continuous predictor variables.
- There should not be multicollinearity

All these assumptions are satisfied during the data pre-processing. Now applying logistic regression algorithm on train data to build a logistic model. Then the model is used to predict the target on the test data. The predicted values will be in the form of probabilities between 0 and 1. Generally the threshold of 0.5 is set to classify the predicted data. For our analysis, the predicted probabilities less than 0.5 are less likely to churn and greater than 0.5 are highly likely to churn.

3.2 Random Forest:

Working of Random Forest: Random Forest is a tree-based algorithm which build huge number of trees to improve the accuracy of prediction. Therefore, it's called as an ensemble learning technique. The idea of Random Forest is to build n no. of trees to increase accuracy. To reduce misclassification rate and increase accuracy in classification, we combine many decision trees to form a strong classifier. Random Forest works on the basis of bagging, Gini Index (to select the best parent node) and build many trees. Then this model is applied on test data to predict the target.

3.3 Evaluation:

After predicting the target variable in the test data using the model, we find the accuracy of the prediction. Since, this is classification problem we use evaluation metrics such as,

Confusion Matrix, Accuracy, False Negative rate, Recall or Sensitivity or True Positive Rate, Specificity or True Negative Rate are some of the evaluation metrics used for classification algorithms.

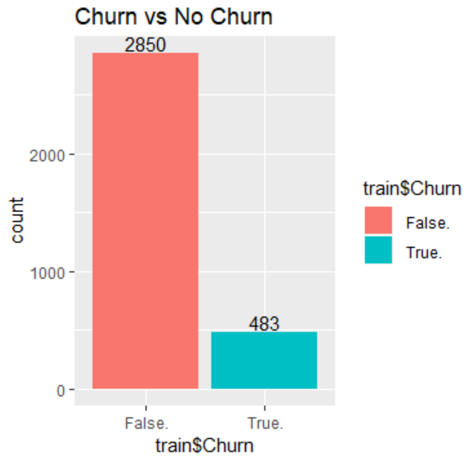
My Analysis, Outputs & Interpretations

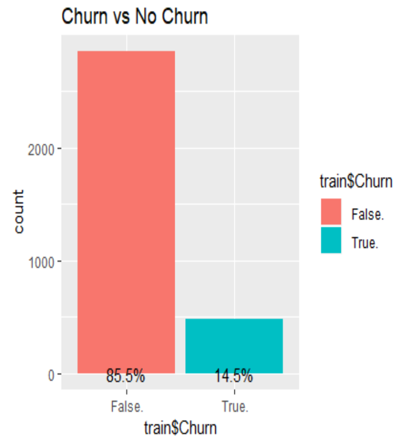
Chapter 4

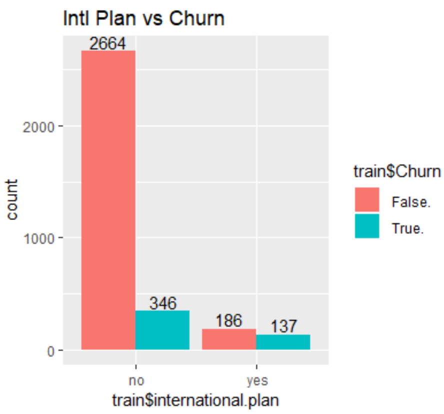
Outputs of my analysis

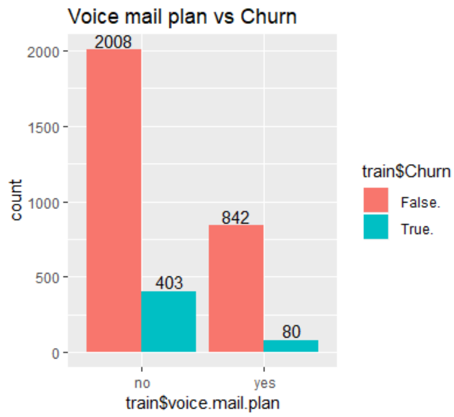
4.1 Churn Prediction in R (The outputs of my analysis are given below) (I didn't give the R codes near to each output. I've submitted the R File for running the code)

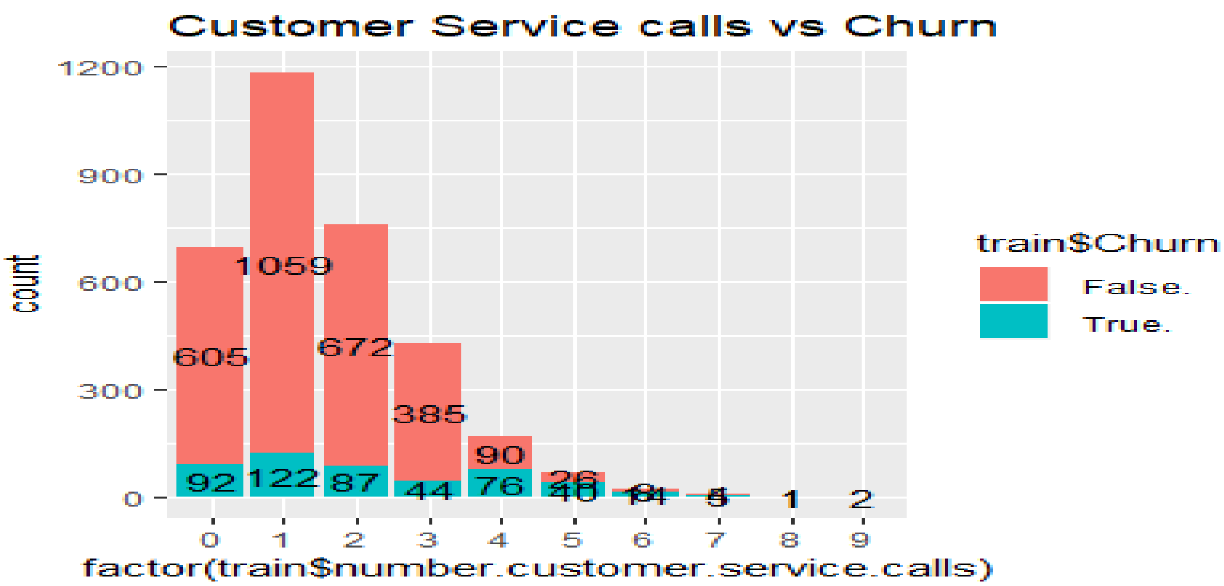
Output of Exploratory Data Analysis in R:

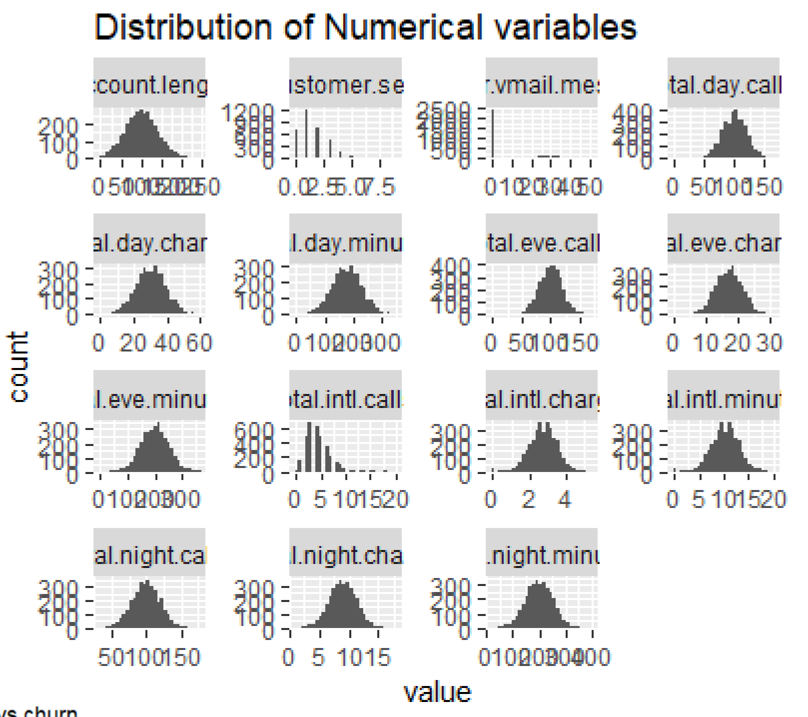


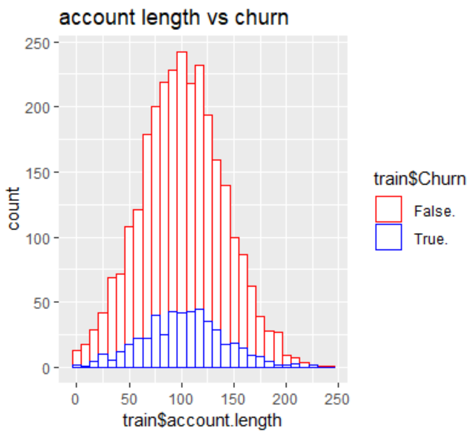


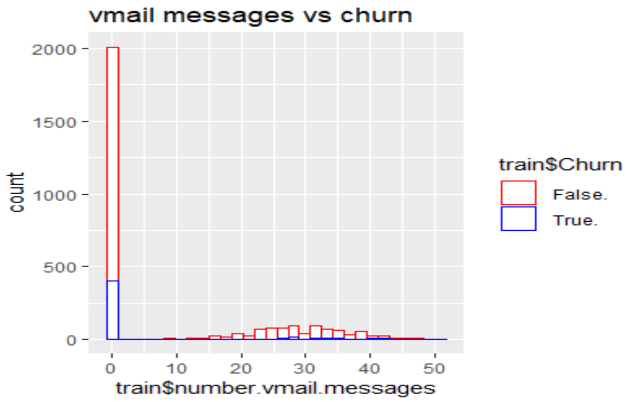


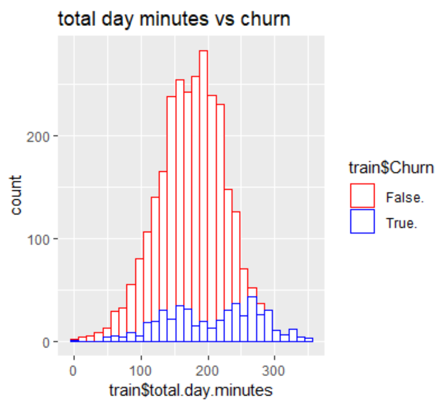


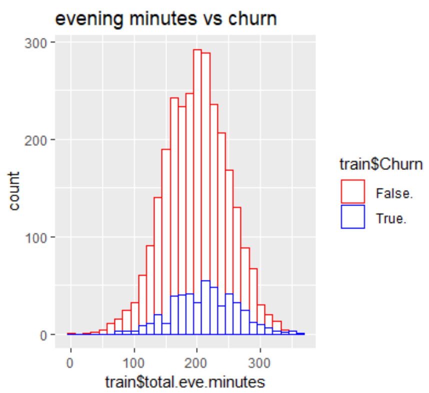


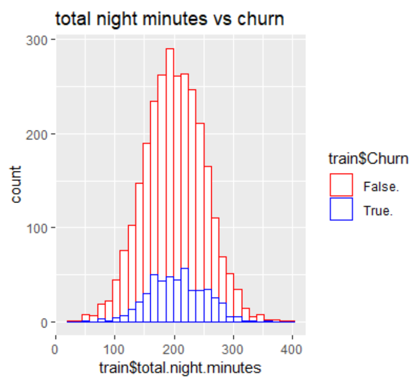


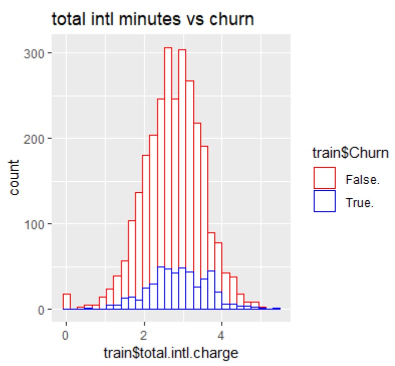












Likewise, many charts can be plotted to understand the data better.

