# CHURN REDUCTION

Shankar Samidala

24 June 2018

# Contents

# 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.

### Problem Statement-

The objective of this Case is to predict customer behaviour. We are providing you a public dataset that has customer usage pattern and if the customer has moved or not. We expect you to develop an algorithm to predict the churn score based on usage pattern.

### 1.2 Data

Table 1.1: Churn Reduction Train data (Columns: 1-5)

| State | Account Length | Area Code | Phone Number | International Plan |
|---|---|---|---|---|
| KS | 128 | 415 | 382-4657 | No |
| OH | 107 | 415 | 371-7191 | No |
| NJ | 137 | 415 | 358-1929 | No |
| OH | 84 | 408 | 375-9999 | Yes |
| OK | 75 | 415 | 330-6626 | Yes |

Table 1.2: Churn Reduction Train data (Columns: 6-10)

| Voice mail Plan | Number vmail Messages | Total day minutes | Total day calls | Total day charge |
|---|---|---|---|---|
| Yes | 25 | 265.1 | 110 | 45.07 |
| Yes | 26 | 161.6 | 123 | 27.47 |
| No | 0 | 243.4 | 114 | 41.38 |
| No | 0 | 299.4 | 71 | 50.90 |
| No | 0 | 166.7 | 113 | 28.34 |

Table 1.3: Churn Reduction Train data (Columns: 11-15)

| Total eve minutes | Total eve calls | Total eve charge | Total night minutes | Total night calls |
|---|---|---|---|---|
| 197.4 | 99 | 16.78 | 244.7 | 91 |
| 195.5 | 103 | 16.62 | 254.4 | 103 |
| 121.2 | 110 | 10.30 | 162.6 | 104 |
| 61.9 | 88 | 5.26 | 196.9 | 89 |
| 148.3 | 122 | 12.61 | 186.9 | 121 |

Table 1.4: Churn Reduction Train data (Columns: 16-21)

| Total night charge | Total intl minutes | Total intl calls | Total intl charge | Number Customer Service Calls | Churn |
|---|---|---|---|---|---|
| 11.01 | 10.0 | 3 | 2.70 | 1 | False. |
| 11.45 | 13.7 | 3 | 3.70 | 1 | False. |
| 7.32 | 12.2 | 5 | 3.29 | 0 | False. |
| 8.86 | 6.6 | 7 | 1.78 | 2 | False. |
| 8.41 | 10.1 | 3 | 2.73 | 3 | False. |

The predictors provided are as follows

- account length
- international plan
- voicemail plan
- number of voicemail messages
- total day minutes used
- day calls made
- total day charge
- total evening minutes
- total evening calls
- total evening charge
- total night minutes
- total night calls
- total night charge
- total international minutes used
- total international calls made
- total international charge

**Target Variable :**

Churn(move) We have to predict whether the customer will move or not.

# Chapter 2

# Methodology

## 2.1 Pre Processing

Data pre processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre processing is a proven method of resolving such issues. Data pre processing prepares raw data for further processing.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data pre-processing includes cleaning ,outlier deduction, normalization, feature extraction and selection , etc. The product of data pre-processing is the final data. This is often called as Exploratory Data Analysis .

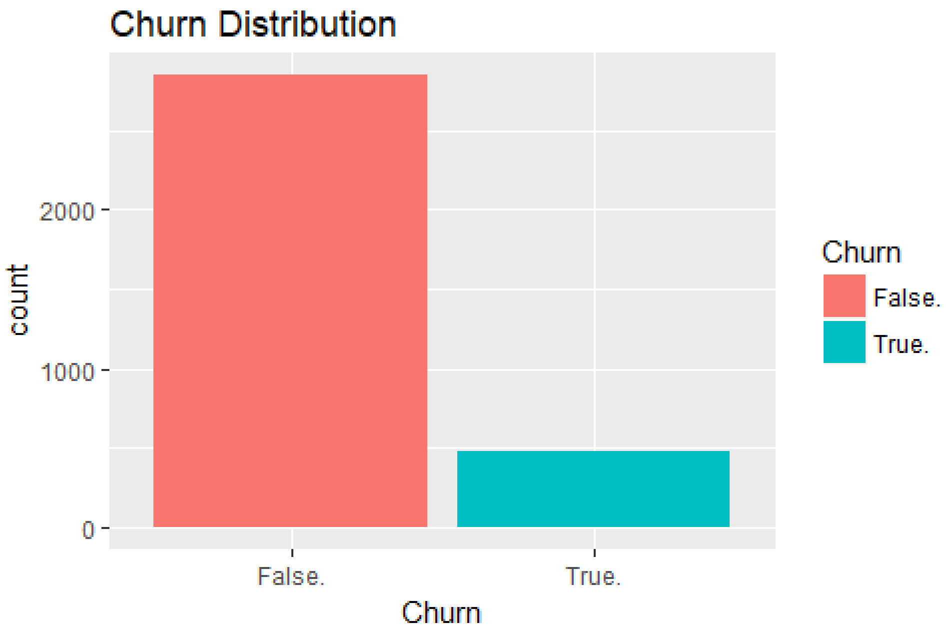
### 2.1.1 Missing value analysis

Missing data can occur because of nonresponse: no information is provided for one or more items or for a whole unit ("subject"). Some items are more likely to generate a nonresponse than others: for example items about private subjects such as income. Attrition ("Dropout") is a type of missingness that can occur in longitudinal studies - for instance studying development where a measurement is repeated after a certain period of time. Missingness occurs when participants drop out before the test ends and one or more measurements are missing.

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned. Another concern is that the assumptions behind many statistical procedures are based on complete cases, and missing values can complicate the theory required.

**Example.** In evaluating a treatment for cancer, several variables are measured. However, not all measurements are available for every patient. The patterns of missing data are displayed, tabulated, and found to be random. An EM (expectation-maximization) analysis is used to estimate the means, correlations, and covariances. It is also used to determine that the data are missing completely at random. Missing values are then replaced by imputed values and saved into a new data file for further analysis.

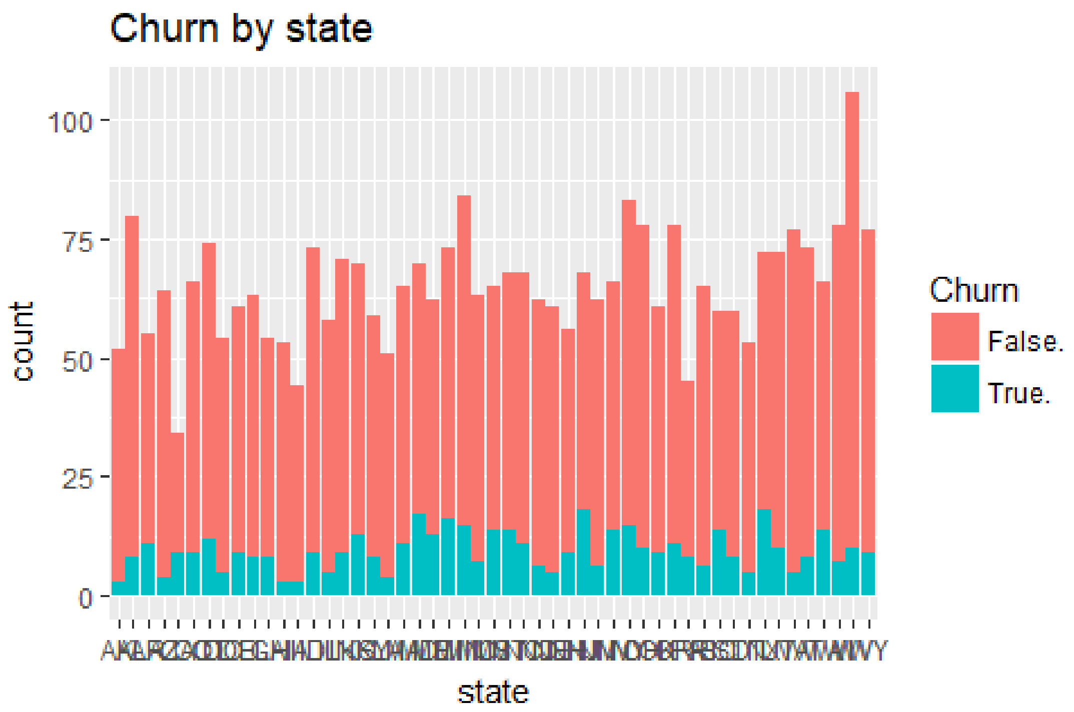
```
ggplot(data_tr,aes(Churn,fill=Churn))+  geom_bar()+ggtitle("Churn Distribution")
```



