CHURN REDUCTION

MARCH 19, 2019

AKASH HEMANT SAXENA

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this case is to predict the whether the customer will churn or not. 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 Data

The task is to build a classification model which will predict whether the customer will churn or not with respect to other parameters.
Following table shows all the features of the data set:

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge
KS	128	415	382-4657	no	yes	25	265.1	110	45.07
OH	107	415	371-7191	no	yes	26	161.6	123	27.47
NJ	137	415	358-1921	no	no	0	243.4	114	41.38
OH	84	408	375-9999	yes	no	0	299.4	71	50.9
OK	75	415	330-6626	yes	no	0	166.7	113	28.34
AL	118	510	391-8027	yes	no	0	223.4	98	37.98

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	Churn
197.4	99	16.78	244.7	91	11.01	10	3	2.7	1	False.
195.5	103	16.62	254.4	103	11.45	13.7	3	3.7	1	False.
121.2	110	10.3	162.6	104	7.32	12.2	5	3.29	0	False.
61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False.
148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False.
220.6	101	18.75	203.9	118	9.18	6.3	6	1.7	0	False.

As per the above table following are the variable using which the classification model has to predict churn's.

1. STATE (Total 36 states present)
2. ACCOUNT LENGTH
3. AREA CODE
4. PHONE NUMBER
5. INTERNATIONAL PLAN (Whether customer has international plan activated or not)

6. VOICE MAIL PLAN (Whether customer has voice mail plan activated or not)
7. NUMBER VMAIL MESSAGES
8. TOTAL DAY MINUTES
9. TOTAL DAY CALLS
10. TOTAL DAY CHARGE
11. TOTAL EVE MINUTES
12. TOTAL EVE CALLS
13. TOTAL EVE CHARGE
14. TOTAL NIGHT MINUTES
15. TOTAL NIGHT CALLS
16. TOTAL NIGHT CHARGE
17. TOTAL INTL MINUTES
18. TOTAL INTL CALLS
19. TOTAL INTL CHARGE
20. NUMBER CUSTOMER SERVICE CALLS
21. CHURN (whether the customer churn or not)

Chapter 2
Methodology

2.1 Pre-processing

The output of the Machine Learning data is fully depended upon the data feed in the algorithm, ML algorithm does not generate the proper outcome on raw data. So it's important to transform the data so that the model accuracy of the model can be increased. In other words, we must apply some cleaning and processing for a better outcome with respect to the Machine learning algorithm. Some Machine learning algorithm required well-processed data like Decision Tree and random forest does not take null value. So it is important to process the data before feeding it in the respective machine learning algorithm.

Following are the pre-processing operation performed on the data to reduce the error rate and produce the optimal output.

2.1.1 Missing Value Analysis.

There are several options to handle missing value, but it mainly depends upon the nature of the data set, which missing analytics produce the optimum solution. Missing of data can occur due to nonresponse occur for example privacy concern. Moreover, if the value is missing completely at random, the data sample still represents the population but if the value is missing in some pattern, analysis produces bias output. There is two form of random missing values: MCAR and MAR. MCAR exists when the missing values are randomly distributed across all observations. This form can be confirmed by partitioning the data into two parts: one set containing the missing values, and the other containing the non-missing values. The second form is missing at random (MAR). In MAR, the missing values are not randomly distributed across observations but are distributed within one or more subsamples. Dropping the missing value is the good option if the data set has a low percentage of missing value and remaining data set can be used for further processing but it has its cons, dropping the missing value observation reduce the quantity of data. If our data set has a large number of missing value with respect to observation or feature, dropping is a good option. **Data set doesn't have missing value.**

Moreover, missing value analysis is done after outlier analysis due to the presence of outlier. Further description will be in Outlier analysis section.