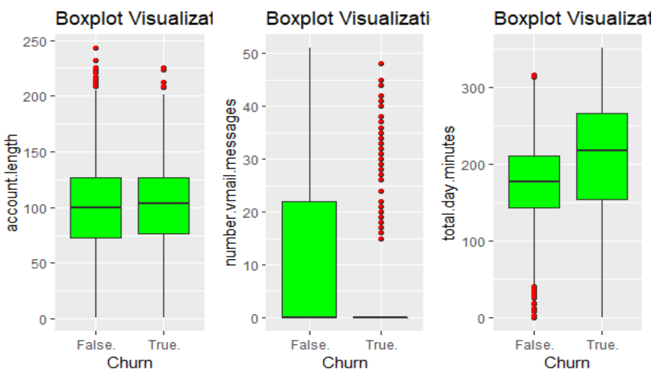
Missing Value Analysis in R:

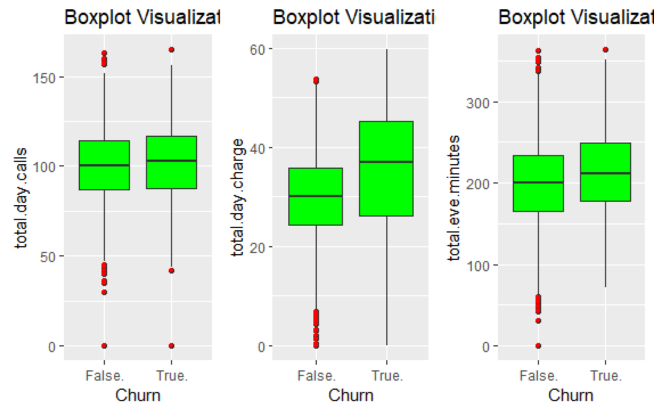
```
                                  apply.train..2..function.x...
account.length                                                0
international.plan                                            0
voice.mail.plan                                               0
number.vmail.messages                                         0
total.day.minutes                                             0
total.day.calls                                               0
total.day.charge                                              0
total.eve.minutes                                             0
total.eve.calls                                               0
total.eve.charge                                              0
total.night.minutes                                           0
total.night.calls                                             0
total.night.charge                                            0
total.intl.minutes                                            0
total.intl.calls                                              0
total.intl.charge                                             0
number.customer.service.calls                                 0
Churn                                                         0
```

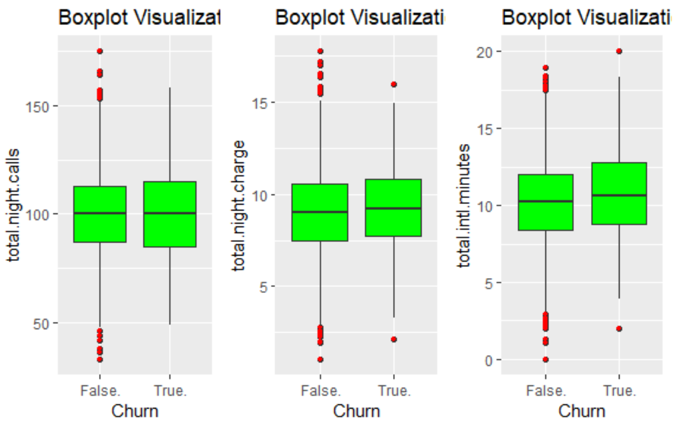
The above image shows that there are no missing values in our dataset.

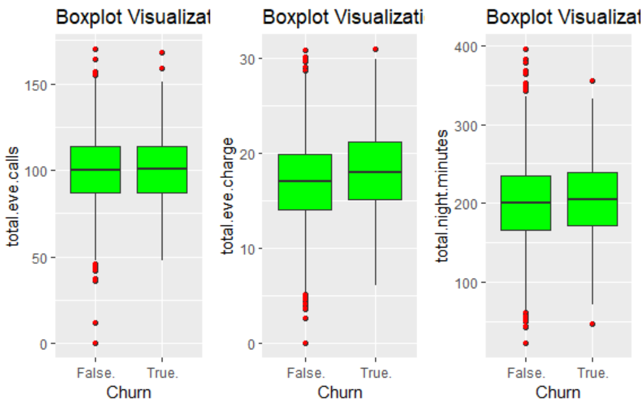
Outlier Detection & Treatment in R:

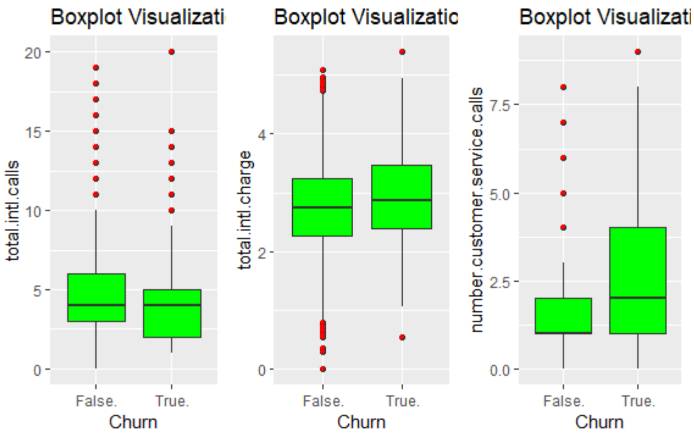
The below images show that each numerical feature has outlier values.



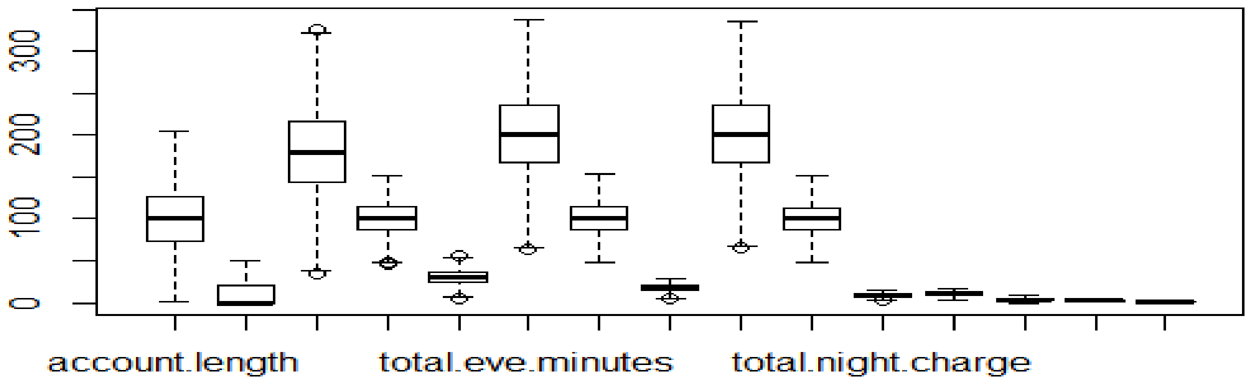






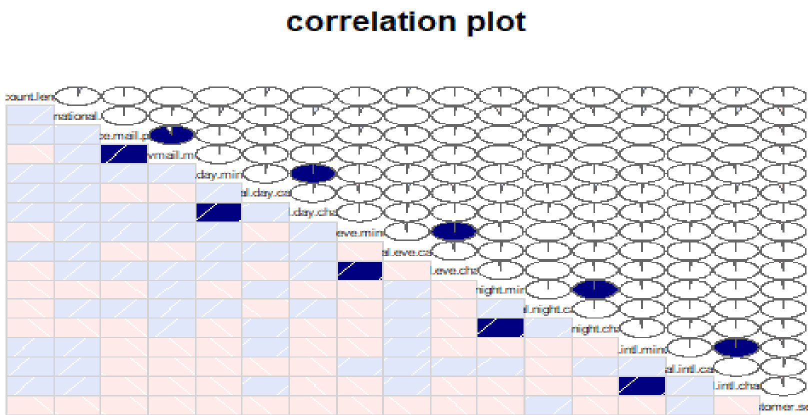


After Imputing Outliers, the outliers have been highly reduced.



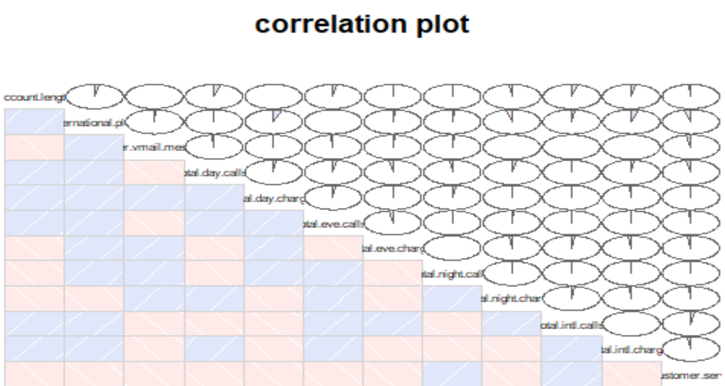
Feature Selection:

Visualizing the predictors in Correlation Plot in order to find the multicollinearity issues. The below chart shows that multicollinearity issue exists in the dataset.



Total day minutes & total day charge, total evening minutes & total evening charge, total night minutes & total night charge, total intl minutes & total intl charge, voice mail plan & number voice mail messages are highly correlated among each other. So, I removed all the minutes variable and voice mail plan variable out of these to get rid of multicollinearity.

After removing multicollinearity affected variables, the below chart looks good.