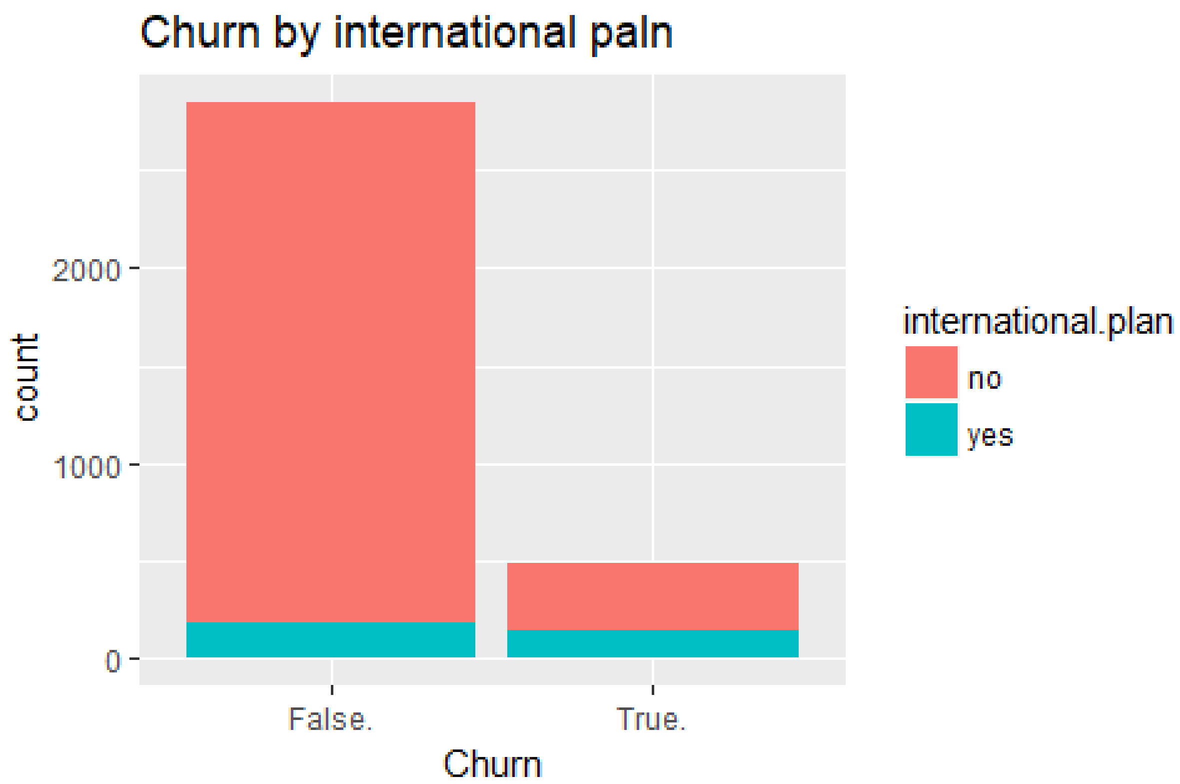

**Distribution of Churn with each categorical variable in data.**

**Conclusion 1 = Imbalanced data - Lesser data points in "True" Churn category**

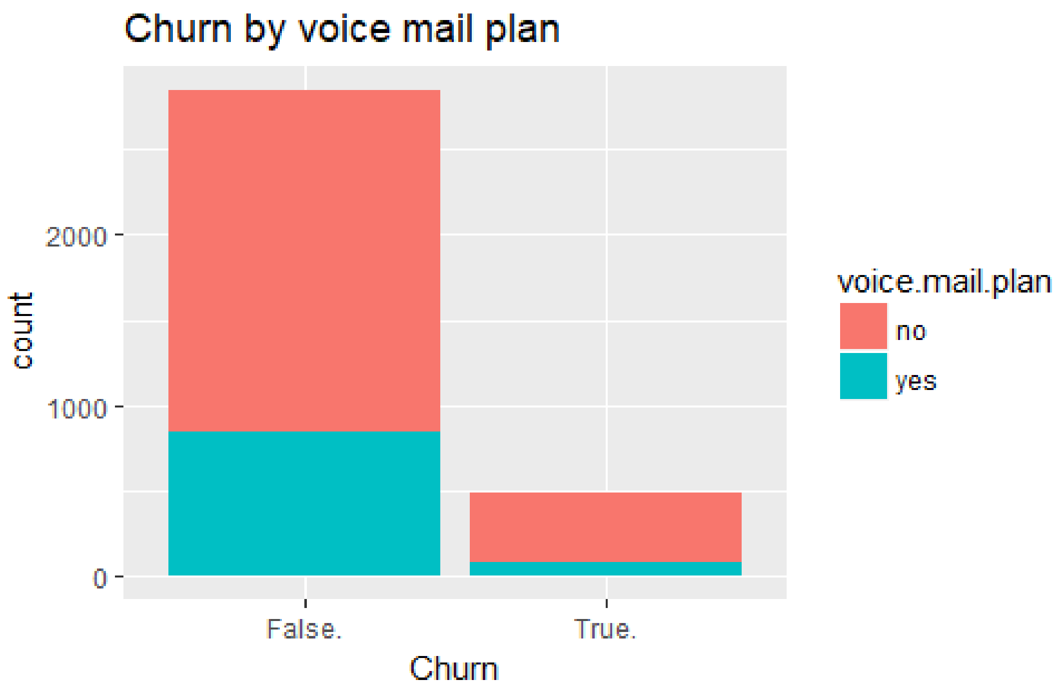
```
ggplot(data_tr,aes(state,fill=Churn))+  geom_bar(position="dodge")+ggtitle("Churn by state")
```



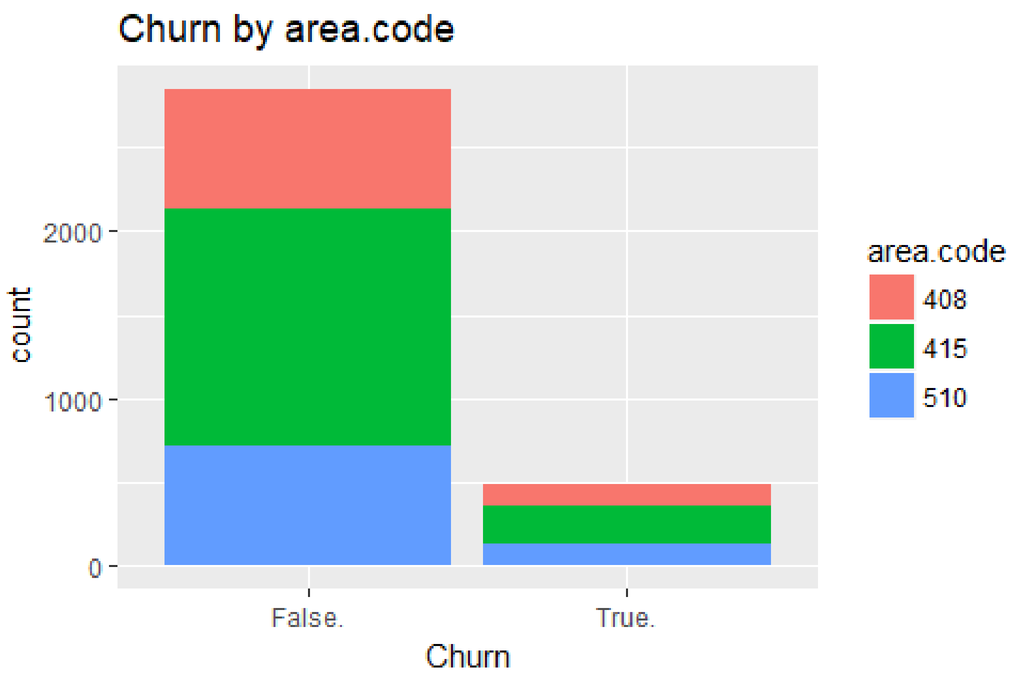

```
ggplot(data_tr,aes(Churn,fill=international.plan))+  geom_bar()+ggtitle("Churn by international paln")
```



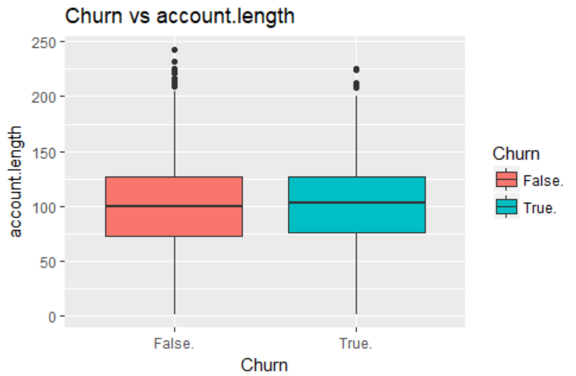
```
ggplot(data_tr,aes(Churn,fill=voice.mail.plan))+  geom_bar()+ggtitle("Churn by voice mail plan")
```

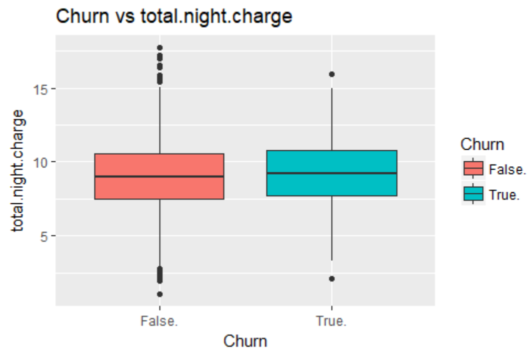


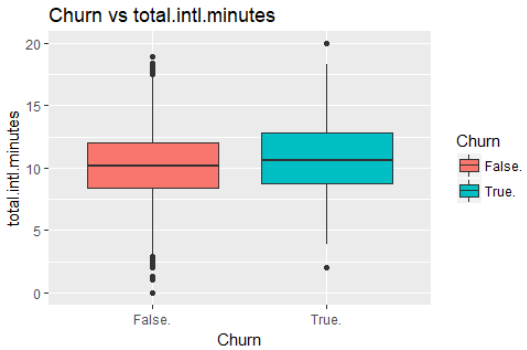

```
ggplot(data_tr,aes(Churn,fill=area.code))+  geom_bar()+ggtitle("Churn by area.code")
```



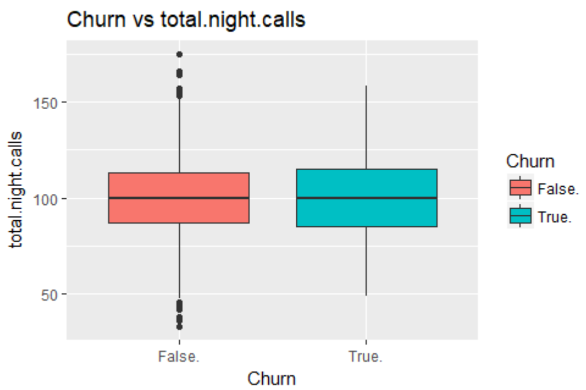

Let check for outliers and how they are effecting on the target variables

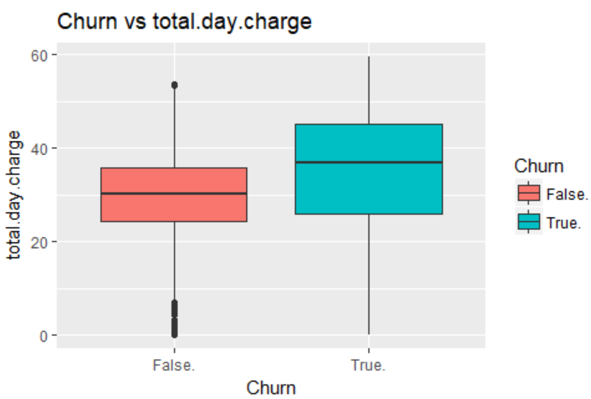

Churn vs account.length
account.length
250
200
150
100
50
0
False.
True.
Churn
Churn
False.
True.


Churn vs total.night.charge
total.night.charge
15
10
5
False.
True.
Churn
Churn
False.
True.


Churn vs total.intl.minutes
total.intl.minutes
20
15
10
5
0
False.
True.
Churn
Churn
False.
True.


Churn vs total.eve.minutes
total.eve.minutes
300
200
100
0
False.
True.
Churn
Churn
False.
True.


Churn vs total.night.calls
total.night.calls
150
100
50
False.
True.
Churn
Churn
False.
True.


Churn vs total.day.charge
total.day.charge
60
40
20
0
False.
True.
Churn
Churn
False.
True.

### 2.1.2 Feature Selection

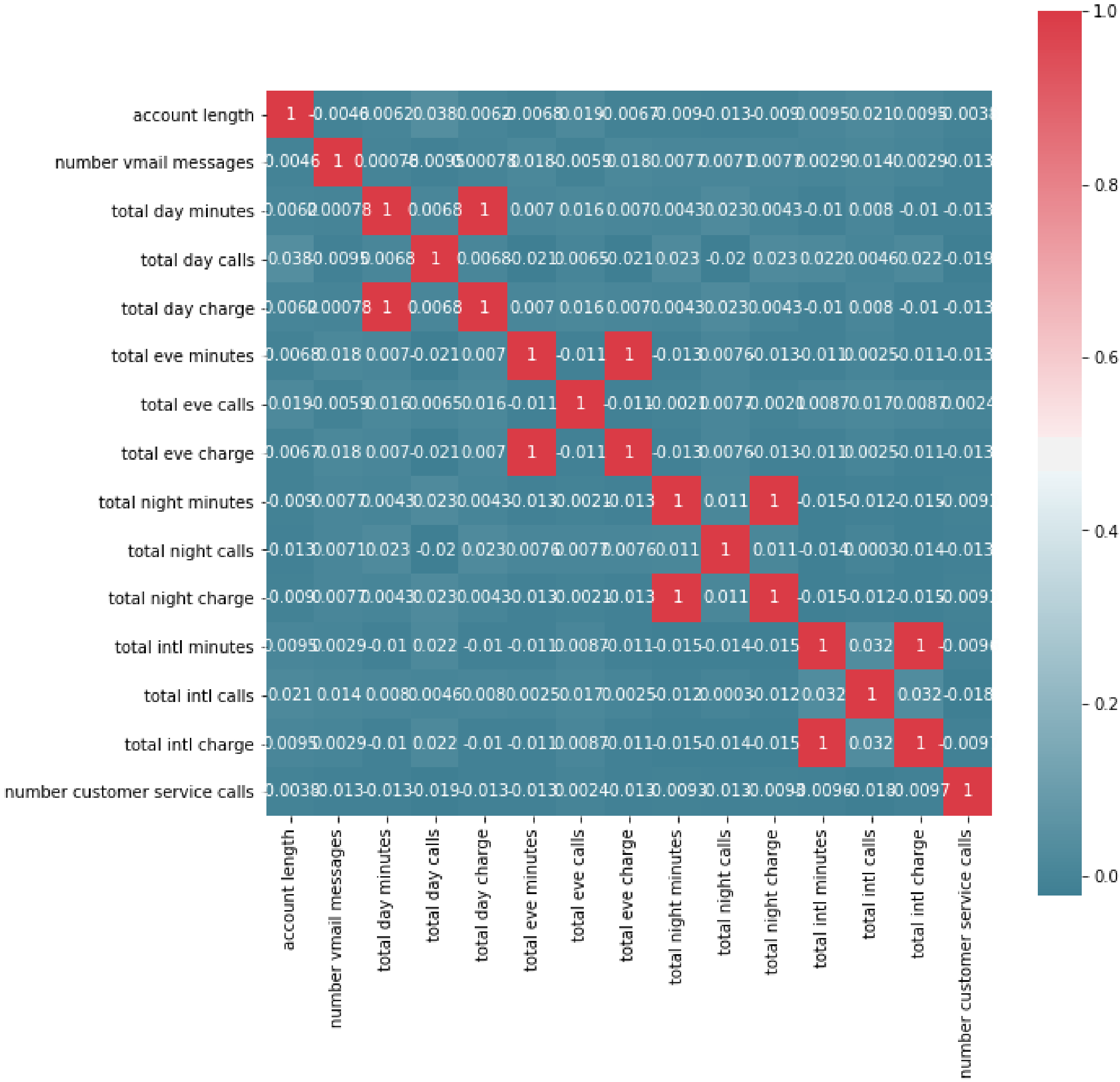
Feature selection is critical to building a good model for several reasons. One is that feature selection implies some degree of *cardinality reduction*, to impose a cutoff on the number of attributes that can be considered when building a model. Data almost always contains more information than is needed to build the model, or the wrong kind of information. For example, you might have a dataset with 500 columns that describe the characteristics of customers; however, if the data in some of the columns is very sparse you would gain very little benefit from adding them to the model, and if some of the columns duplicate each other, using both columns could affect the model.