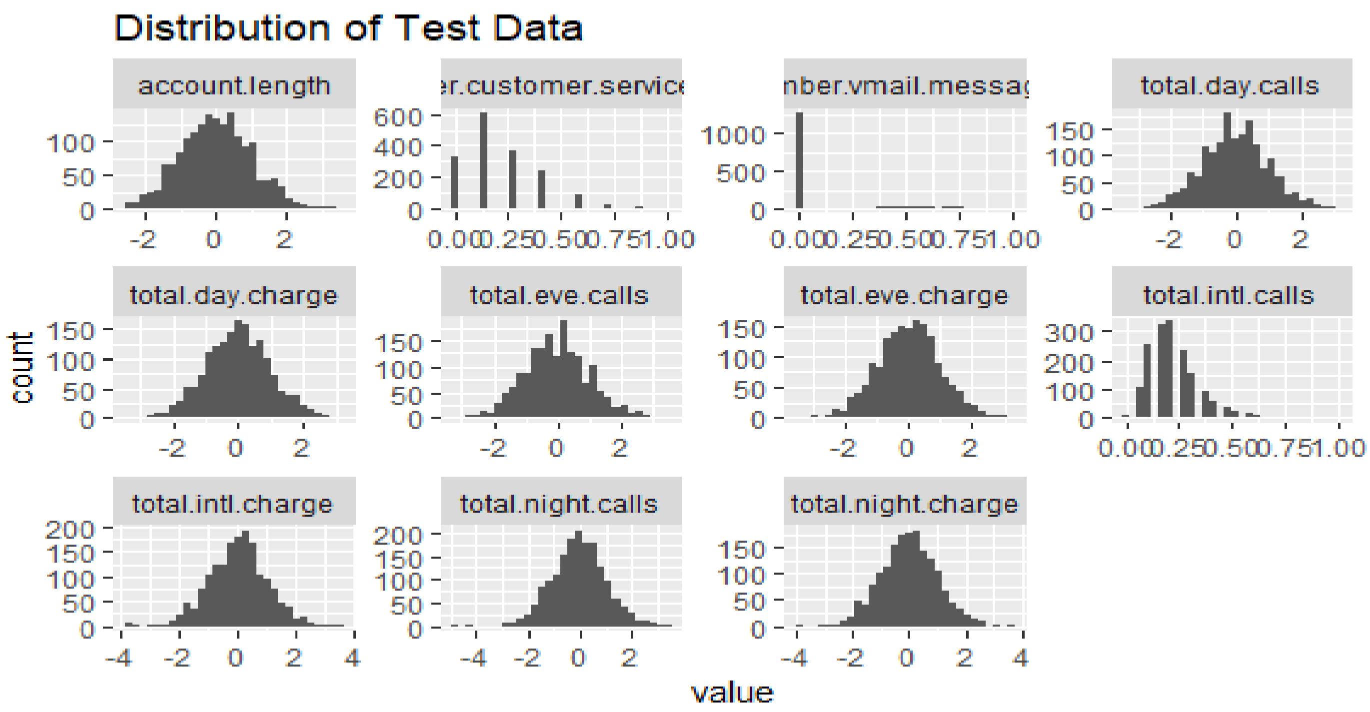
Feature Scaling:

```
> summary(train1)
 account.length      international.plan number.vmail.messages total.day.calls
 Min.   :-2.55534   Min.   :1.000      Min.   :0.0000        Min.   :-2.76460
 1st Qu.:-0.67989   1st Qu.:1.000      1st Qu.:0.0000        1st Qu.:-0.64944
 Median :-0.01192   Median :1.000      Median :0.0000        Median : 0.02122
 Mean   : 0.00000   Mean   :1.097      Mean   :0.1618        Mean   : 0.00000
 3rd Qu.: 0.68175   3rd Qu.:1.000      3rd Qu.:0.4000        3rd Qu.: 0.69188
 Max.   : 2.68566   Max.   :2.000      Max.   :1.0000        Max.   : 2.65228
 total.day.charge    total.eve.calls     total.eve.charge     total.night.calls
 Min.   :-2.747691   Min.   :-2.699842   Min.   :-2.786795   Min.   :-2.7348251
 1st Qu.:-0.679065   1st Qu.:-0.681519   1st Qu.:-0.694696   1st Qu.:-0.6837819
 Median :-0.008491   Median :-0.008744   Median : 0.005067   Median :-0.0001009
 Mean   : 0.000000   Mean   : 0.000000   Mean   : 0.000000   Mean   : 0.0000000
 3rd Qu.: 0.683282   3rd Qu.: 0.664031   3rd Qu.: 0.690451   3rd Qu.: 0.6835802
 Max.   : 2.745213   Max.   : 2.785858   Max.   : 2.768171   Max.   : 2.7346234
 total.night.charge total.intl.calls total.intl.charge   number.customer.service.calls
 Min.   :-2.7836    Min.   :0.0000   Min.   :-2.707505   Min.   :0.0000
 1st Qu.:-0.6896    1st Qu.:0.3000   1st Qu.:-0.657943   1st Qu.:0.3333
 Median : 0.0068    Median :0.4000   Median : 0.001357   Median :0.3333
 Mean   : 0.0000    Mean   :0.4283   Mean   : 0.000000   Mean   :0.4348
 3rd Qu.: 0.6987    3rd Qu.:0.6000   3rd Qu.: 0.660657   3rd Qu.:0.6667
 Max.   : 2.7605    Max.   :1.0000   Max.   : 2.710218   Max.   :1.0000
 Churn   
 0:2850  
 1: 483  
```

Thus, the features are scaled down.

Test Data: (~Same kind of activities are done on test data)*

Distribution of Test Data:



Modelling: Logistic Regression

```
> model1=glm(Churn~.,data=final,family="binomial")
> summary(model1)

Call:
glm(formula = Churn ~ ., family = "binomial", data = final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5626  -0.5489  -0.4206  -0.2833   2.9965

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -3.403673   0.211354 -16.104  < 2e-16 ***
account.length                 0.022709   0.052954   0.429  0.66804
international.plan             1.824291   0.135851  13.429  < 2e-16 ***
number.vmail.messages         -1.304995   0.227340  -5.740 9.45e-09 ***
total.day.calls                0.054678   0.052291   1.046  0.29572
total.day.charge               0.571125   0.054949  10.394  < 2e-16 ***
total.eve.calls                0.002705   0.053330   0.051  0.95955
total.eve.charge               0.271664   0.053905   5.040 4.66e-07 ***
total.night.calls              0.003323   0.052873   0.063  0.94988
total.night.charge             0.166803   0.052999   3.147  0.00165 **
total.intl.calls              -1.094950   0.265439  -4.125 3.71e-05 ***
total.intl.charge              0.170683   0.053722   3.177  0.00149 **
number.customer.service.calls  0.004601   0.166541   0.028  0.97796
---
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: 2378.8  on 3320  degrees of freedom
AIC: 2404.8

Number of Fisher Scoring iterations: 5
```

The above model has some insignificant variables. In order to build a better model, I use Step AIC technique to select the significant variables. **After applying Step AIC technique:** (To remove the insignificant variables from model1)

```
Call:
glm(formula = Churn ~ international.plan + number.vmail.messages +
    total.day.charge + total.eve.charge + total.night.charge +
    total.intl.calls + total.intl.charge, family = "binomial",
    data = final)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5735  -0.5507  -0.4204  -0.2844   3.0196

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -3.40033    0.19649 -17.305  < 2e-16 ***
international.plan     1.82621    0.13562  13.466  < 2e-16 ***
number.vmail.messages -1.30304    0.22703  -5.740 9.50e-09 ***
total.day.charge       0.57247    0.05492  10.423  < 2e-16 ***
total.eve.charge       0.26993    0.05383   5.014 5.32e-07 ***
total.night.charge     0.16793    0.05296   3.171  0.00152 **
total.intl.calls      -1.10084    0.26497  -4.155 3.26e-05 ***
total.intl.charge      0.17177    0.05366   3.201  0.00137 **
---
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: 2380.2  on 3325  degrees of freedom
AIC: 2396.2

Number of Fisher Scoring iterations: 5
```

Prediction on Test Data:

Now the above model is used to predict the target on test data. Then evaluation metrics are used to calculate the accuracy of prediction.

Evaluation Metrics: Confusion Matrix

```
> #Confusion matrix for Logistic Regression:
> confusionMatrix(compare_glm)
Confusion Matrix and Statistics


         0    1
  0 1417   26
  1  190   34

               Accuracy : 0.8704
                 95% CI : (0.8534, 0.8862)
    No Information Rate : 0.964
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1937
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8818
            Specificity : 0.5667
         Pos Pred Value : 0.9820
         Neg Pred Value : 0.1518
             Prevalence : 0.9640
         Detection Rate : 0.8500
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.7242

       'Positive' Class : 0
```

False Negative Rate: 84%

Accuracy: 87%

Important Notes:

- Even though the accuracy of logistic regression model is 87%, but the false negative rate is 84% (FNR is high). This is a problem with this logistic regression model.
- So, we go for other algorithm such as Random Forest.

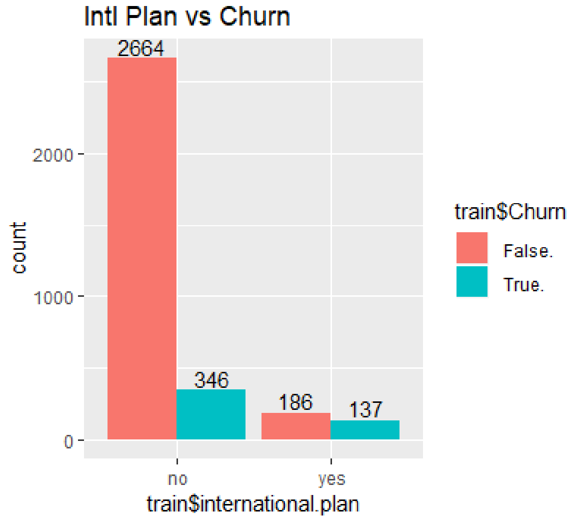
Interpretation of Logistic Regression model:

```
model2$coefficients
        (Intercept)    international.plan number.vmail.messages       total.day.charge
         -3.4003329             1.8262076            -1.3030355              0.5724718
   total.eve.charge    total.night.charge      total.intl.calls      total.intl.charge
          0.2699268             0.1679270            -1.1008420              0.1717700
.
```

Model Interpretation:

By looking at the above beta coefficients value, we can understand that

- **International.plan has high impact on Churn.**



In the above graph, out of 186 International Plan customers, 137 are churning.

It means that those who have chosen International Plan are highly likely to churn. Also, the total. intl.charge is also impacting the churn. So, most of our international plan customers are churning due to high charge for international calls. Also, we might look in to international services to enhance the customer experience & plan to optimise the charges of international services.

- **All the charge (price) variables (total.day.charge, total.eve.charge, total.night.charge, total.intl.charge) are highly impacting churn in our telecom company**.

In this regard, my suggestion for our telecom company is to optimise the price or charges. The customers are churning because the competitor's price might be very less. So, our company needs to look at the charges of competitors and optimise the charges to reduce the churn and retain the customers.

Modelling: Random Forest Model

```
> rf_model1=randomForest(Churn~.,train1,importance=TRUE,ntree=500)
> rf_model1

Call:
 randomForest(formula = Churn ~ ., data = train1, importance = TRUE,      ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 7.98%
Confusion matrix:
     0   1 class.error
0 2838  12 0.004210526
1  254 229 0.525879917
```

Here, the error rate is 7.86%

So, changing the **mtry** value & looking at the error rate. **mtry is no. of predictors available for splitting at each tree node**. Generally mtry value is automatically taken using **m=sqrt(no.of predictors)** and rounded down. Changing the mtry value manually result in changing the error rate in random forest model. A value which gives less error rate can be chosen. Changing the mtry value to 9 gives error rate of 6% approx

```
> # Fine tuning parameters of Random Forest model
> rf_model2<- randomForest(Churn ~ ., train1, ntree = 500, mtry = 9, importance = TRUE)
> rf_model2

Call:
 randomForest(formula = Churn ~ ., data = train1, ntree = 500,      mtry = 9, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 6.21%
Confusion matrix:
     0   1 class.error
0 2827  23 0.008070175
1  184 299 0.380952381
```

Confusion Matrix:

```
> #Confusion matrix for Random Forest:
> confusionMatrix(compare_rf)
Confusion Matrix and Statistics

   prediction_rf
       0    1
  0 1315  128
  1   72  152

               Accuracy : 0.88
                 95% CI : (0.8635, 0.8952)
    No Information Rate : 0.832
    P-Value [Acc > NIR] : 2.85e-08

                  Kappa : 0.5335
 Mcnemar's Test P-Value : 0.0001006

            Sensitivity : 0.9481
            Specificity : 0.5429
         Pos Pred Value : 0.9113
         Neg Pred Value : 0.6786
             Prevalence : 0.8320
         Detection Rate : 0.7888
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.7455

       'Positive' Class : 0
```

False Negative Rate: 33%

Accuracy: 88%

Important Notes:

- The accuracy of Random Forest model is 88% and the false negative rate is 32%. When I built the first random forest model, the FNR was 48%. But after model iterations we got FNR as 32%. (FNR is less and ok)

- Since, the Random Forest in ensemble learning, the FNR is lesser than the logistic regression model.

- We can finalise the Random Forest Model here.

Interpretation of Random Forest model:

```
> rf_model2$importance
                                        0             1 MeanDecreaseAccuracy MeanDecreaseGini
account.length                -9.020477e-05 -0.0020228081        -0.0003728806         32.96224
international.plan             2.476294e-02  0.1588071127         0.0441240988         62.34172
number.vmail.messages          9.447313e-03  0.0563354897         0.0162169258         54.29604
total.day.calls               -4.788035e-04 -0.0001819786        -0.0004416974         36.68242
total.day.charge               3.450896e-02  0.2195585701         0.0611955912        173.03292
total.eve.calls               -5.213629e-04 -0.0012210064        -0.0006254818         32.72156
total.eve.charge               1.008656e-02  0.0786317486         0.0199861669        109.43454
total.night.calls             -3.559545e-04 -0.0013473933        -0.0004935431         35.21415
total.night.charge             2.585189e-03  0.0156756632         0.0044671295         58.27170
total.intl.calls               1.036056e-02  0.0668529256         0.0185335261         69.27825
total.intl.charge              6.825632e-03  0.0613647573         0.0147133718         80.65443
number.customer.service.calls  1.139750e-02  0.0777368669         0.0209534192         80.82222
```

With the above image, we can find the top most impacting features of churn. Generally, a higher Mean Decrease in Gini indicates higher variable importance.

Model Interpretation:

From the model, the top features impacting Churn are,

- Total day charge
- Total evening charge
- Total intl charge
- Number.customer.service.calls
- International plan
- Total night charge

It’s clear that charges of each service lead to churn in our telecom company. So, we need to optimise the charges.

International call using customers are churning more in our company due to high charges.

Also, if customers are calling to customer care more times, then there is high likelihood of churn.

4.2 Churn Prediction in Python:

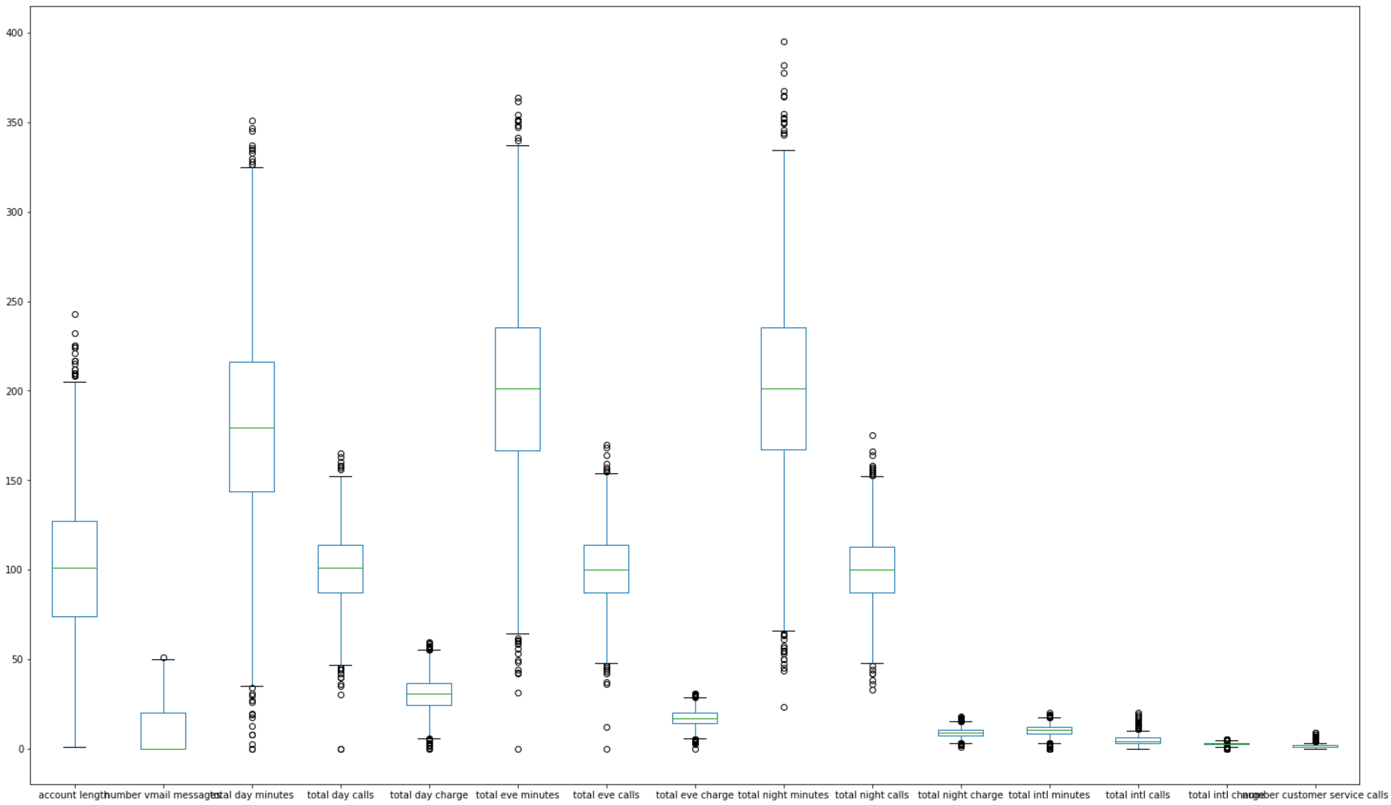
Missing Value Analysis in Python:

```
#Checking Missing value:

mv=pd.DataFrame(train.isnull().sum())

mv
```

	0
account length	0
international plan	0
voice mail plan	0
number vmail messages	0
total day minutes	0
total day calls	0
total day charge	0
total eve minutes	0
total eve calls	0
total eve charge	0
total night minutes	0
total night calls	0
total night charge	0
total intl minutes	0
total intl calls	0
total intl charge	0
number customer service calls	0
Churn	0

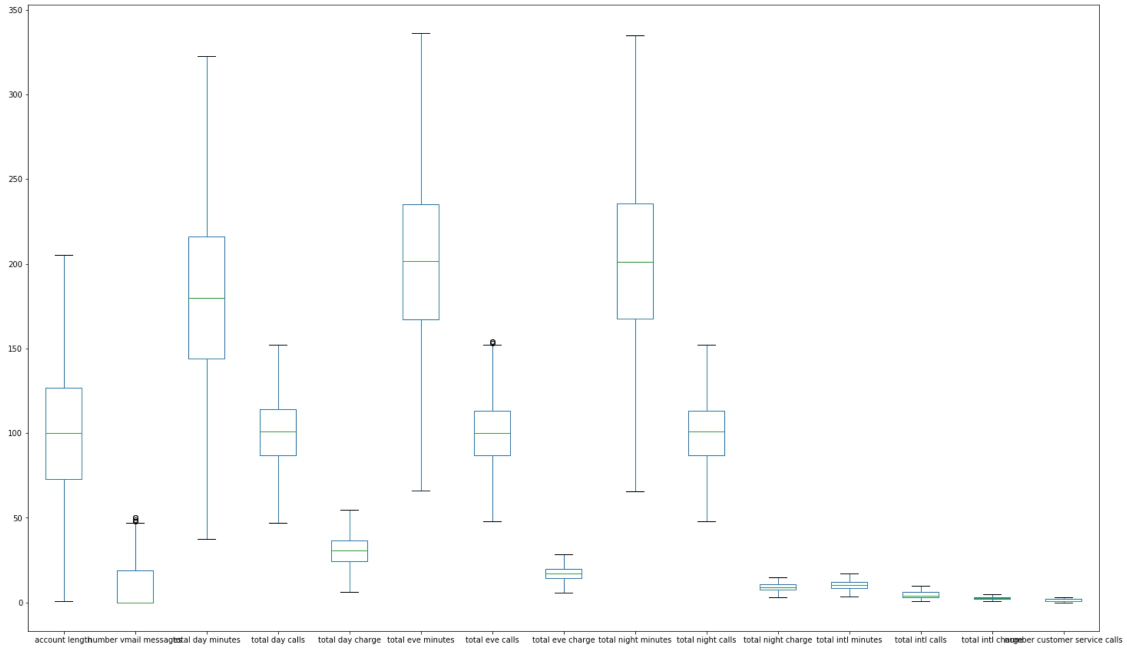
The above image shows that there are no missing values in our dataset.

Outlier Detection & Treatment in Python:



In Python, I tried removing the outliers instead of imputing through KNN. (In R, I used KNN imputation).

After removing the outliers,



Feature Selection:

Visualising the predictors in Correlation Plot in order to find the multicollinearity issues. The below chart shows that multicollinearity issue exists in the dataset.