Not only does feature selection improve the quality of the model, it also makes the process of modeling more efficient. If you use unneeded columns while building a model, more CPU and memory are required during the training process, and more storage space is required for the completed model. Even if resources were not an issue, you would still want to perform feature selection and identify the best columns, because unneeded columns can degrade the quality of the model in several ways:

- Noisy or redundant data makes it more difficult to discover meaningful patterns.
- If the data set is high-dimensional, most data mining algorithms require a much larger training data set.

### (A)Correlation analysis

A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to find and predict which pairs have the highest correlation. If two variables are highly correlated we have two drop one variable .




Correlation Matrix

The above correlation matrix shows some interesting results as follows :

1. Total day minutes and total day charge are very highly correlated.

2. Total eve minutes and total eve charge are very highly correlated.

3. Total night minutes and total night charge are very highly correlated.

4. Total intl minutes and total intl charge are very highly correlated.Now we have to remove the highly correlated variable so that our model can perform well and it gives much accuracy.

## (B)Chi square test of Independence

The Chi-Square test of independence is used to determine if there is a significant relationship between two categorical variables. The frequency of each category for one variable is compared across the categories of the second variable. It is used to determine whether there is a significant association between the two variables. Uses contingency table for better representation. We conduct chi square test of independence for each of the categorical variable with our target variable that is churn so that we will remove the variable that is independent with target variable. Scores of chi square test of independence of each categorical variable is as follows

[1] **"state"**

Pearson's Chi-squared test

data:  table(fa_dt$Churn, fa_dt[, i])
X-squared = 83.044, df = 50, p-value = 0.002296

[1] "**area.code"**

Pearson's Chi-squared test

data:  table(fa_dt$Churn, fa_dt[, i])
X-squared = 0.17754, df = 2, p-value = 0.9151

[1] **"phone.number"**

Pearson's Chi-squared test

data:  table(fa_dt$Churn, fa_dt[, i])
X-squared = 3333, df = 3332, p-value = 0.4919

[1] **"international.plan"**

Pearson's Chi-squared test with Yates' continuity correction

data:  table(fa_dt$Churn, fa_dt[, i])
X-squared = 222.57, df = 1, p-value < 2.2e-16

[1] **"voice.mail.plan"**

Pearson's Chi-squared test with Yates' continuity correction

data:  table(fa_dt$Churn, fa_dt[, i])
X-squared = 34.132, df = 1, p-value = 5.151e-09

If the p value of the categorical variable is less than 0.05 it is representing the Alternative hypothesis. If the p value is grater than 0.05 it is representing Null Hypothesis. We have to drop the variables which are having null hypothesis because those are not carrying much information to explain the target variable. We will consider only the variables having Alternative hypothesis.

Therefore from both the correlation analysis and chi square test of independence we got some variable which are not carrying much information. We have to delete those variables

Numerical: total day minutes, total eve minutes, total night minutes, total intl minutes
Categorical: phone number,area.code

### 2.1.3 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre processing steps. If training an algorithm using different features and some of them are off the scale in their magnitude, then the results might be dominated by them. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. We use normalization here for feature scaling.

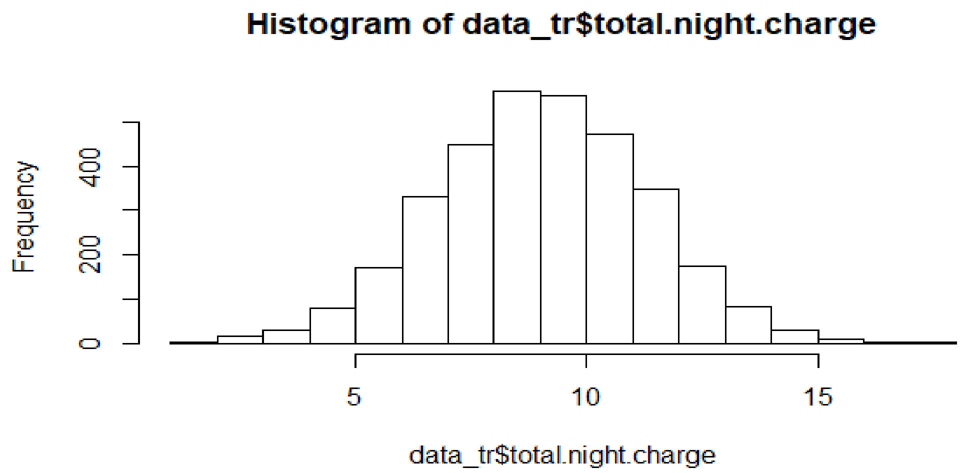
Normalization also called Min-Max scaling. It is the process of reducing unwanted variation either within or between variables. Normalization brings all of the variables into proportion with one another. It transforms data into a range between 0 and 1. We have to see the variables that are scattered highly and apply normalization. We normalize the following variables in our data so that we can process to the modeling phase.

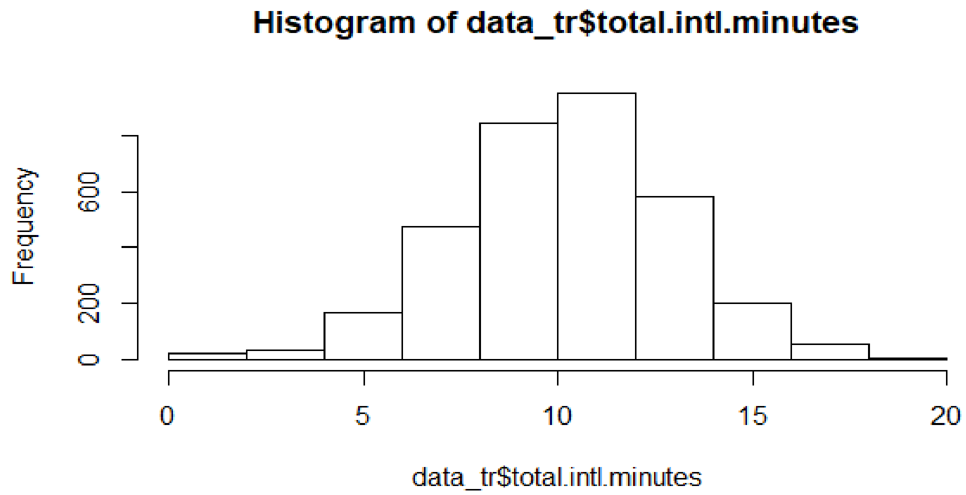
Variables: account length, number vmail messages, total day calls, total day charge, total eve calls, total eve charge, total night calls, total night charge, total intl calls, total intl charge, number customer service calls.

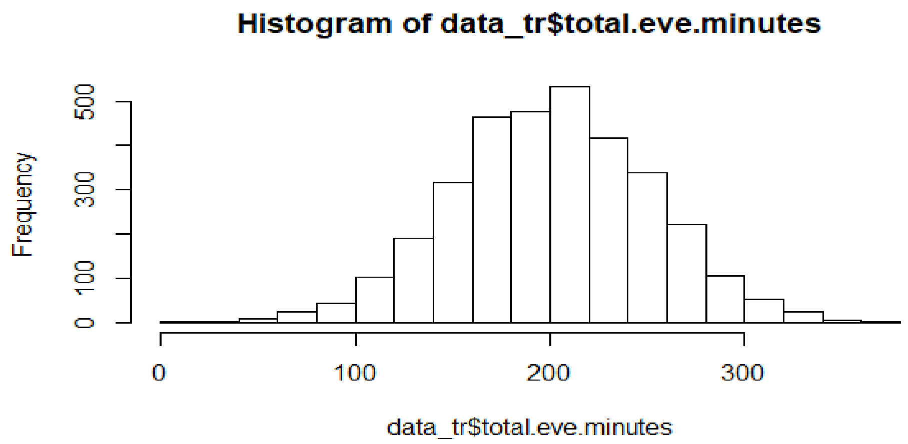
Formulae used for normalization is

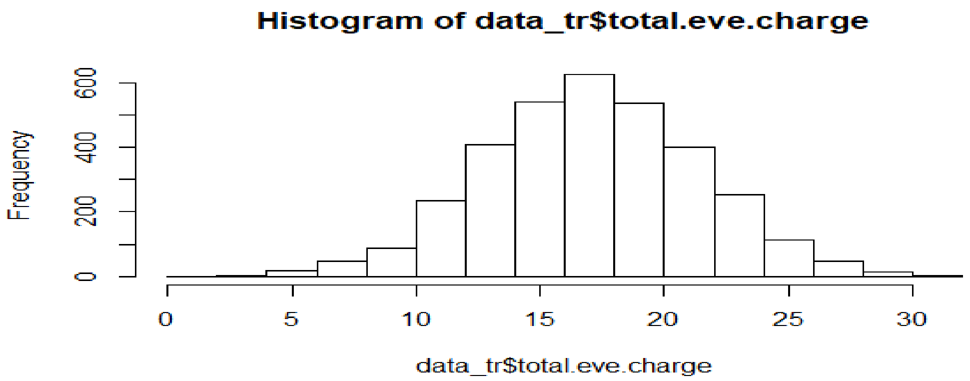
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

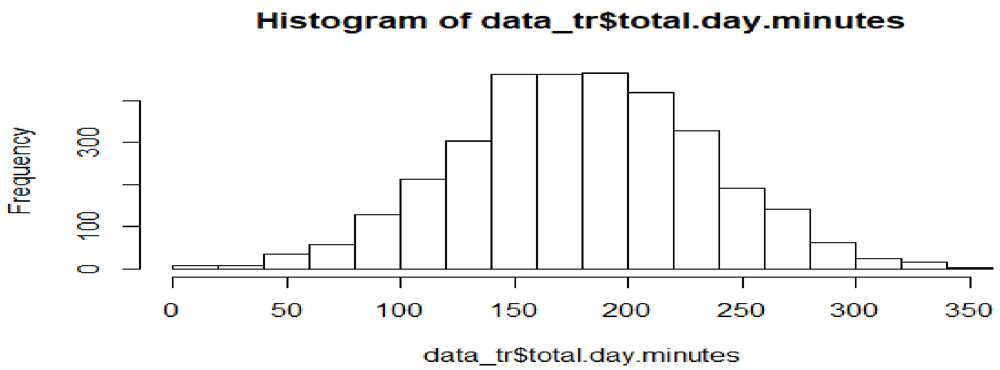
Data Distribution before Normalization

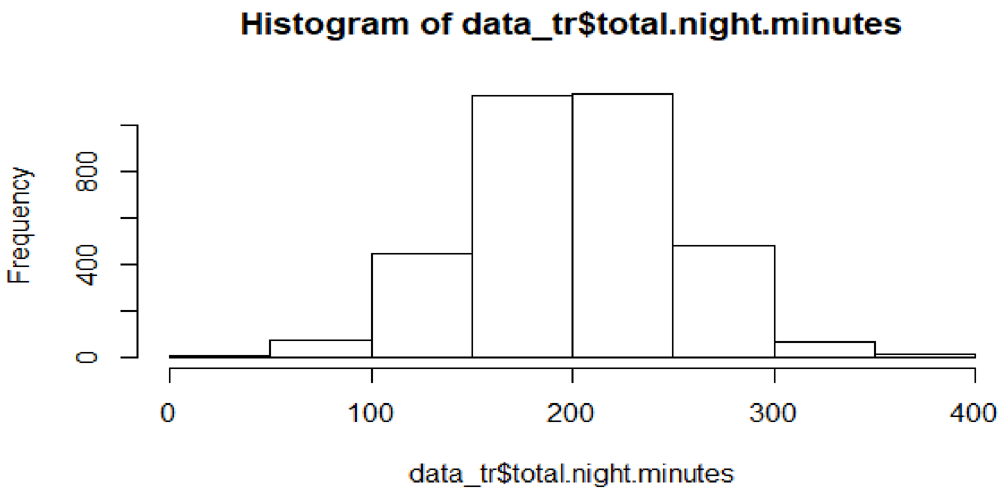


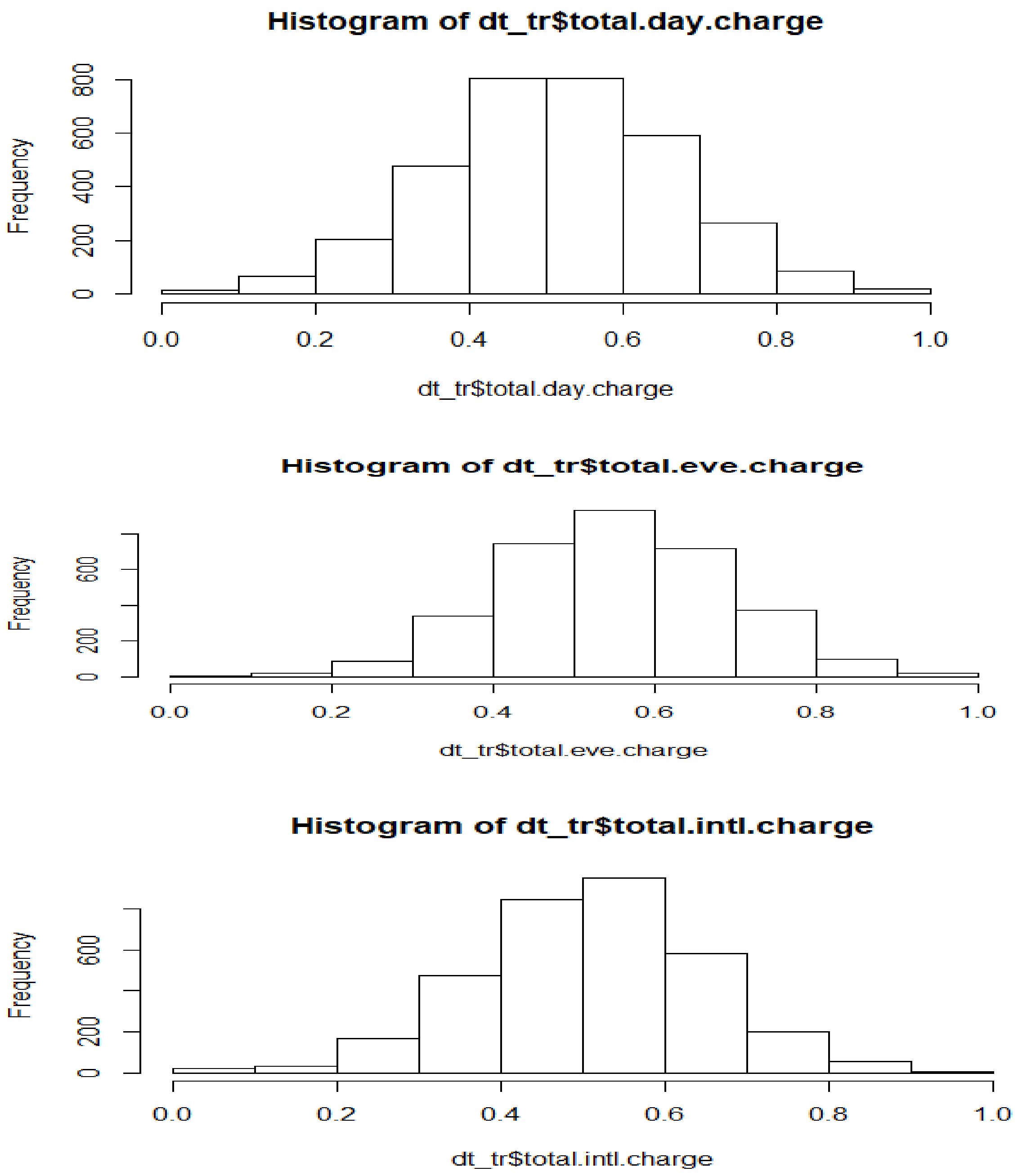

Histogram of data_tr$total.eve.charge
Frequency
0
200
400
600
0
5
10
15
20
25
30
data_tr$total.eve.charge


Histogram of data_tr$total.day.minutes
Frequency
0
100
300
0
50
100
150
200
250
300
350
data_tr$total.day.minutes


Histogram of data_tr$total.night.minutes
Frequency
0
400
800
0
100
200
300
400
data_tr$total.night.minutes

**Data after normalization**

We have applied normalization to convert the whole data in a single scale




Now the data is ready for applying models. Now we will use several machine learning classifications models and predicts who will and churn who will not churn with highest accuracy.

# 2.2 Modeling

### 2.2.1 Model Selection

**Stratified Cross Validation( stratified sampling) - Since the Response values are not balanced**

Our data is already divided into two sets train and test using stratified sampling.

After pre processing we will be using some classifications models on our processed data to predict the target variable.

**2.2.2 Decision Tree:** Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. It can be used for classification and regression. It is a Supervised machine learning algorithm. Accept continuous and categorical variables as independent variables. Extremely easy to understand by the business users. There are many types of decision trees.

**Categorical Variable Decision Tree:** Decision Tree which has categorical target variable then it called as categorical variable decision tree. Example:- a scenario of student problem, where the target variable was “Student will Pass or not” i.e. YES or NO.

**Continuous Variable Decision Tree:** Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

We are using C5.0 model which is entropy based. When we applied this model on our train data, we got certain rules which is provided in a text file. You can also visualize the decision tree with the dot in python . The accuracy of the model as following

```
Confusion Matrix and Statistics

   prd_c50
       1    2
  1 1354   83
  2   59  173

               Accuracy : 0.9149
                 95% CI : (0.9005, 0.9279)
    No Information Rate : 0.8466
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6593
 Mcnemar's Test P-Value : 0.05359

            Sensitivity : 0.9582
            Specificity : 0.6758
         Pos Pred Value : 0.9422
         Neg Pred Value : 0.7457
             Prevalence : 0.8466
         Detection Rate : 0.8113
   Detection Prevalence : 0.8610
      Balanced Accuracy : 0.8170

       'Positive' Class : 1

> cf_c50
   prd_c50
       1    2
  1 1354   83
  2   59  173
> rc_smote=roc.curve(test$Churn,prd_c50)
> 59+173
[1] 232
> 59/232
[1] 0.2543103
```

**2.2.3 Random Forest :** Random Forest or decision tree forests is an ensemble learning method for classification, regression and other tasks. Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning.

We can see certain rules of random forest in the R code. The accuracy of the model is as follows.

```
> confusionMatrix(cf_rf)
Confusion Matrix and Statistics

   rf_prd
       1    2
  1 1280  157
  2   43  189

               Accuracy : 0.8802
                 95% CI : (0.8636, 0.8954)
    No Information Rate : 0.7927
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5849
 Mcnemar's Test P-Value : 1.346e-15

            Sensitivity : 0.9675
            Specificity : 0.5462
         Pos Pred Value : 0.8907
         Neg Pred Value : 0.8147
             Prevalence : 0.7927
         Detection Rate : 0.7669
   Detection Prevalence : 0.8610
      Balanced Accuracy : 0.7569

       'Positive' Class : 1

> cf_rf
   rf_prd
       1    2
  1 1280  157
  2   43  189
> 43+189
[1] 232
> 43/232
[1] 0.1853448
```

**2.2.4 Logistic Regression:** 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. it is the appropriate regression analysis to conduct when the dependent variable is binary cases where the dependent variable has more than two outcome categories may be analyzed in multinomial logistic regression or if the multiple categories are ordered, in ordinal logistic regression.

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6382  -0.7316  -0.3255   0.7245   2.8095

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -9.09402    0.81197 -11.200  < 2e-16 ***
state2                          0.37096    0.68858   0.539 0.590076
state3                          1.59329    0.69155   2.304 0.021226 *
state4                          1.32385    0.67725   1.955 0.050613 .
state5                          2.96068    0.73822   4.011 6.06e-05 ***
state6                          1.24851    0.66379   1.881 0.059987 .
state7                          1.33205    0.68867   1.934 0.053084 .
state8                          0.59924    0.71909   0.833 0.404658
state9                          1.71095    0.67672   2.528 0.011461 *
state10                         1.23661    0.71050   1.740 0.081772 .
state11                         1.66407    0.68389   2.433 0.014964 *
state12                         0.19378    0.80761   0.240 0.810373
state13                         0.31989    0.86369   0.370 0.711106
state14                         1.63007    0.63756   2.557 0.010566 *
state15                         2.37393    0.65692   3.614 0.000302 ***
state16                         0.83443    0.70463   1.184 0.236330
state17                         1.37446    0.67018   2.051 0.040278 *
state18                         1.97406    0.66251   2.980 0.002886 **
state19                         0.58996    0.71967   0.820 0.412352
state20                         1.31518    0.68898   1.909 0.056278 .
state21                         1.19323    0.66860   1.785 0.074313 .
state22                         1.94515    0.65523   2.969 0.002991 **
state23                         1.52737    0.66627   2.292 0.021882 *
state24                         1.36297    0.65792   2.072 0.038300 *
state25                         1.05763    0.67191   1.574 0.115475

state26                         1.37020    0.66625   2.057 0.039727 *
state27                         2.27986    0.64826   3.517 0.000437 ***
state28                         1.25066    0.67190   1.861 0.062691 .
state29                         0.83216    0.68184   1.220 0.222288
state30                         0.82994    0.69885   1.188 0.234999
state31                         2.45271    0.66413   3.693 0.000221 ***
state32                         2.93256    0.63462   4.621 3.82e-06 ***
state33                         2.20853    0.67760   3.259 0.001117 **
state34                         1.70196    0.65865   2.584 0.009766 **
state35                         1.41843    0.64709   2.192 0.028378 *
state36                         2.07121    0.63901   3.241 0.001190 **
state37                         1.80641    0.64406   2.805 0.005036 **
state38                         1.91680    0.66238   2.894 0.003806 **
```

```
state39                         0.46454    0.72493   0.641 0.521649
state40                         0.24528    0.75978   0.323 0.746819
state41                         1.74968    0.69618   2.513 0.011962 *
state42                         0.44269    0.73717   0.601 0.548155
state43                         1.36729    0.66749   2.048 0.040521 *
state44                         2.58775    0.63307   4.088 4.36e-05 ***
state45                         1.52647    0.66359   2.300 0.021429 *
state46                        -0.58112    0.78391  -0.741 0.458509
state47                         0.79892    0.67948   1.176 0.239683
state48                         2.70481    0.66941   4.041 5.33e-05 ***
state49                         0.18350    0.75681   0.242 0.808424
state50                         1.12087    0.64520   1.737 0.082345 .
state51                         0.02844    0.69790   0.041 0.967493
account.length                  1.01793    0.29145   3.493 0.000478 ***
international.plan2             2.52522    0.12628  19.996  < 2e-16 ***
voice.mail.plan2                0.33293    0.14471   2.301 0.021411 *
number.vmail.messages          -0.82190    0.28049  -2.930 0.003387 **
total.day.calls                 0.53240    0.35260   1.510 0.131058
total.day.charge                3.35577    0.30424  11.030  < 2e-16 ***
total.eve.calls                 0.54686    0.35524   1.539 0.123704
total.eve.charge                2.83403    0.33608   8.433  < 2e-16 ***
total.night.calls              -0.68730    0.34463  -1.994 0.046116 *
total.night.charge              0.99101    0.36853   2.689 0.007164 **
total.intl.calls               -1.72406    0.38885  -4.434 9.26e-06 ***
total.intl.charge               3.18162    0.35596   8.938  < 2e-16 ***
number.customer.service.calls   4.00390    0.27970  14.315  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4541.3  on 3324  degrees of freedom
Residual deviance: 3054.4  on 3261  degrees of freedom
AIC: 3182.4

Number of Fisher Scoring iterations: 5

    logit_prd
        0    1
  1 1194  239
  2  108  128
> 108+128
[1] 236
> 108/236
[1] 0.4576271
```

**2.2.5 K-NN Implementation:** K-Nearest Neighbours algorithm (KNN) is a non-parametric method used for classification and regression. KNN is a type of instance-based learning or lazy learning and simplest of all machine learning algorithms. Even with such simplicity, it can give highly competitive results. It is more widely used in classification problems in the industry. KNN fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time. K-NN is a lazy learner because it doesn't learn a discriminative function from the training data but "memorizes" the training dataset instead. Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbours) and summarizing the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidea distance.

Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.