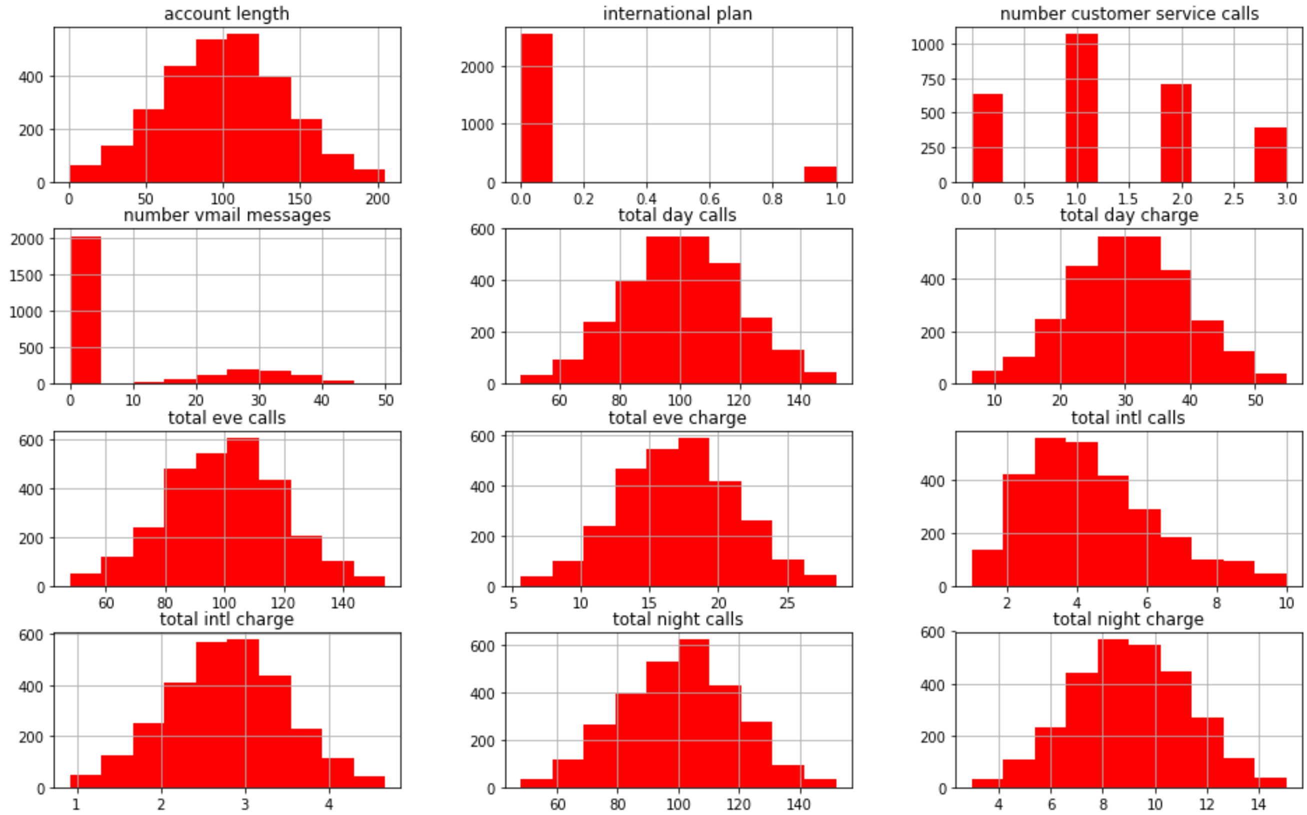
```
correlation_train.style.background_gradient(cmap='summer')
```

	account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	tota
account length	1	0.0208327	0.0101142	0.00308635	0.00605682	0.042087	0.00605497	-0.012879	0.0224776	-0.0128613	-0.00376814	-0.0
iternational plan	0.0208327	1	0.00496562	0.00878618	0.0555571	0.0249213	0.0555589	0.00324282	-0.00300287	0.00324799	-0.026875	0.00
voice mail plan	0.0101142	0.00496562	1	0.95664	0.000677823	-0.0141319	0.000676292	0.0105075	-0.0109142	0.01052	0.00616803	0.
number vmail messages	0.00308635	0.00878618	0.95664	1	0.00579417	-0.011522	0.00579171	0.0051532	-0.0108512	0.0051688	0.0119289	0.00
total day minutes	0.00605682	0.0555571	0.000677823	0.00579417	1	0.0131918	1	0.00397473	0.00640318	0.00396169	0.00388349	0.000
total day calls	0.042087	0.0249213	-0.0141319	-0.011522	0.0131918	1	0.0131952	-0.0149999	0.0200736	-0.0149899	0.0244436	-0.00
total day charge	0.00605497	0.0555589	0.000676292	0.00579171	1	0.0131952	1	0.00398008	0.00640267	0.00396705	0.00387911	0.000
total eve minutes	-0.012879	0.00324282	0.0105075	0.0051532	0.00397473	-0.0149999	0.00398008	1	-0.0222539	1	-0.0135516	-0.0
total eve calls	0.0224776	-0.00300287	-0.0109142	-0.0108512	0.00640318	0.0200736	0.00640267	-0.0222539	1	-0.0222544	0.0153648	0.00
total eve charge	-0.0128613	0.00324799	0.01052	0.0051688	0.00396169	-0.0149899	0.00396705	1	-0.0222544	1	-0.0135634	-0.0
total night minutes	-0.00376814	-0.026875	0.00616803	0.0119289	0.00388349	0.0244436	0.00387911	-0.0135516	0.0153648	-0.0135634	1	0.0
total night calls	-0.0044084	0.00737094	0.010364	0.00186075	0.000443686	-0.00592843	0.000447681	-0.0100278	0.00554836	-0.0100155	0.0005846	

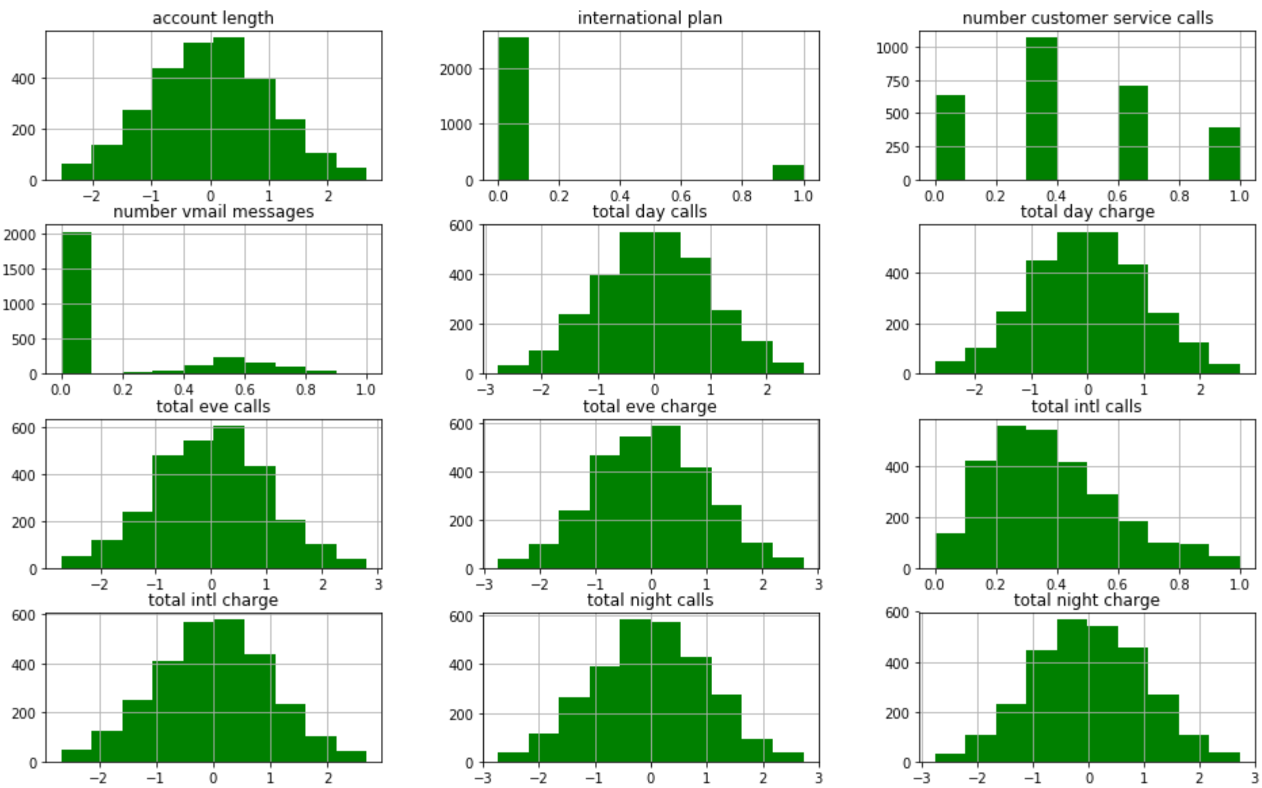
Total day minutes & total day charge, total evening minutes & total evening charge, total night minutes & total night charge, total intl minutes & total intl charge, voice mail plan & number voice mail messages are highly correlated among each other. So, I removed all the minutes variable and voice mail plan variable out of these to get rid of multicollinearity.

Feature Scaling:

Distribution of Variables:



After Scaling down the numerical features,



In the above chart all the features except international plan have been scaled down. The x axis units are scaled down in the chart.

Modelling: Logistic Regression

(Logistic Regression output)

Logit Regression Results

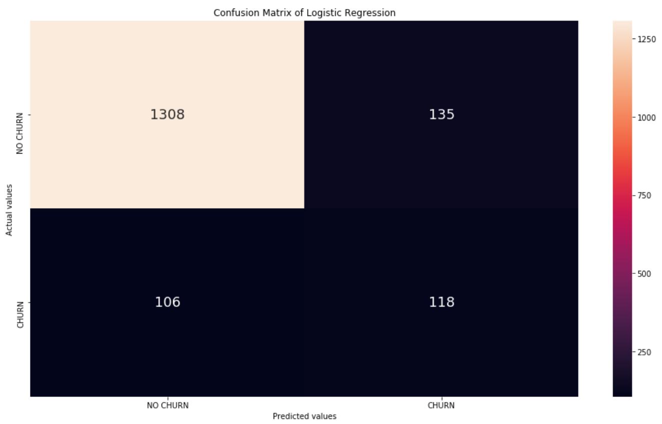
Dep. Variable:	Churn	No. Observations:	2797
Model:	Logit	Df Residuals:	2785
Method:	MLE	Df Model:	11
Date:	Sat, 16 Mar 2019	Pseudo R-squ.:	0.1629
Time:	11:21:45	Log-Likelihood:	-804.87
converged:	True	LL-Null:	-961.51
		LLR p-value:	1.402e-60

	coef	std err	z	P>|z|	[0.025	0.975]
account length	0.0804	0.064	1.253	0.210	-0.045	0.206
international plan	1.9480	0.180	10.839	0.000	1.596	2.300
number vmail messages	-2.8281	0.308	-9.170	0.000	-3.433	-2.224
total day calls	0.0125	0.063	0.201	0.841	-0.110	0.135
total day charge	0.9735	0.072	13.444	0.000	0.832	1.115
total eve calls	-0.0347	0.065	-0.536	0.592	-0.162	0.092
total eve charge	0.4898	0.066	7.405	0.000	0.360	0.619
total night calls	0.0607	0.064	0.947	0.343	-0.065	0.186
total night charge	0.2548	0.065	3.928	0.000	0.128	0.382
total intl calls	-4.5128	0.273	-16.527	0.000	-5.048	-3.978
total intl charge	0.2444	0.065	3.757	0.000	0.117	0.372
number customer service calls	-1.7870	0.177	-10.118	0.000	-2.133	-1.441

Prediction on Test Data:

Now the above model is used to predict the target on test data. Then evaluation metrics are used to calculate the accuracy of prediction.