```
Confusion Matrix and Statistics

knn_prd    1    2
      1 1243  143
      2  190   93

               Accuracy : 0.8005
                 95% CI : (0.7805, 0.8194)
    No Information Rate : 0.8586
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.2414
 Mcnemar's Test P-Value : 0.01171

            Sensitivity : 0.8674
            Specificity : 0.3941
         Pos Pred Value : 0.8968
         Neg Pred Value : 0.3286
             Prevalence : 0.8586
         Detection Rate : 0.7448
   Detection Prevalence : 0.8304
      Balanced Accuracy : 0.6307

       'Positive' Class : 1
```

**Naive Bayes:** It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

```
> confusionMatrix(cm_nb)
Confusion Matrix and Statistics

        predicted
observed    1    2
       1 1253  180
       2  117  119

               Accuracy : 0.822
                 95% CI : (0.8028, 0.8401)
    No Information Rate : 0.8209
    P-Value [Acc > NIR] : 0.4645528

                  Kappa : 0.3406
 Mcnemar's Test P-Value : 0.0003212

            Sensitivity : 0.9146
            Specificity : 0.3980
         Pos Pred Value : 0.8744
         Neg Pred Value : 0.5042
             Prevalence : 0.8209
         Detection Rate : 0.7507
   Detection Prevalence : 0.8586
      Balanced Accuracy : 0.6563

       'Positive' Class : 1
```

# Chapter 3

# Conclusion

### 3.1 Model Evaluation

Model evaluation is done on basis of error metrics. Error metrics explain the performance of a model. An important aspect of error metrics is their capability to discriminate among model results. Simply, building a predictive model is not our final thing. Creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check accuracy or other metric of the model prior to computing predicted values. In our data as we applied classification models we have error metrics like confusion matrix out of which we can check specificity, recall, false negative rate, false positive rate. We will evaluate error metrics for each of the model that is applied

Confusion matrix has True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN)

**True Positive is the number of correct predictions that an instance is Yes,**

**False Negative is the number of incorrect predictions that an instance is No,**

**False Positive is the number of incorrect of predictions that an instance Yes, and**

**True Negative is the number of correct predictions that an instance is No.**

**Decision Tree:**

Confusion Matrix :

```
     prd_c50
        1    2
1 1354   83
2   59  173
```

Accuracy = 0.91

Sensitivity or Recall: = 0.95

Specificity: = 0.67

False Negative rate: = 0.25

**Random Forest:**

```
   rf_prd
        1    2
  1 1280  157
  2   43  189
```

Accuracy = 0.88

Sensitivity or Recall: = 0.96

Specificity: = 0.54

False Negative rate:0.18

**Logistic Regression:**

```
   logit_prd
       0    1
1 1194  239
2  108  128
```

Accuracy = 0.79

False Negative rate:0.45

**KNN Implementation:**

Confusion Matrix

```
knn_prd    1    2
      1 1243  143
      2  190   93
```

Accuracy = 0.80

Sensitivity or Recall: = 0.86

Specificity:  = 0.39

False Negative rate:0.67

**Naïve Bayes:**

Confusion Matrix

```
        predicted
observed    1    2
       1 1253  180
       2  117  119
```

Accuracy = 0.82

Sensitivity or Recall: = 0.91

Specificity:  = 0.39

False Negative rate:0.49

## 3.2 Model Selection

We can observe that from all the above models they are performing comparatively on average and therefore we have to select either decision tree or random forest classifier models for better prediction

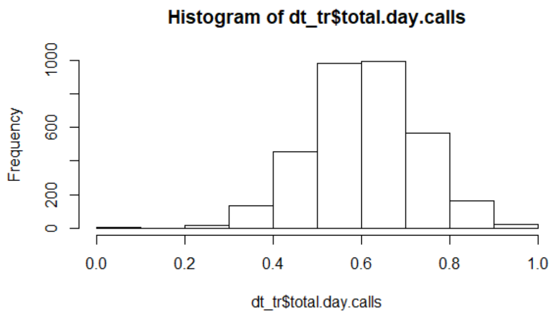
**REPORT**

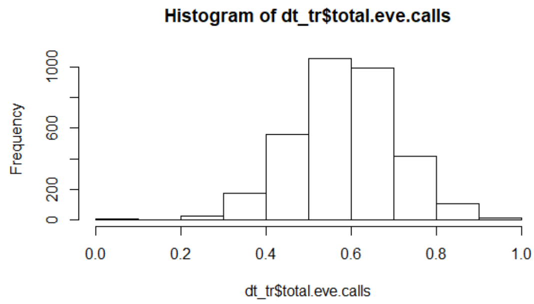
1. High churn rate among customers with international plans.

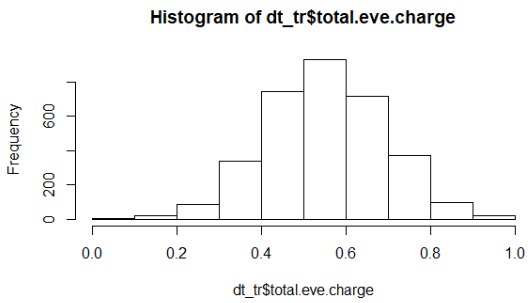
2. Customers with four or more service calls are more likely to leave the company. Companies should improve their service call centers to resolve customer issues in fewer than three calls.

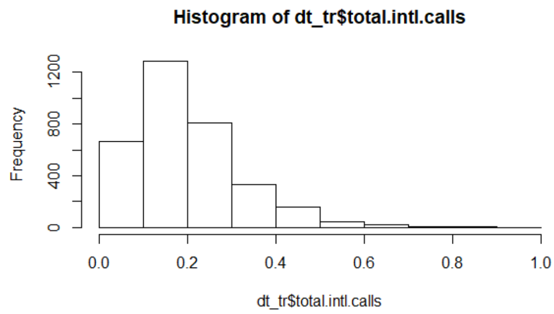
# Appendix A - Extra Figures

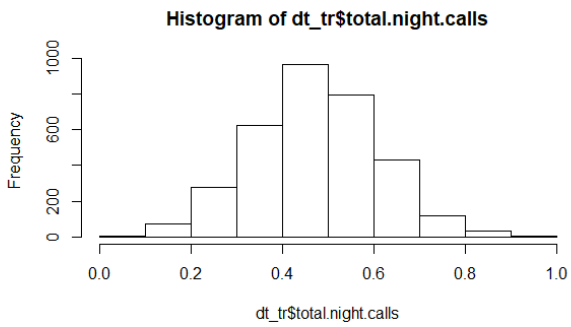
## Normality graphs



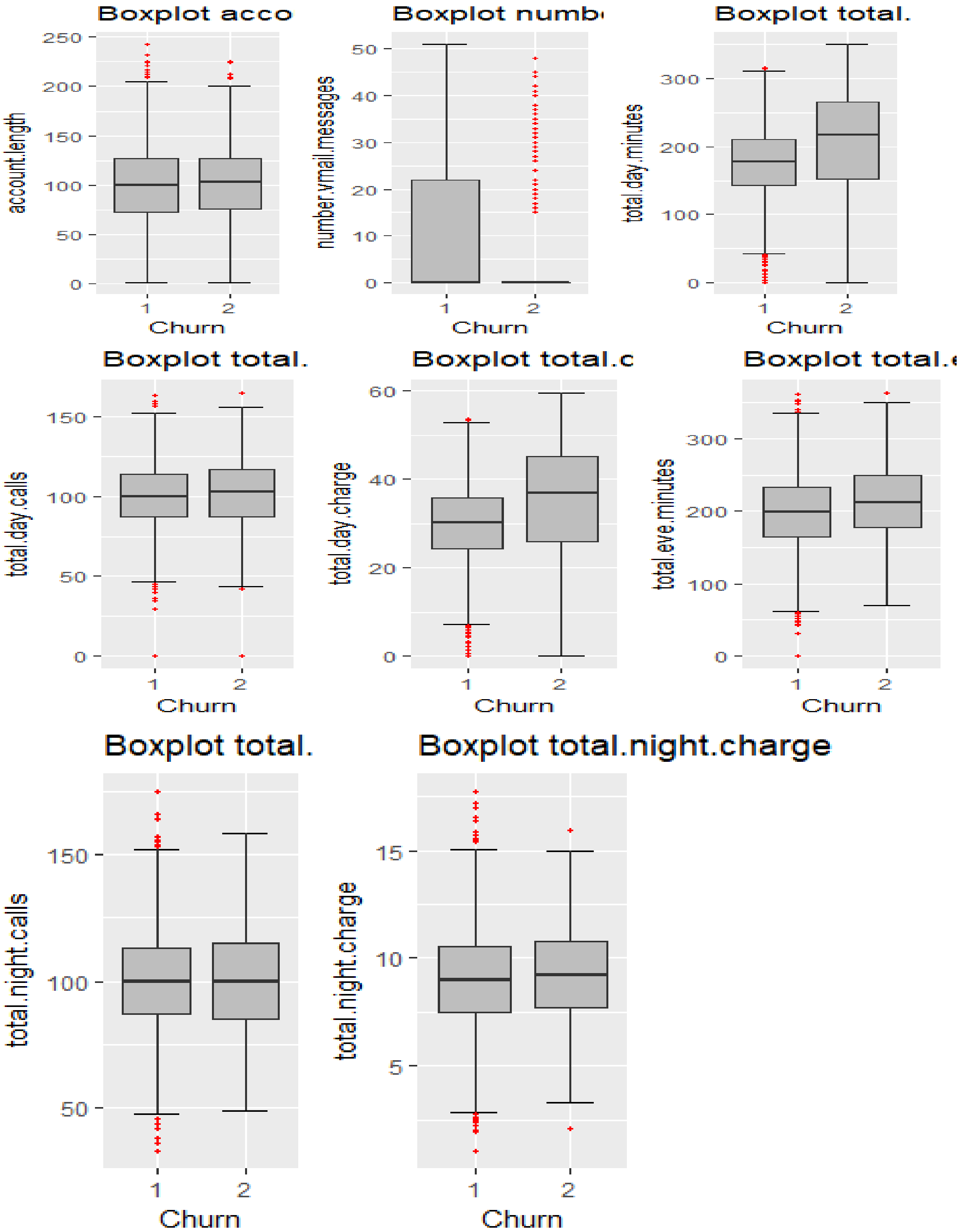



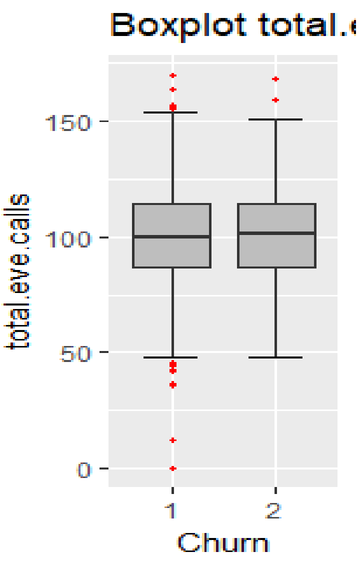



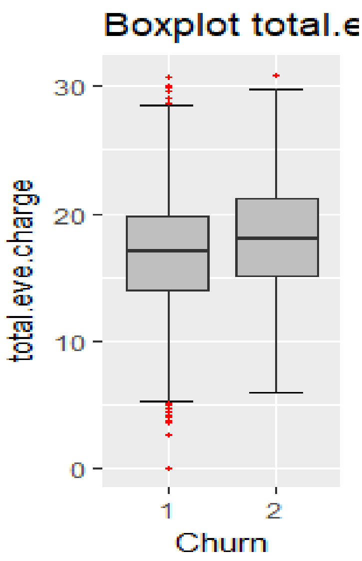



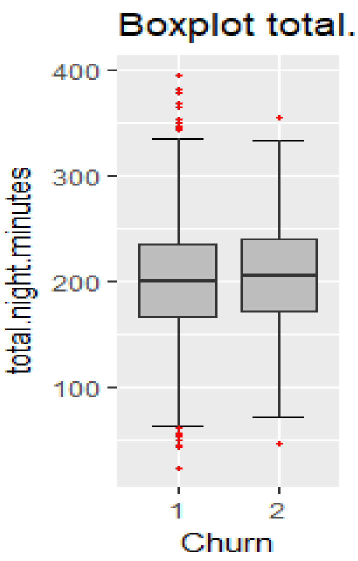



## Outliers graphs

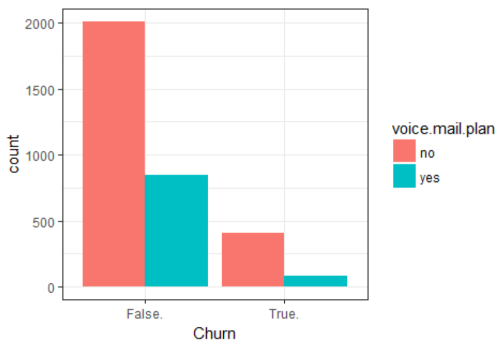


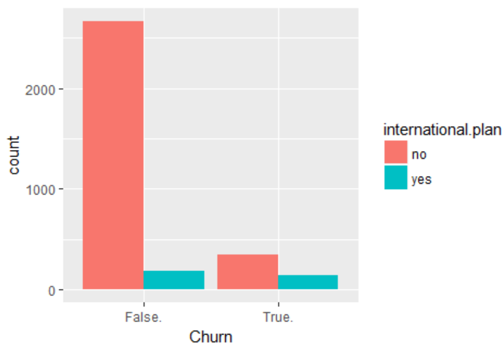


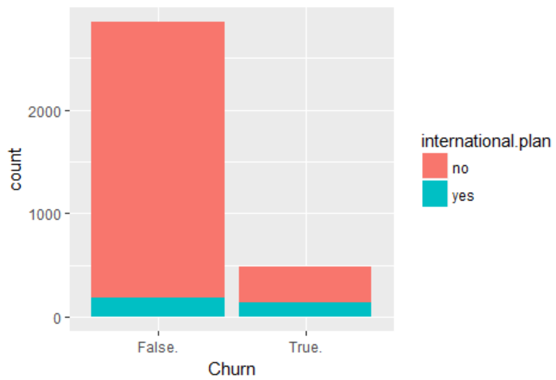



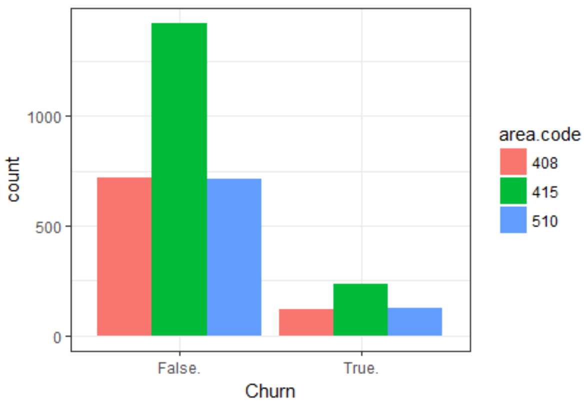




## CHURN DISTRIBUTION

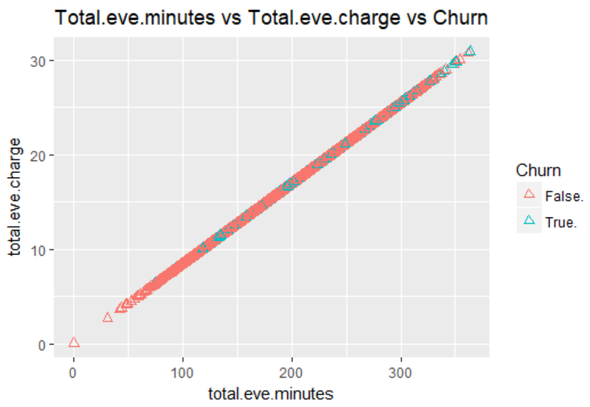


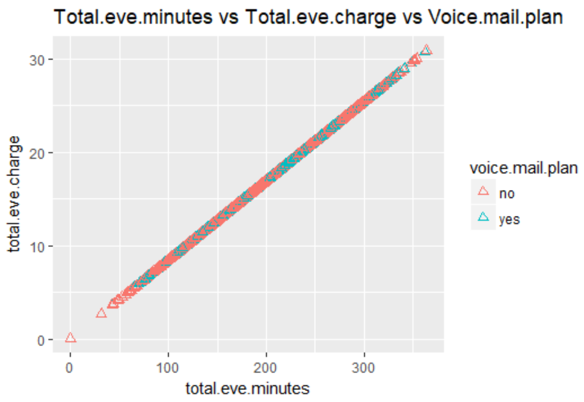



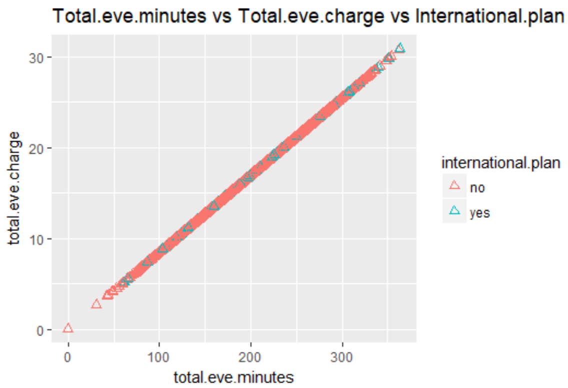



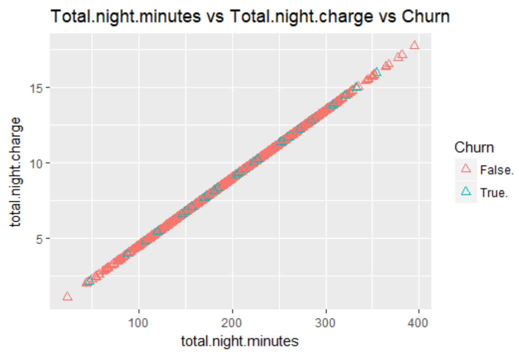



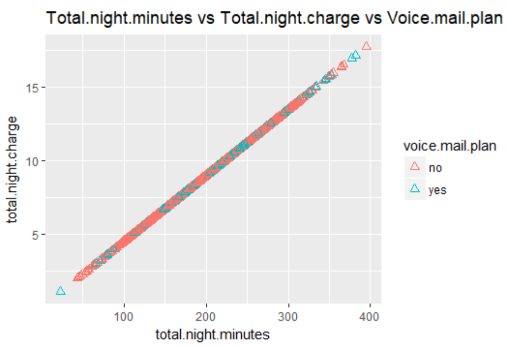
## Correlation plots

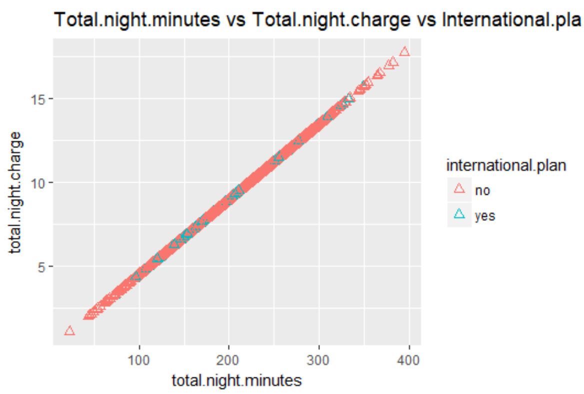


# Appendix B - Code

## R-code

```
rm(list=ls())

getwd()

setwd("E:/project_1")

# loading packages

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",

      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees',"readr","class")

lapply(x, require,character.only=T)

rm(x)

data_tr=read_csv("Train_data.csv")

data_tst=read_csv("Test_data.csv")

# Basic stat and data preparation

str(data_tr)

str(data_tst)

class(data_tr)

class(data_tst)

data_tr=data.frame(data_tr)

data_tst=data.frame(data_tst)

data_tr[1:5,1:6]

# converting variables into their respective types in train data

data_tr$phone.number=as.factor(data_tr$phone.number)

data_tr$international.plan=as.factor(as.character(data_tr$international.plan))

data_tr$voice.mail.plan=as.factor(as.character(data_tr$voice.mail.plan))

data_tr$Churn=as.factor(as.character(data_tr$Churn))

data_tr$area.code=as.factor(data_tr$area.code)

data_tr$state=as.factor(data_tr$state)

# converting variables into their respective types in test data

data_tst$phone.number=as.factor(data_tst$phone.number)

data_tst$international.plan=as.factor(as.character(data_tst$international.plan))
```

```
data_tst$voice.mail.plan=as.factor(as.character(data_tst$voice.mail.plan))

data_tst$Churn=as.factor(as.character(data_tst$Churn))

data_tst$area.code=as.factor(data_tst$area.code)

data_tst$state=as.factor(data_tst$state)

# plottings

data_tr = dt_tr

attach(data_tr)

ggplot(data_tr, aes(x=Churn,fill=Churn)) +

  geom_bar(position="dodge")+theme_bw()+ggtitle("Churn distribution")

ggplot(data_tr, aes(x=Churn,fill=international.plan)) +

  geom_bar(position="dodge")+theme_bw()+ggtitle("Churn vs international.plan")

ggplot(data_tr, aes(x=Churn,fill=voice.mail.plan)) +

  geom_bar(position="dodge")+theme_bw()+ggtitle("Churn vs voice.mail.plan")

ggplot(data_tr, aes(x=state,fill=Churn)) +

  geom_bar(position="dodge")+theme_bw()+ggtitle("Churn vs state")

ggplot(data_tr, aes(account.length,fill=Churn)) +

  geom_histogram(binwidth = 1,position="identity")+ggtitle("account.length vs churn distribution")

ggplot(data_tr, aes(number.vmail.messages,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("number.vmai.messages vs Churn distribution")

ggplot(data_tr, aes(number.customer.service.calls,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("number.customer.service.calls vs Churn distribution")

ggplot(data_tr, aes(total.day.charge,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("total.day.charge vs Churn")

ggplot(data_tr, aes(total.eve.charge,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("total.eve.charge vs Churn")

ggplot(data_tr, aes(total.intl.charge,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("total.intl.charge vs Churn")

ggplot(data_tr, aes(total.night.charge,fill=Churn)) +

  geom_histogram(binwidth = 5,position="dodge")+ggtitle("total.night.charge vs Churn")

#### scatter plot

ggplot(data_tr, aes(x = total.intl.minutes, y = total.intl.charge, col= Churn)) +

  geom_point(shape = 2, size = 2)+ggtitle("Total.intl.minutes vs Total.intl.charge vs Churn")
```