Evaluation Metrics: Confusion Matrix



False Negative Rate: 47%

Accuracy: 85.5%

	precision	recall	f1-score	support
0	0.93	0.91	0.92	1443
1	0.47	0.53	0.49	224
micro avg	0.86	0.86	0.86	1667
macro avg	0.70	0.72	0.71	1667
weighted avg	0.86	0.86	0.86	1667

Interpretation of Logistic Regression model:

	coef
account length	0.0804
international plan	1.9480
number vmail messages	-2.8281
total day calls	0.0125
total day charge	0.9735
total eve calls	-0.0347
total eve charge	0.4898
total night calls	0.0607
total night charge	0.2548
total intl calls	-4.5128
total intl charge	0.2444
number customer service calls	-1.7870

By looking at the coefficients of each feature from the logit model, we can find the top most features impacting the churn in our telecom company.

Model Interpretation:

- **The International plan variable has high impact towards churning.**

  It means that the customers who have taken International Plan are likely to churn. Since the total intl charge is also impacting the Churn. It's evident that International Plan customers are churning due to high charges levied on international calls.

  So, we need to optimise the international call charges and enhance the service to our international plan opted customers.

- **All the charge variables are impacting towards churn**. It's evident that the customers are churning due to huge charges. So, we need to optimise the charges in order to retain the customers.

Modelling: Random Forest Model

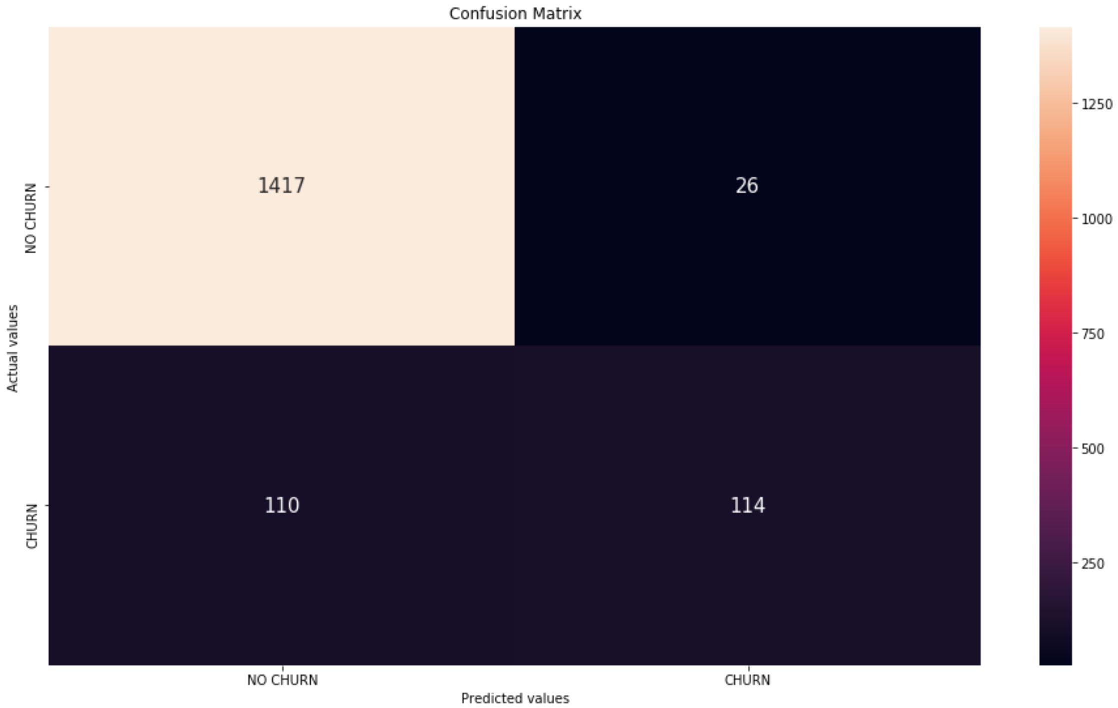
```
model_rf=RandomForestClassifier(n_estimators=500).fit(train[train_x],train_y)
model_rf
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Prediction on Test Data:

Now the above model is used to predict the target on test data. Then evaluation metrics are used to calculate the accuracy of prediction.

Evaluation Metrics: Confusion Matrix



False Negative Rate: 49%

Accuracy: 91%

	precision	recall	f1-score	support
0	0.93	0.98	0.95	1443
1	0.81	0.51	0.63	224
micro avg	0.92	0.92	0.92	1667
macro avg	0.87	0.75	0.79	1667
weighted avg	0.91	0.92	0.91	1667

Interpretation of Random Forest model:

	Variable	Importance
4	total day charge	0.275943
6	total eve charge	0.109508
1	international plan	0.108095
10	total intl charge	0.089898
9	total intl calls	0.073903
8	total night charge	0.067766
2	number vmail messages	0.052601
0	account length	0.052084
7	total night calls	0.051781
3	total day calls	0.051410
5	total eve calls	0.049097
11	number customer service calls	0.017913

This table talks about the importance of each predictor in impacting the churn in our telecom company.

Model Interpretation:

Top features impacting Churn are,

- Total day charge
- Total evening charge
- International plan
- Total intl charge
- Total intl calls
- Total night charge

It shows that charges (price) in our telecom company leads to churn. So, we need to optimise the charges for all ore telecom services. Most importantly we need to reduce the charges levied on international calls and enhance the service.

Chapter 5

Story Telling

Story Telling is a major step in analytics project path. It's interpreting the analysis results to the business people in business terms. It will be like answering the business queries and giving suggestions on what business needs to do proactively, in order to develop the business.

Here are my collective suggestions to the Telecom company,

Suggestion 1:

Insight: **The major reason for Churn in this telecom company is the huge charges (price) on each service.**

Action: In this regard, my suggestion for this telecom company is to optimise (reduce) the charges.

The customers are churning because the competitor's price might be very less. So, our company need to look at the charges of competitors and optimise the charges to reduce the churn and retain the customers. In order to retain the customers in the future, the charges have to be reduced.

The company can introduce price optimised plans like bundle plans which benefit the customers.

Benefits: If the Charges are optimised then the customers will not churn. Low Prices may attract the new prospects and there will be increase in customer rate to our telecom company. Also, the usage of our service will increase.

Suggestion 2:

Insight: **Most of the International Plan chosen customers are churning due to huge charges levied on International calls. Also, there might be any service issue in international service.**

Action: The company must look in to it to enhance the quality of International service with low price.

Benefits: Automatically this will reduce the churn rate. Also, will attract the new prospects who need to do international calls often.

Suggestion 3:

Insight: If customers call the customer care many times, then it's a sign for churn.

Action: So, if any customer is calling the customer care many times, their query needs to be addressed properly. For those customers we can give any special offers to retain them.

Benefits: Churn rate will reduce.

Chapter 6

6.1 R Codes:

```
#CHURN PREDICTION (TELCOM SECTOR)


#Setting working directory
setwd("C:/Users/Ranjith P/Desktop/EDWISOR")
getwd()


#Loading the train and test dataset:
train_data=read.csv("C:/Users/Ranjith P/Desktop/EDWISOR/Train_data.csv")
test_data=read.csv("C:/Users/Ranjith P/Desktop/EDWISOR/Test_data.csv")


#Removing 3 variables named "state","area code","phone number";
#which are not given as predictors in data dictionary
train=train_data[,-c(1,3,4)]
dim(train)


test=test_data[,-c(1,3,4)]
dim(test)


#Installing Packages:
#install.packages("MASS","DMwR","ggplot2","purrr","tidyr","corrgram","caret","randomForest","RRF
")


#Loading the libraries:
library(MASS)
library(DMwR)
library(plyr)
library(ggplot2)
library(purrr)
library(tidyr)
library(corrgram)
library(caret)
library(randomForest)
library(RRF)


#Exploratory  Data Analysis of Train Data:
summary(train)
dim(train)
str(train)


#Data Cleaning & Data Preparation:


#Missing value Analysis and Treatment:
mv=data.frame(apply(train, 2, function(x){sum(is.na(x))}))
mv
#It's found that there is no missing value in any variable in our dataset.


#Data Visualization (EDA):

#Here Target variable is Churn
#Target class proportion:
```

```
ggplot(train,aes(train$Churn))+geom_bar(aes(fill = train$Churn),position = "dodge")+
  labs(title = "Churn vs No Churn")+
  geom_text(aes(label=scales::percent(..count../sum(..count..))),
            stat='count',position=position_fill(vjust=0.5))


#Plotting categorical variable"international.plan" and Target:
ggplot(train,aes(x=train$international.plan,fill=train$Churn))+
  geom_bar(position="dodge")+labs(title = "Intl Plan vs Churn")+
  geom_text(aes(label=..count..),stat='count',position=position_dodge(0.9),vjust=-0.2)


#Plotting categorical variable"train$voice.mail.plan" and Target:
ggplot(train,aes(x=train$voice.mail.plan,fill=train$Churn))+
  geom_bar(position="dodge")+labs(title = "Voice mail plan vs Churn")+
  geom_text(aes(label=..count..),stat='count',position=position_dodge(0.9),vjust=-0.2)


#Plotting number.customer.service.calls and Target:
ggplot(train,aes(x=factor(train$number.customer.service.calls),fill=train$Churn))+
  geom_bar()+labs(title = "Customer Service calls vs Churn")+
  geom_text(aes(label=..count..),stat="count",position=position_stack(0.5))


#Plotting all numerical variables:
train %>%keep(is.numeric) %>% gather() %>%ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +geom_histogram()+
  labs(title = "Distribution of Numerical variables")


#Plotting some numerical variables with Target variable in order to understand the nature of dat
a:

ggplot(train, aes(x = train$account.length))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "account length vs churn")


ggplot(train, aes(x = train$number.vmail.messages))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "vmail messages vs churn")


ggplot(train, aes(x = train$total.day.minutes))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "total day minutes vs churn")


ggplot(train, aes(x = train$total.eve.minutes))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "evening minutes vs churn")


ggplot(train, aes(x = train$total.night.minutes))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "total night minutes vs churn")


ggplot(train, aes(x = train$total.intl.charge))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
  scale_color_manual(values = c("red", "blue"))+labs(title = "total intl minutes vs churn")


ggplot(train, aes(x = train$number.customer.service.calls))+
  geom_histogram(aes(color = train$Churn), fill = "white", position = "identity") +
```

```
  scale_color_manual(values = c("red", "blue"))+labs(title = "No. of Customer Service calls vs C
hurn")

#We can change the x variables here to view the plots of other independent numeric variables.#Al
so, it's evident that most of the variables are normally distributed.


#Outlier Analysis and Treatment:
#Taking only the continuous variables to deal with outliers.
numerics=sapply(train, is.numeric)
numeric_train=train[,numerics]
train_numeric_names=colnames(numeric_train)
train_numeric_names


#Visulaizing the outliers using boxplot:
boxplot(numeric_train)


#Plotting the variables seperately based on Target:
for(i in 1:length(train_numeric_names))
{
  assign(paste0("Train",i),ggplot(aes_string(y = (train_numeric_names[i]),x="Churn"),data = subs
et(train))+
           stat_boxplot(geom = "Boxplot",width = 0.5)+
           geom_boxplot(outlier.colour = "red",fill = "green",outlier.shape = 20,
                        outlier.size = 2,notch = FALSE)+
           theme(legend.position = "Top")+
           labs(y=train_numeric_names[i],x="Churn")+
           ggtitle(paste("Boxplot Visualization",train_numeric_names[i])))
}
gridExtra::grid.arrange(Train1,Train2,Train3,ncol = 3)
gridExtra::grid.arrange(Train4,Train5,Train6,ncol = 3)
gridExtra::grid.arrange(Train7,Train8,Train9,ncol =3)
gridExtra::grid.arrange(Train10,Train11,Train12,ncol = 3)
gridExtra::grid.arrange(Train13,Train14,Train15,ncol = 3)



#Since the no.of observations in train dataset are just 3333, I'm not removing the outliers.
#Instead I'm imputing the outliers using KNN.