```
ggplot(data_tr, aes(x = total.intl.minutes, y = total.intl.charge, col= voice.mail.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.intl.minutes vs Total.intl.charge vs Voice.mail.plan ")
ggplot(data_tr, aes(x = total.intl.minutes, y = total.intl.charge, col= international.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.intl.minutes vs Total.intl.charge vs International.plan ")
ggplot(data_tr, aes(x = total.night.minutes, y = total.night.charge, col= international.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.night.minutes vs Total.night.charge vs International.plan ")
ggplot(data_tr, aes(x = total.night.minutes, y = total.night.charge, col= voice.mail.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.night.minutes vs Total.night.charge vs Voice.mail.plan ")
ggplot(data_tr, aes(x = total.night.minutes, y = total.night.charge, col= Churn)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.night.minutes vs Total.night.charge vs Churn ")
ggplot(data_tr, aes(x = total.eve.minutes, y = total.eve.charge, col= international.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.eve.minutes vs Total.eve.charge vs International.plan ")
ggplot(data_tr, aes(x = total.eve.minutes, y = total.eve.charge, col= voice.mail.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.eve.minutes vs Total.eve.charge vs Voice.mail.plan ")
ggplot(data_tr, aes(x = total.eve.minutes, y = total.eve.charge, col= Churn)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.eve.minutes vs Total.eve.charge vs Churn ")
##########
ggplot(data_tr, aes(x = total.day.minutes, y= total.day.charge, col= international.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.day.minutes vs Total.day.charge vs International.plan ")
ggplot(data_tr, aes(x = total.day.minutes, y = total.day.charge, col= voice.mail.plan)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.day.minutes vs Total.day.charge vs Voice.mail.plan ")
ggplot(data_tr, aes(x = total.day.minutes, y = total.day.charge, col= Churn)) +
  geom_point(shape = 2, size = 2)+ggtitle("Total.day.minutes vs Total.day.charge vs Churn ")
############## boxplot
ggplot(data_tr,aes(x=Churn,y=total.day.calls,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.day.calls)")
ggplot(data_tr,aes(x=Churn,y=total.day.charge,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.day.charge)")
ggplot(data_tr,aes(x=Churn,y=total.day.minutes,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.day.minutes)")
ggplot(data_tr,aes(x=Churn,y=total.intl.calls,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.intl.calls)")
```

```
ggplot(data_tr,aes(x=Churn,y=total.intl.charge,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.intl.charge)")
ggplot(data_tr,aes(x=Churn,y=total.intl.minutes,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.intl.minutes)")
ggplot(data_tr,aes(x=Churn,y=total.eve.calls,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.eve.calls)")
ggplot(data_tr,aes(x=Churn,y=total.eve.charge,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.eve.charge)")
ggplot(data_tr,aes(x=Churn,y=total.eve.minutes,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.eve.minutes)")
ggplot(data_tr,aes(x=Churn,y=total.night.charge,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.night.charge)")
ggplot(data_tr,aes(x=Churn,y=total.night.calls,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.night.calls)")
ggplot(data_tr,aes(x=Churn,y=total.night.minutes,fill=Churn))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs total.night.minutes)")
ggplot(data_tr,aes(x=Churn,y=account.length,fill=area))+
  geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("outlier analysis(churn vs account.length)")
# missing value analysis
sum(is.na(data_tr))
sum(is.na(data_tst))
data_tr_missing = data.frame(apply(data_tr,2,function(x){sum(is.na(x))}))
data_tst_missing = data.frame(apply(data_tst,2,function(x){sum(is.na(x))}))
data_tr_missing$colums=row.names(data_tr_missing)
row.names(data_tr_missing)=NULL
names(data_tr_missing)[1]="PERCENT"
data_tr_missing=data_tr_missing[,c(2,1)]
data_tst_missing$colums=row.names(data_tst_missing)
row.names(data_tst_missing)=NULL
names(data_tst_missing)[1]="PERCENT"
data_tst_missing=data_tst_missing[,c(2,1)]
# we dont have any missing values in the two data sets
```

```
#assigning levels to categorical variables
for(i in 1:ncol(data_tr)){
 if(class(data_tr[,i])== 'factor'){
    data_tr[,i] = factor(data_tr[,i], labels = (1:length(levels(factor(data_tr[,i])))))
  }
}
for(i in 1:ncol(data_tst)){
  if(class(data_tst[,i])== 'factor'){
    data_tst[,i] = factor(data_tst[,i], labels = (1:length(levels(factor(data_tst[,i])))))
  }
}
# outlier analysis
num_tr_ind=sapply(data_tr,is.numeric)
num_tr_data=data_tr[,num_tr_ind]
cnames=colnames(num_tr_data)
for(i in cnames){
  print(i)
  v=data_tr[,i][data_tr[,i]%in%boxplot.stats(data_tr[,i])$out]
  print(length(v))
  print(v)
}
library("ggplot2")
for (i in 1:length(cnames)) {
  assign(paste0("gn",i), ggplot(aes_string( y = (cnames[i]), x= "Churn") , data = subset(data_tr)) +
           stat_boxplot(geom = "errorbar" , width = 0.5) +
           geom_boxplot(outlier.color = "red", fill = "grey", outlier.shape = 20, outlier.size = 1, notch = FALSE)+
           theme(legend.position = "bottom")+
           labs(y = cnames[i], x= "Churn")+
           ggtitle(paste("Boxplot" , cnames[i])))
  #print(i)
}
#Now plotting the plots
```

```
gridExtra::grid.arrange(gn1, gn2,gn3, ncol=3)

gridExtra::grid.arrange(gn4,gn5,gn6, ncol=3)

gridExtra::grid.arrange(gn7,gn8,gn9, ncol =3)

gridExtra::grid.arrange(gn10,gn11, ncol =3 )

## WE ARE NOT APPLYING OUTLIER ANALYSIS ON TEST DATA BECAUSE WE HAVE LESS NUMBER OF OUTLIERS

# saving num variables

dt_tr=data_tr

dt_tst=data_tst

dt_tr_ind=sapply(dt_tr,is.numeric)

## Correlation Plot

corrgram(dt_tr[,dt_tr_ind], order = F,

        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

 ## Chi-squared Test of Independence

dt_tr_fac = sapply(dt_tr,is.factor)

fa_dt = dt_tr[,dt_tr_fac]

for (i in 1:5){

  print(names(fa_dt)[i])

  print(chisq.test(table(fa_dt$Churn,fa_dt[,i])))

}

# droping variables which are not carring much information

dt_tr=subset(dt_tr,select=-c(area.code,total.day.minutes, total.eve.minutes, total.night.minutes, total.intl.minutes, phone.number))

dt_tst=subset(dt_tst,select=-c(area.code,total.day.minutes, total.eve.minutes, total.night.minutes, total.intl.minutes, phone.number))

# feature scaling

# checking normality

hist(dt_tr$total.day.calls)

hist(dt_tr$number.customer.service.calls)

hist(dt_tr$number.vmail.messages)

#Normalisation

cnames_num = c("account.length","number.vmail.messages","total.day.calls","total.day.charge","total.eve.calls","total.eve.charge","total.night.calls","total.night.charge","total.intl.calls", "total.intl.charge",  "number.customer.service.calls")
```

```
for(i in cnames_num){
  dt_tr[,i]=(dt_tr[,i]-min(dt_tr[,i]))/(max(dt_tr[,i]-min(dt_tr[,i])))
}
# for test data
for(i in cnames_num){
  dt_tst[,i]=(dt_tst[,i]-min(dt_tst[,i]))/(max(dt_tst[,i]-min(dt_tst[,i])))
}
# building model
rmExcept(c("dt_tr","dt_tst"))
train=dt_tr
test=dt_tst
#### scince the data is unbalanced we are applying balencing methods
# combining data
data=rbind(train,test)
d_ind=createDataPartition(data$Churn,p=0.666,list=F)
train=data[d_ind,]
test=data[-d_ind,]
# Decision tree on unbalanced data
c50_mod=C5.0(Churn~.,train,trails=50,rules=T)
summary(c50_mod)
write(capture.output(summary(c50_mod)), "c50Rules.txt")
#predicting the test cases
prd_c50=predict(c50_mod,test[,-15],type = "class")
# evaluating the performance
cf_c50=table(test$Churn,prd_c50)
confusionMatrix(cf_c50)
cf_c50
#accuracy= 89.2%
#FNR=FN/TP+FN=34.3 %
rc_dt=roc.curve(test$Churn,prd_c50)
rc_dt
# the value of auc = 79.3% it is quite low
```

```
# applying ROSE method

rose_data=ROSE(Churn~.,data=train,seed=111)$data

c50_mod=C5.0(Churn~.,rose_data,trails=50,rules=T)

summary(c50_mod)

write(capture.output(summary(c50_mod)), "c50Rules.txt")

#predicting the test cases

prd_c50=predict(c50_mod,test[,-15],type = "class")

# evaluating the performance

cf_c50=table(test$Churn,prd_c50)

confusionMatrix(cf_c50)

cf_c50

rc_rose=roc.curve(test$Churn,prd_c50)

rc_rose

########### SMOTE method

smote_data=SMOTE(Churn~.,data=train,perc.over = 200,perc.under = 200)

table(smote_data$Churn)

c50_mod=C5.0(Churn~.,smote_data,trails=50,rules=T)

summary(c50_mod)

write(capture.output(summary(c50_mod)), "c50Rules.txt")

#predicting the test cases

prd_c50=predict(c50_mod,test[,-15],type = "class")

# evaluating the performance

cf_c50=table(test$Churn,prd_c50)

confusionMatrix(cf_c50)

cf_c50

rc_smote=roc.curve(test$Churn,prd_c50)

rc_smote

# accuracy=91.7%

# FNR=25.7%%

# Area under the curve (AUC): 0.869

# we can observe that smote method is working fine

# Random forest
```

```
rf_mod=randomForest(Churn~.,smote_data,ntree=500,importance=T)

#extracting rules from random forest

tree_list=RF2List(rf_mod)

#extract rules

exct=extractRules(tree_list,smote_data[,-15])

#visualize some rules

exct[1:2,]

#making the rules readable

readable_rules=presentRules(exct,colnames(smote_data))

readable_rules[1:2,]

# rule  metric

rule_metric=getRuleMetric(exct,smote_data[,-15],smote_data$Churn)

rule_metric[1:2,]

#predicting the model

rf_prd=predict(rf_mod,test[,-15])

# confusion matrix

cf_rf=table(test$Churn,rf_prd)

confusionMatrix(cf_rf)

rc_rf=roc.curve(test$Churn,rf_prd)

rc_rf

# accuracy=88.02%

# FNR=18.5%%

# logistic regression

logit_mod=glm(Churn~.,smote_data,family = "binomial")

summary(logit_mod)

#predicting the model

logit_prd=predict(logit_mod,test,type="response")

logit_prd=ifelse(logit_prd>0.5,1,0)

#evaluating the performance

logit_cm=table(test$Churn,logit_prd)

#ACCURACY

sum(diag(logit_cm)/nrow(test))
```

#FNR

#FN/FN+TP

#accuracy=78.9%

#FNR=44.6%

# KNN model implemenation

require(class)

#predict test data

knn_prd=knn(smote_data[,1:14],test[,1:14],smote_data$Churn,k=7)

#confusion matrix

cm_knn=table(knn_prd,test$Churn)

cm_knn

confusionMatrix(cm_knn)

#accuracy

sum(diag(cm_knn)/nrow(test))

#FNR

#FNR=FN/FN+TP

# acc =81.18%

# FNR=66.02%

# naive bayes implementation

require(e1071)

# devloping a model

nb_mod=naiveBayes(Churn~.,smote_data)

# prediction on test cases

nb_prd=predict(nb_mod,test[,1:14],type="class")

#confusion matrix

cm_nb=table(observed=test[,15],predicted =nb_prd)

confusionMatrix(cm_nb)

# accuracy=82.09%

# FNR=46.2%

**Python – code**

```
In [ ]:

# loading libraries into environment
```

```
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
from subprocess import check_output
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
```

In [ ]:

```
# checking  working dir

os.getcwd()
```

In [ ]:

```
# setting working dir

os.chdir('C:/Users/Adhya/Desktop/python')
```

In [ ]:

```
# load train and test data

data_tr=pd.read_csv("train_data.csv")

data_tst=pd.read_csv("test_data.csv")
```

In [ ]:

```
## Basic stat and data preparation

data_tr.shape
```

In [ ]:

```
data_tst.shape
```

In [ ]:

```
data_tr.dtypes
```

In [ ]:

```
data_tst.dtypes
```

In [ ]:

```
#storing target variable
train_dep = data_tr.Churn
```

In [ ]:

```
test_dep = data_tst.Churn
```

In [ ]:

```
print(data_tst.info())
```

In [ ]:

```
print(data_tr.info())
```

In [ ]:

```
data_tr.shape
```

In [ ]:

```
data_tr.tail(5)
```

In [ ]:

```
data_tr.describe(include=['O'])
```

In [ ]:

```
data_tr.drop(["phone number"], axis = 1, inplace=True)
```

In [ ]:

```
data_tst.drop(["phone number"], axis = 1, inplace=True)
```

In [ ]:

```
print(data_tr.info())
```

In [ ]:

```
# changing variables tinto their respective types
```

```
data_tr["area code"]=data_tr["area code"].astype(object)
```

In [ ]:

```
data_tst["area code"]=data_tst["area code"].astype(object)
```

In [ ]:

```
# storing number of missing values into data frame

data_tr_miss=pd.DataFrame(data_tr.isnull().sum())
```

In [ ]:

```
data_tr_miss.head()
```

In [ ]:

```
data_tst_miss=pd.DataFrame(data_tst.isnull().sum())
```

In [ ]:

```
data_tst_miss.head()
```

In [ ]:

```
# resetting index

data_tr_miss=data_tr_miss.reset_index()
```

In [ ]:

```
data_tr_miss.head()
```

In [ ]:

```
data_tst_miss=data_tst_miss.reset_index()
```

In [ ]:

```
data_tst_miss.head()
```

In [ ]:

```
# changing column names of the datasets

data_tr_miss=data_tr_miss.rename(columns={'index':'columns',0:'percent'})
```

In [ ]:

```
# checking for outliers inthe data

# make copy of the data

dt_tr=data_tr.copy()
```

In [ ]:

```
dt_tst=data_tst.copy()
```

In [ ]:

```
# plot barplot to visualize the outliers

%matplotlib inline
plt.boxplot(data_tr["total eve minutes"])
```

In [ ]:

```
%matplotlib inline
plt.boxplot(data_tst["total eve minutes"])
```

In [ ]:

```
# save numeric names

cnames=["account length","number vmail messages","total day minutes","total day cal
ls","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"]
```

In [ ]:

```
for i in range(0,data_tr.shape[1]):
    if(data_tr.iloc[:,i].dtypes =='object'):
        data_tr.iloc[:,i]=pd.Categorical(data_tr.iloc[:,i])
        data_tr.iloc[:,i]=data_tr.iloc[:,i].cat.codes
```

In [ ]:

```
data_tr.head(5)
```

In [ ]:

```
for i in range(0,data_tst.shape[1]):
    if(data_tst.iloc[:,i].dtypes =='object'):
        data_tst.iloc[:,i]=pd.Categorical(data_tst.iloc[:,i])
        data_tst.iloc[:,i]=data_tst.iloc[:,i].cat.codes
```

In [ ]:

```
data_tst.head()
```

In [ ]:

```
data_tst.dtypes
```

In [ ]:

```
# replace -1 with NA

for i in range(0,data_tst.shape[1]):
    data_tst.iloc[:,i]=data_tst.iloc[:,i].replace(-1,np.nan)
```

In [ ]:

```
# replace -1 with NA

for i in range(0,data_tr.shape[1]):
    data_tr.iloc[:,i]=data_tr.iloc[:,i].replace(-1,np.nan)
```

In [ ]:

```
data_tr.head(5)
```

In [ ]:

```
data_tr.head(7)
```

In [ ]:

```
data_tr.dtypes
```

In [ ]:

```
data_tr['Churn']=data_tr['Churn'].astype(object)
```

In [ ]:

```
data_tr['state']=np.round(data_tr['state']).astype(object)
```

In [ ]:

```
data_tr['international plan']=data_tr['international plan'].astype(object)
```

In [ ]:

```
data_tr['voice mail plan']=data_tr['voice mail plan'].astype(object)
```

In [ ]:

```
data_tr['area code']=data_tr['area code'].astype(object)
```

In [ ]:

```
data_tr.dtypes
```

In [ ]:

```
data_tr.head(5)
```

**feature selection¶**

In [ ]:

```
## correlation analysis

#correlation plot

dt_cor=data_tr.loc[:,cnames]
```

In [ ]:

```
# set width and hight of the plot

f,ax=plt.subplots(figsize=(10,10))

#generating corr plot

corr=dt_cor.corr()

# plot using seaborn lib

sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),annot=True,cmap=sns.divergi
ng_palette(220,10,as_cmap=True),square=True,ax=ax)
```

In [ ]:

```
data_tr.head(5)
```

In [ ]:

```
data_tr["international plan"].max()
```

In [ ]:

```
# chi sq test
```

```
# save categorical names

cat_names=["state","area code","international plan","voice mail plan"]
```

In [ ]:

```
for i in cat_names:
    print(i)
    chi2, p, dof, ex=chi2_contingency(pd.crosstab(data_tr["Churn"],data_tr[i]))
    print(p)
```

In [ ]:

```
data_tr=data_tr.drop(['area code','total day minutes', 'total eve minutes', 'total
night minutes', 'total intl minutes'],axis=1)
```

In [ ]:

```
data_tst=data_tst.drop(['area code','total day minutes', 'total eve minutes', 'tota
l night minutes', 'total intl minutes'],axis=1)
```

In [ ]:

```
data_tr.shape
```

In [ ]:

```
data_tst.shape
```

**feature selection¶**

In [ ]:

```
data_tr.head()
```

In [ ]:

```
data_tr.head()
```

In [ ]:

```
# normality check

%matplotlib inline
plt.hist(data_tr['total intl calls'],bins='auto')
```

In [ ]:

```
cnames = ["account length","number vmail messages","total day calls","total day cha
rge",
           "total eve calls","total eve charge","total night calls","total night ch
arge","total intl calls",
          "total intl charge", "number customer service calls"]
```

In [ ]:

```
cnames
```

In [ ]:

```
for i in cnames:
    print(i)
    data_tr[i]=(data_tr[i]-min(data_tr[i]))/(max(data_tr[i])-min(data_tr[i]))
```

In [ ]:

```
data_tr.head(5)
```

In [ ]:

```
for i in cnames:
    print(i)
    data_tst[i]=(data_tst[i]-min(data_tst[i]))/(max(data_tst[i])-min(data_tst[i]))
```

In [ ]:

```
data_tst.head()
```

In [ ]:

```
# make a copy of data

test=data_tst.copy()
```

In [ ]:

```
train=data_tr.copy()
```

In [ ]:

```
test.shape
```

In [ ]:

```
train.shape
```

In [ ]:

```
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
```

In [ ]:

```
train.dtypes
```

In [ ]:

```
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
```

In [ ]:

```
# replacing target with trueor false

train["Churn"]=train["Churn"].replace(0,'False')
```

In [ ]:

```
train["Churn"]=train["Churn"].replace(1,'True')
```

In [ ]:

```
test["Churn"]=test["Churn"].replace(0,'False')
```

In [ ]:

```
test["Churn"]=test["Churn"].replace(1,'True')
```

In [ ]:

```
from imblearn.over_sampling import SMOTE
```

In [ ]:

```
sm = SMOTE(random_state=123, ratio = .2)
```

In [ ]:

```
x_tr,y_tr=sm.fit_sample(train.iloc[:,0:14],train.iloc[:,14])
```

In [ ]:

```
x_tr.shape
```

In [ ]:

```
y_tr.shape
```

In [ ]:

```
clf = tree.DecisionTreeClassifier(criterion ='entropy').fit(x_tr,y_tr)
```

In [ ]:

```
clf
```

In [ ]:

```
prd=clf.predict(test.iloc[:,0:14])
```

In [ ]:

```
CM = pd.crosstab(test.iloc[:,14],prd)
```

In [ ]:

```
CM
```

In [ ]:

```
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

In [ ]:

```
accuracy_score(test.iloc[:,14], prd)*100
```

In [ ]:

```
(FN*100)/(FN+TP)
```

In [ ]:

```
cn=['state', 'account length', 'international plan', 'voice mail plan',
       'number vmail messages', 'total day calls', 'total day charge',
       'total eve calls', 'total eve charge', 'total night calls',
       'total night charge', 'total intl calls', 'total intl charge',
       'number customer service calls']
```

In [ ]:

```
# create a dot file to visualize tree #http://webgraphviz.com/

dotfile=open("pt.dot",'w')
df = tree.export_graphviz(clf, out_file=dotfile, feature_names =cn)
```

In [ ]:

```
#testing accuracy of model
from sklearn.metrics import confusion_matrix
```

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier

rf_mod = RandomForestClassifier(n_estimators = 100).fit(x_tr,y_tr)
```

In [ ]:

```
rf_prd = rf_mod.predict(test.iloc[:,0:14])
```

In [ ]:

```
rf_prd
```

In [ ]:

```
cm=pd.crosstab(test.iloc[:,14],rf_prd)
```

In [ ]:

```
cm
```

In [ ]:

```
#let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
FP = cm.iloc[0,1]

#accuracy test
```

In [ ]:

```
#accuracy test
accuracy_score(test.iloc[:,14],rf_prd)
```

In [ ]:

```
(TP+TN)*100/(TN+TP+FN+FP)
```

In [ ]:

```
#FNR
(FN*100)/(FN+TP)
```

In [ ]:

```
from sklearn.metrics import roc_auc_score
```

In [ ]:

```
# results
# if number of trees =100

#accuracy=89.2%
#FNR=28.8%

# if number of trees =500

#accuracy=90.8%
#FNR=27.8%
```

In [ ]:

```
#KNN implementation
from sklearn.neighbors import KNeighborsClassifier


knn_mod = KNeighborsClassifier(n_neighbors =9).fit(x_tr,y_tr)
```

In [ ]:

```
knn_prd=knn_mod.predict(test.iloc[:,0:14])
```

In [ ]:

```
cm=pd.crosstab(test.iloc[:,14],knn_prd)
```

In [ ]:

```
cm
```

In [ ]:

```
# accuracy
#let us save TP, TN, FP, FN
```

```
TN = cm.iloc[0,0]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
FP = cm.iloc[0,1]
```

In [ ]:

```
# accuracy test
accuracy_score(test.iloc[:,14],knn_prd)
```

In [ ]:

```
#FNR
(FN)/(FN+TP)*100
```

In [ ]:

```
# results of KNN

# if k(n nearest neighbors)=1
#accuracy=83%
#FNR=63.4%

# if k(n nearest neighbors)=3
#accuracy=85.8%
#FNR=71.6%

# if k(n nearest neighbors)=5
#accuracy=86.6%
#FNR=77.5%


# if k(n nearest neighbors)=7
#accuracy=86.8%
#FNR=83.3%


# if k(n nearest neighbors)=9
#accuracy=86.9%
#FNR=90%
```

In [ ]:

```
# naivebayes
```

In [ ]:

```
from sklearn.naive_bayes import GaussianNBIn
[ ]:
nb_mod=GaussianNB().fit(x_tr,y_tr)
```

```
In [ ]:

#predicting test cases
nb_prd=nb_mod.predict(test.iloc[:,0:14])
```

```
In [ ]:

# confusion matrix

cm=pd.crosstab(test.iloc[:,14],nb_prd)
```

```
In [ ]:

cm
```

```
In [ ]:

##let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
FP = cm.iloc[0,1]
```

```
In [ ]:

#accuracy check

accuracy_score(test.iloc[:,14],nb_prd)
```

```
In [ ]:

(TP+TN)/(TP+FN+TN+FP)*100
```

```
In [ ]:

#FNR
(FN)/(FN+TP)*100
```

```
In [ ]:

#results
#accuracy=85.8%
#FNR=44.2%
```

```
In [ ]:

rf_prd
```

```
In [ ]:
```

```
#now we will generate example out for out sample input test data with Random forest
predictions

move=pd.DataFrame(rf_prd)


In [ ]:

move=move.rename(columns={0:"move"})


In [ ]:

test=test.join(move['move'])


In [ ]:

test=test.drop('Churn',1)


In [ ]:

test.head()


In [ ]:

test['move']=test['move'].replace('True',1)


In [ ]:

test['move']=test['move'].replace('False',0)


In [ ]:

test.to_csv("example_output1.csv", index = False)
```