#Changing the outlier values to NA's
for(i in train_numeric_names){
  outlier_value=train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
  train[,i][train[,i] %in% outlier_value]=NA
}


sum(is.na(train))
mv1=data.frame(apply(train, 2, function(x){sum(is.na(x))}))
mv1
#Thus the outlier values have been changed as NA's


#Imputing NA's using KNN Algorithm
train1=knnImputation(train,k=3)
sum(is.na(train1))
#Thus treated the NA's using KNN


#Boxplot visualization after imputing outliers:
boxplot(train[,-c(2,3,18)])
```

```
#Transforming the categorical variables:
#Changing the target to 0=False. 1=True.
summary(train1$Churn)
unique(train1$Churn)# Identifying the no. of levels
train1$Churn=factor(train1$Churn,levels = c(" False.", " True."),labels = c(0,1))


#Changing the categorical independent variables in to factor. Normally we will create dummy vari
ables.
#Since there are only two levels in both categorical variables, I'm performing the following.
unique(train1$international.plan) # Identifying the no. of levels
train1$international.plan=as.integer(train1$international.plan)


unique(train1$voice.mail.plan) # Identifying the no. of unique levels
train1$voice.mail.plan=as.integer(train1$voice.mail.plan)


#Feature Selection:
str(train1)
summary(train1)


#Correlation to check for multicollinearity among independent variables:


#Correlation Plot:
corrgram(train1[,],order=FALSE,upper.panel=panel.pie,text.panel=panel.txt,main="correlation  plot
")
#The chart shows that some variables are highly correlated. So, we drop variables to get rid of m
ulticollinearity.


colnames(train1)
train1=subset(train1,select=-c(voice.mail.plan,total.day.minutes,total.eve.minutes,total.night.m
inutes,total.intl.minutes))
#Thus removed the multicollineared varibles.


corrgram(train1[,],order=FALSE,upper.panel=panel.pie,text.panel=panel.txt,main="correlation  plot
")
#Thus there is no multicollinearity problem in the dataset.


#Feature Scaling:
range(train1$account.length)


#Ploting all predictors together in histogram:
train1 %>%keep(is.numeric) %>% gather() %>%ggplot(aes(value)) +facet_wrap(~ key, scales = "free"
) +geom_histogram()


#The above graph shows that some variables such as "account.length","total.day.calls","total.day
.charge",
#"total.eve.calls","total.eve.charge","total.intl.charge","total.night.calls","total.night.charg
e" are mostly normally distributed. So, we perform standardisation on it.


#Other variables such as "number.customer.service.calls","number.vmail.messages","total.intl.cal
ls" are skewed.
#So, I perform normalization on these variables.


#Standardisation:
```

```
train_normal_names=c("account.length","total.day.calls","total.day.charge","total.eve.calls","to
tal.eve.charge","total.intl.charge","total.night.calls","total.night.charge")


for(i in train_normal_names){
  print(i)
  train1[,i]=(train1[,i]-mean(train1[,i]))/sd(train1[,i])
}


range(train1$account.length)
sum(is.na(train1))


#Normalisation:
train_skew_names=c("number.customer.service.calls","number.vmail.messages","total.intl.calls" )


for (i  in train_skew_names){
  print(i)
  train1[,i]=(train1[,i]-min(train1[,i]))/(max(train1[,i]-min(train1[,i])))
}


summary(train1)



#Saving final training dataframe as final
final=train1


################################################################################################
#################3
#Test data:


#Exploratory Data Analysis of Test Data:
summary(test)
dim(test)
str(test)


#Missing value Analysis and Treatment on Test Data:
mvt=data.frame(apply(test, 2, function(x){sum(is.na(x))}))
mvt


#Visualising outliers:
#Taking only the continuous variables to deal with outliers.
test_numerics_index=sapply(test, is.numeric)
test_numerics=test[,test_numerics_index]
#test_numeric_names=colnames(test_numerics)
#test_numeric_names
boxplot(test_numerics)


#Transforming the categorical variables:
summary(test$Churn)
test$Churn<- factor(test$Churn,levels = c(" False.", " True."),labels = c(0,1))


#test$international.plan <- factor(test$international.plan,levels = c(" no", " yes"),labels = c(
0,1))
test$international.plan=as.integer(test$international.plan)
summary(test$international.plan)
```

```
test$voice.mail.plan=as.integer(test$voice.mail.plan)
summary(test$voice.mail.plan)


str(test)


#Feature Selection:
corrgram(test[,],order=FALSE,upper.panel=panel.pie,text.panel=panel.txt,main="correlation plot")


#Removing multicollineared variables:(These were removed in train data. So, removing in teat dat
a also.)
test1=subset(test,select=-c(voice.mail.plan,total.day.minutes,total.eve.minutes,total.night.minu
tes,total.intl.minutes))


corrgram(test1[,],order=FALSE,upper.panel=panel.pie,text.panel=panel.txt,main="correlation  plot"
)


#Feature Scaling:
test1 %>%keep(is.numeric) %>% gather() %>%ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +geom_histogram()+
  labs(title = "Distribution of Test Data")


#Standardisation:
test_normal_names=c("account.length","total.day.calls","total.day.charge","total.eve.calls","tot
al.eve.charge","total.intl.charge","total.night.calls","total.night.charge")
for(i in test_normal_names){
  print(i)
  test1[,i]=(test1[,i]-mean(test1[,i]))/sd(test1[,i])
}


#Normalisation
test_skew_names=c("number.customer.service.calls","number.vmail.messages","total.intl.calls" )
for (i  in test_skew_names){
  print(i)
  test1[,i]=(test1[,i]-min(test1[,i]))/(max(test1[,i]-min(test1[,i])))
}


summary(test1)


#Removing target variable form test data seperately.
test2=test1[,-13]

summary(test2)


################################################################################################
####################
#Model Building and Prediction:
#Logistic Regression:

model1=glm(Churn~.,data=final,family="binomial")
summary(model1)
```

```
#Step AIC to remove insignificant variables:
stepAIC(model1)
model2=glm(Churn ~ international.plan + number.vmail.messages +
             total.day.charge + total.eve.charge + total.night.charge +
             total.intl.calls + total.intl.charge, family = "binomial",
             data = final)
summary(model2)


model2$coefficients


prediction_glm=round(predict(model2,newdata =test2,type="response"),2)
prediction_glm


#Setting threshold value:
#prediction_glm=ifelse(prediction_glm>0.5,1,0)


compare_glm=table(test1$Churn,round(prediction_glm))
compare_glm


#Confusion matrix for Logistic Regression:
confusionMatrix(compare_glm)


#FNR:(~)
190/(190+34)
#FNR: 84% which is very high. So, need to try with other model.
#Accuracy: 87%


#Creating Data Frame of actual and predicted values:
df_glm=data.frame(test_data$phone.number,test1$Churn,round(prediction_glm))
df_glm$round.prediction_glm.=as.factor(df_glm$round.prediction_glm.)
View(df_glm)


#Logistic Regression:   Accuracy=87%     FNR= 84%  (Need to reduce the FNR)



#Random Forest:
rf_model1=randomForest(Churn~.,train1,importance=TRUE,ntree=500)
rf_model1
summary(rf_model1)


# Fine tuning parameters of Random Forest model (ie., changing mtry value to 9)
rf_model2<- randomForest(Churn ~ ., data=train1, ntree = 500, mtry = 9, keep.forest=TRUE,importa
nce = TRUE)
rf_model2
#Thus the error rate is reduced.


prediction_rf=predict(rf_model2,test2)
prediction_rf


compare_rf=table(test1$Churn,prediction_rf)
compare_rf


#Confusion matrix for Random Forest:
```

```
confusionMatrix(compare_rf)


#FNR: (~may change slightly)
75/(75+149)
#FNR ~:33% (ok)
#Accuracy:88%


#Creating dataframe of actual and predicted values:
df_rf=data.frame(test_data$phone.number,test1$Churn,prediction_rf)
df_rf$prediction_rf=as.factor(df_rf$prediction_rf)
View(df_rf)



#Variable Importances from Random Forest:
varImp(rf_model2)
rf_model2$importance

#Random Forest:       Accuracy=88%     FNR=33%   (Model is optimal)



#Data Frame of prediction using both algorithms and actual values:
df=data.frame(test_data$phone.number,test1$Churn,round(prediction_glm),prediction_rf)
summary(df)


###############################################################################################
```

6.2 Python Codes:

```
# # CUSTOMER CHURN PREDICTION
# In[1]:
pwd
# # Importing Libraries
# In[2]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# # Importing Train & Test datasets
# In[3]:
#Importing train & test datasets:
train_data=pd.read_csv("Train_data.csv")
test_data=pd.read_csv("Test_data.csv")
# # Data Understanding
# In[4]:
train_data.head(3)
# In[5]:
test_data.head(3)
# In[6]:
#Dropping 3 variables named 'state', 'area code', 'phone number' ; which are not
mentioned as predictors in data dictionary.
train=train_data.drop(['state', 'area code', 'phone number'], axis=1)
test=test_data.drop(['state', 'area code', 'phone number'], axis=1)
# In[7]:
print(train.columns)
print(test.columns)
# In[8]:
#Summary
train.describe(include="all")
# In[9]:
train.shape
# In[10]:
train.dtypes
# In[11]:
train.columns
# In[12]:
#Data Visualizations:
plt.rcParams["figure.figsize"] = [16,9]
train.hist(figsize = (16,10),color="pink")
# # Preparation of Train Data
# # Detecting Missing Values on Train Data
# In[13]:
#Checking Missing value:
mv=pd.DataFrame(train.isnull().sum())
mv
# In[14]:
#The above dataframe and th edescribe function also shows that there is no missing
value in train data.
# # Outlier Detection and Treatment on Train Data
# In[15]:
#Outlier Analysis and Treatment:
#Box-plot visualisation:
train.plot(kind='box', figsize=[25,15])
# In[16]:
train_numeric_names=['account length','number vmail messages', 'total day minutes',
'total day calls',
                    'total day charge', 'total eve minutes', 'total eve calls',
                    'total eve charge', 'total night minutes', 'total night calls',
                    'total night charge', 'total intl minutes', 'total intl calls',
```

```
                    'total intl charge', 'number customer service calls']

train_numeric_names

# In[17]:
for i in train_numeric_names:
    print(i)
    q75,q25=np.percentile(train.loc[:,i],[75,25])
    iqr=q75-q25

    min=q25-(1.5*iqr)
    max=q75+(1.5*iqr)
    print(min)
    print(max)
    train=train.drop(train[train.loc[:,i]<min].index)
    train=train.drop(train[train.loc[:,i]>max].index)

# In[18]:
#Box-plot visualisation after removing outliers:
train.plot(kind='box', figsize=[25,15])
# # Transforming Categorical variables
# In[19]:
train['voice mail plan'] = train['voice mail plan'].map(lambda x: x.strip())
train['international plan'] =train['international plan'].map(lambda x: x.strip())
train['Churn'] = train['Churn'].map(lambda x: x.strip())
# In[20]:
# Convert the categorical variable into numeric variable:
train['voice mail plan'] = train['voice mail plan'].map({'no':0, 'yes':1})
train['international plan'] = train['international plan'].map({'no':0, 'yes':1})
train['Churn'] = train['Churn'].map({'False.':0, 'True.':1})
train.head()
# In[21]:
train["Churn"]=train["Churn"].astype(object)
# In[22]:
train.dtypes
# # Feature Selection
# In[23]:
#Feature Selection:
#Using correlation:
train_num_names=train.select_dtypes(['int64','float64']).columns
train_corr=train.loc[:,train_num_names]
train_corr.shape
correlation_train=train_corr.corr()
correlation_train
# In[24]:
correlation_train.style.background_gradient(cmap='summer')
# In[25]:
#Dropping some multicollineared variables to get rid of multicollinearity:
train=train.drop(['voice mail plan', 'total day minutes', 'total eve
minutes','total night minutes','total intl minutes'], axis=1)
# In[26]:
train.shape
# # Feature Scaling
# In[27]:
#Feature Scaling:
#Visualising the numerical data in histogram:
plt.rcParams["figure.figsize"] = [16,9]
train.hist(figsize = (16,10),color="red")
# In[28]:
#train['number customer service calls'] =train['number customer service
calls'].astype('int')
#train['number vmail messages'] =train['number vmail messages'].astype('int')
```

```
#train['total intl calls'] =train['total intl calls'].astype('int')
# In[29]:
# Normalisation for Train Dataset
train_skew_names=["number vmail messages","total intl calls","number customer service calls"]
minVec = train[train_skew_names].min().copy()
maxVec = train[train_skew_names].max().copy()
train[train_skew_names] = (train[train_skew_names]-minVec)/(maxVec-minVec)
#Performing feature scaling:
train_scaling_vars=['account length','total day calls','total day charge', 'total eve calls',
                   'total eve charge', 'total night calls','total night charge','total intl charge']
for i in train_scaling_vars:
    print(i)
    train[i]=(train[i]-train[i].mean())/train[i].std()

# In[30]:
train.head()
# In[31]:
#Visualizing the features after feature scaling:
plt.rcParams["figure.figsize"] = [16,9]
train.hist(figsize = (16,10),color="green")
# In[32]:
#Transform the variables dtype:
train['international plan'] = train['international plan'].astype('int')
train['Churn'] = train['Churn'].astype('int')
# In[33]:
train.dtypes
# # Looking at the proprotion of target class
# In[34]:
#Looking at the proportion of target class:
print(train.Churn.value_counts())
churn_proportion=sns.countplot(x="Churn", data=train)
total = float(len(train))
for p in churn_proportion.patches:
    height = p.get_height()
    churn_proportion.text(p.get_x()+p.get_width()/2.,
            height + 3,
            '{:1.2f}'.format(height/total),
            ha="center")
# In[35]:
train.describe(include="all")
# In[36]:
train.head()
# # Handling Test Data
# # Understanding Test Data
# In[37]:
#Now, Preparing the test data:
#EXploratory data analysis of test data:
test.head()
# In[38]:
test.shape
# In[39]:
test.dtypes
# # Missing Value Analysis on Test Data
# In[40]:
#Checking Missing value:
mvt=pd.DataFrame(test.isnull().sum())
mvt
# # Outlier Analysis and Imputation on Test Data
# In[41]:
```

```
#Box-plot visualisation:
test.plot(kind='box', figsize=[25,15])
# # Transforming Categorical variables on Test Data
# In[42]:
#Transfroming categorical variables:
test['voice mail plan'] = test['voice mail plan'].map(lambda x: x.strip())
test['international plan'] =test['international plan'].map(lambda x: x.strip())
test['Churn'] = test['Churn'].map(lambda x: x.strip())
# In[43]:
#Convert the categorical variable into numeric variable
test['voice mail plan'] = test['voice mail plan'].map({'no':0, 'yes':1})
test['international plan'] = test['international plan'].map({'no':0, 'yes':1})
test['Churn'] = test['Churn'].map({'False.':0, 'True.':1})
test.head()
# In[44]:
test["Churn"]=test["Churn"].astype(object)
# In[45]:
test.dtypes
# # Feature Selection on Test Data
# In[46]:
#Feature Selection: Removing multicollineared variables:
test_num_names = test.select_dtypes(['float64','int64']).columns
test_corr=test.loc[:,test_num_names]
print(test_corr.shape)
correlation_test=test_corr.corr()
correlation_test
# In[47]:
correlation_test.style.background_gradient(cmap='summer')
# In[48]:
test=test.drop(['voice mail plan', 'total day minutes', 'total eve minutes','total
night minutes','total intl minutes'], axis=1)
# In[49]:
test.shape
# In[50]:
#Transform the variable dtype:
test['international plan'] = test['international plan'].astype('int')
test['Churn'] = test['Churn'].astype('int')
# # Feature Scaling on Test Data
# In[51]:
#Feature Scaling:
test.hist(figsize=(16,15))
# In[52]:
test.columns
# In[53]:
test['number customer service calls'] =test['number customer service
calls'].astype('int')
test['number vmail messages'] =test['number vmail messages'].astype('int')
test['total intl calls'] = test['total intl calls'].astype('int')
# In[54]:
# Normalisation for Train Dataset
test_skew_vars= ['number customer service calls','number vmail messages','total
intl calls']
minVec = test[test_skew_vars].min().copy()
maxVec = test[test_skew_vars].max().copy()
test[test_skew_vars] = (test[test_skew_vars]-minVec)/(maxVec-minVec)
#Performing feature scaling on test data:
test_scaling_vars=['account length','total day calls', 'total day charge', 'total
eve calls',
                   'total eve charge', 'total night calls', 'total night charge',
'total intl charge',]
for i in test_scaling_vars:
    print(i)
```

```
    test[i]=(test[i]-test[i].mean())/test[i].std()
# In[55]:
test.describe(include="all")
# In[56]:
#Looking at the proportion of target class in test:
test["Churn"].value_counts()
# In[57]:
test.dtypes
# In[58]:
#Thus, prepared the test data.
# In[59]:
train.head()
# # MODEL BUILDING USING CLASIFICATION ALGORITHMS:
# In[60]:
#Creating train_x, train_y, test_x, test_y for modeling:
#train_x = predictors in train
#train_y = target in tarin
#test_x = predictors in test
#test_y = target in test
train_x=train.columns[0:12]
train_y=train["Churn"]
test_x=test.columns[0:12]
test_y=test["Churn"]
# # Logistic Regression
# In[61]:
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix, classification_report
# In[62]:
model_glm=sm.Logit(train_y,train[train_x]).fit()
model_glm.summary()
# # Prediction on test data using Logistic Regression model
# In[63]:
test['prediction_glm'] = model_glm.predict(test[test_x])
test['prediction_glm']
# # Evaluation of Logistic Regression
# In[64]:
#Confusion Matrix of Logistic Regression model:
cm_lm=confusion_matrix(test_y,test["prediction_glm"].round())
cm_lm
# In[65]:
#FNR:
(106*100)/(106+118)
# In[66]:
#Accuracy:
(1308+118)/(1308+135+106+118)
# In[67]:
print(test["Churn"].value_counts())
print(classification_report(test["Churn"], test["prediction_glm"].round()))
ax= plt.subplot()
sns.heatmap(cm_lm, annot=True, annot_kws={"size": 18}, fmt="d", ax = ax);
ax.set_xlabel('Predicted values');ax.set_ylabel('Actual values');
ax.set_title('Confusion Matrix of Logistic Regression');
ax.xaxis.set_ticklabels(['NO CHURN', 'CHURN']); ax.yaxis.set_ticklabels(['NO
CHURN', 'CHURN']);
# In[68]:
#test["prediction1"]=np.where(test["prediction_lm"]>0.5,1,0)
#test.head()
# In[69]:
#ROC Curve to evaluate the model:
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
#Computing auc
```

```
auc_glm = roc_auc_score(test["Churn"], test["prediction_glm"].round())
print('AUC: %.3f' % auc_glm)
#Computing false and true positive rates
fpr,
tpr,_=roc_curve(test["Churn"],test["prediction_glm"].round(),drop_intermediate=Fals
e)
print(fpr)
print(tpr)
plt.figure()
##Adding the ROC
plt.plot(fpr, tpr, color='red',lw=2, label='ROC curve')
##Random FPR and TPR
plt.plot([0, 1], [0, 1], color='blue', lw=2, linestyle='--')
##Title and label
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE OF LOGISTIC REGRESSION')
plt.show()
# # Random Forest
# In[70]:
#Model building using Random Forest:
from sklearn.ensemble import RandomForestClassifier
# In[71]:
model_rf=RandomForestClassifier(n_estimators=500).fit(train[train_x],train_y)
model_rf
# # Prediction on test using Random Forest model:
# In[72]:
#Prediction on test data using model_rf:
test['prediction_rf']=model_rf.predict(test[test_x])
test['prediction_rf']
# # Evaluation of Random Forest model
# In[73]:
#Confusion Matrix:
cm_rf=confusion_matrix(test["Churn"],test["prediction_rf"])
cm_rf
# In[74]:
#FNR:
(110*100)/(110+114)
# In[75]:
#Accuracy:
(1417+115)/(1417+26+109+115)
# In[76]:
print(test["Churn"].value_counts())
print(classification_report(test["Churn"], test["prediction_rf"].round()))
ax= plt.subplot()
sns.heatmap(cm_rf, annot=True, annot_kws={"size": 15}, fmt="d", ax = ax);
ax.set_xlabel('Predicted values');ax.set_ylabel('Actual values');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['NO CHURN', 'CHURN']); ax.yaxis.set_ticklabels(['NO
CHURN', 'CHURN']);
# In[77]:
# Calculate auc
auc_value = roc_auc_score(test["Churn"], test["prediction_rf"])
auc_value
# In[78]:
from sklearn.metrics import roc_auc_score
#calculate AUC
auc_rf = roc_auc_score(test["Churn"], test["prediction_rf"])
print('AUC: %.3f' % auc_rf)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(test["Churn"], test["prediction_rf"])
# plot no skill
```

```
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE OF RANDOM FOREST MODEL')
# show the plot
plt.show()
# # Finally, Looking at the importance of each feature in Random Forest model
# In[79]:
#Importance of features based on their impact on target:
pd.DataFrame({'Variable':train[train_x].columns,

'Importance':model_rf.feature_importances_}).sort_values('Importance',
ascending=False)
# In[80]:

##################################################################################
##################################################################################
```

Thank you

Regards,
Ranjith P