Following code represent the missing value checking:

Python:

```
Train_data.isna().sum()
```

R code:

```
anyNA(new_dataset)
```

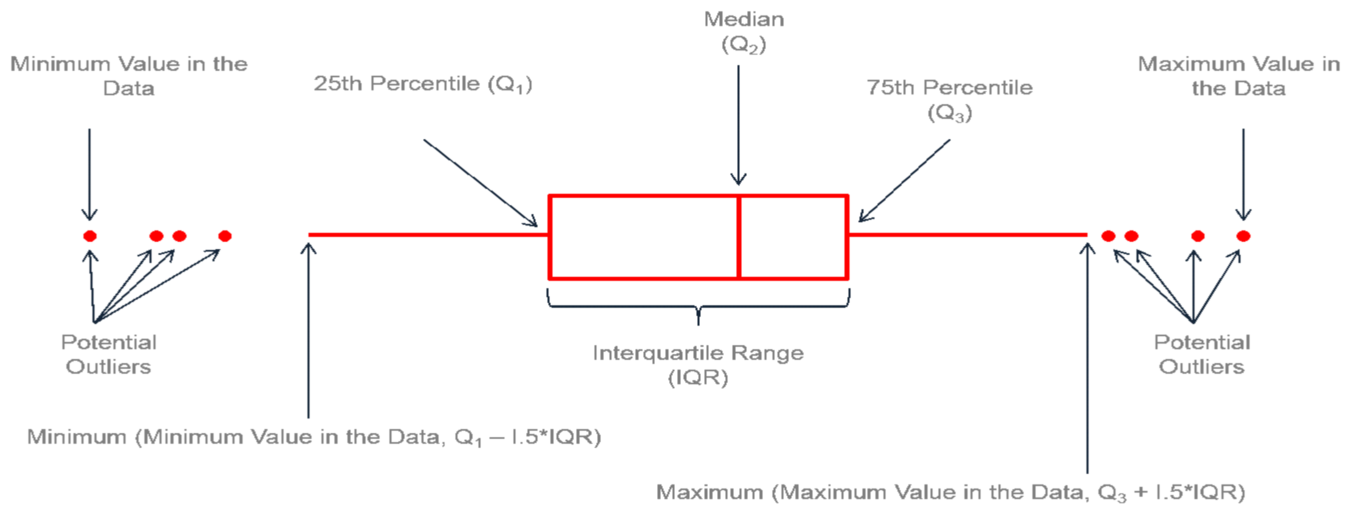
2.1.2 Outlier Analysis

Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

We have used box and Whisker method to detect the outliers and performed the operation over it.

Methodology of detecting the outliers:

As show in the figure the points/terms located outside the outer fences of the box and Whisker graph are considered as the outliers. Outliers are the extreme low and extreme high value in the respective dataset.



Median (Q_2)
Minimum Value in the Data
25th Percentile (Q_1)
75th Percentile (Q_3)
Maximum Value in the Data
Potential Outliers
Interquartile Range (IQR)
Potential Outliers
Minimum (Minimum Value in the Data, Q_1 – 1.5*IQR)
Maximum (Maximum Value in the Data, Q_3 + 1.5*IQR)

Calculation for outlier detection:

1. 25% and 75% of the value is obtain from the data set.
   (25% of the data set is obtain by split the data from its median and finding the median of split data set.)
   | Median (25%) | Median (50%) | Median (75%) |
2. Finding the interquartile range (IQR = 75% **(value)** - 25% **(value)**)
3. Finding the lower fence and upper fence:
   Lower fence = 25% **(value)** – IQR*1.5
   Upper fence = 75% **(value)** – IQR*1.5

The value which are smaller the lower fence and bigger then upper fence are consider as outliers.

Following code is used to detect the outliers in the data set:

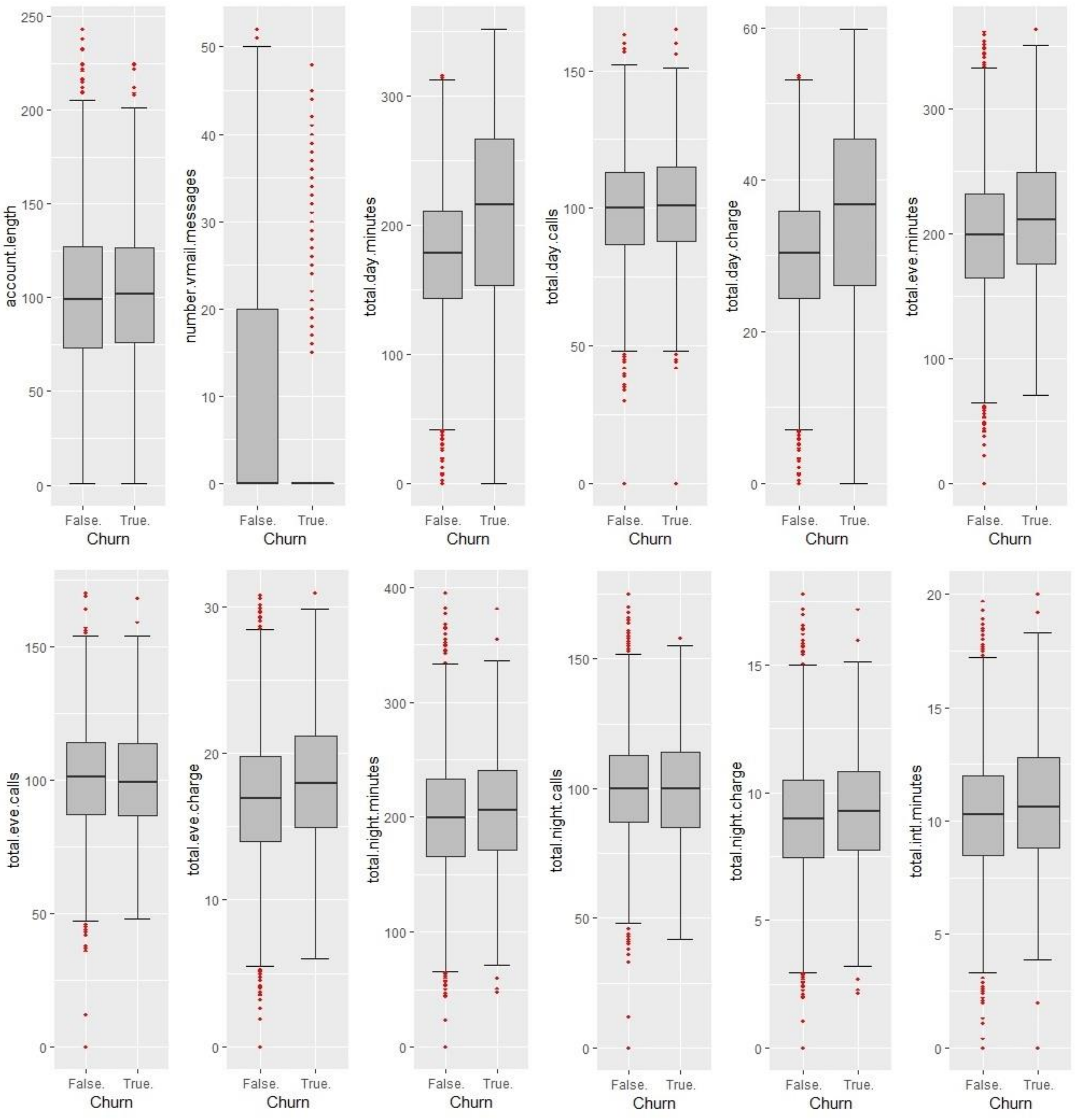
Python:

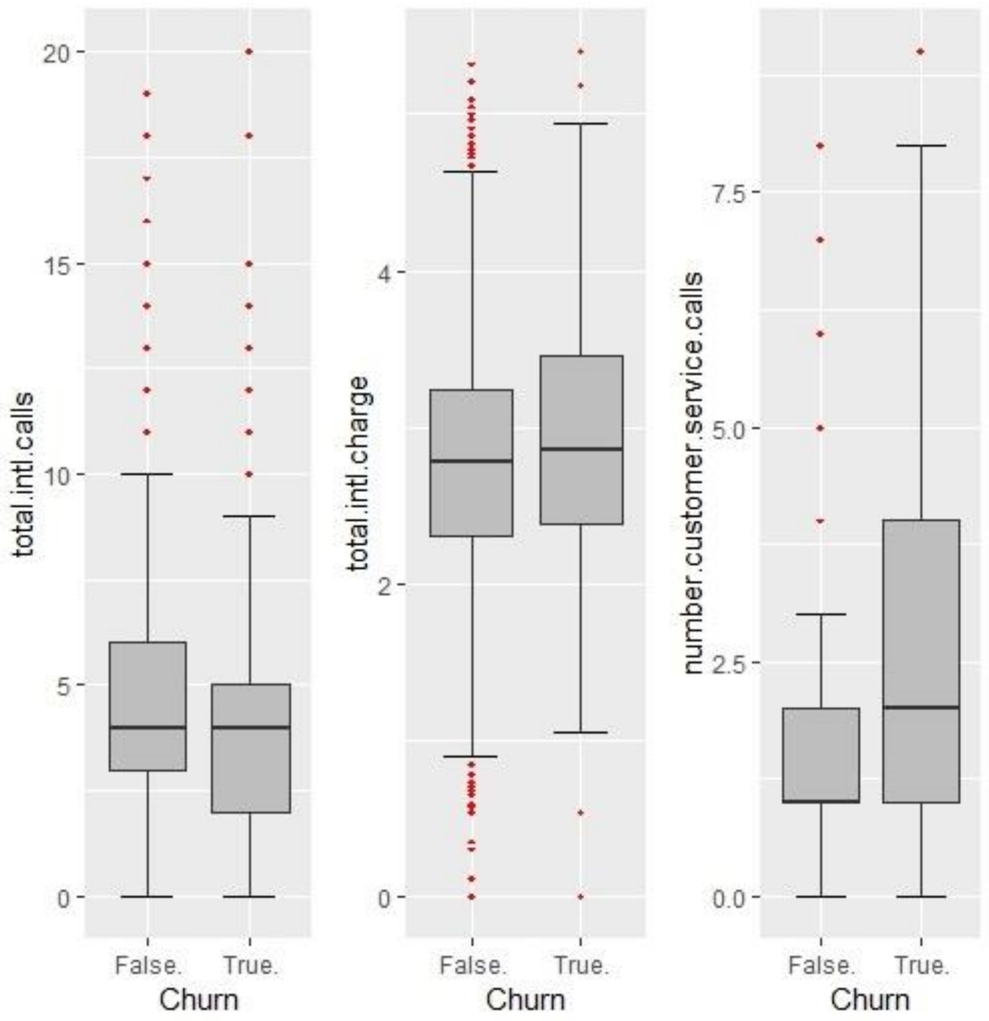
```
count=0
for i in numerical_col:
   q75,q25 = np.percentile(Train_data.loc[:,i],[75,25])
   iqr=q75-q25
   l_fence,u_fence = round(q25-(iqr*1.5),4),round(q75+(iqr*1.5),4)
   total_outlier = len([num for num in Train_data[i] if num > u_fence or num < l_fence ])
   if total_outlier != 0:
```

```
print('+++==========================================================
=============================================+++')
        print ('OUTLIER PRESENT IN ',i)
        plt.figure(figsize=(13,0.8))
        sns.boxplot(Train_data[i])
        plt.show()
        print('25% :{}    Median :{}    75% :{}    IQR :{}'. \
              format(q25,Train_data[i].median(),q75,iqr) )
        print('Minimum :{}    Lower Fence :{}    Upper Fence :{}    Maximum :{}'  \
              .format(Train_data[i].min(),l_fence,u_fence,Train_data[i].max()))
        print('Number of Outliers:{}    Percentage of Outlier in {} :{}%'  \
              .format(total_outlier,i,round(total_outlier/len(Train_data),2)*100))
        count = count + total_outlier

    else: print('+++++++----NO OUTLIER IN --->',i)
print('%'*40)
print('Total outliers is {}'.format(count))
print('Percentage of outliers is {}%'.format(round(count/Train_data.shape[0],2)*100))
```

R code:

```
numeric_index = sapply(new_dataset,is.numeric)
numeric_data = new_dataset[,numeric_index] # creating the dataset with numerical
varaible only
cnames = colnames(numeric_data) # fetching column names
for (i in 1:length(cnames)) # Ploting the boxplot
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"), data =
subset(new_dataset))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",cnames[i])))
}
rm(numeric_data,i)

#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,gn12, ncol = 3)
#gridExtra::grid.arrange(gn13,gn14,gn15, ncol = 3)
```


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
Churn
False.
True.



As there where outliers in our data set we have following option to deal with the outliers:

1. Delete the outliers and perform the further pre-processing
2. Replace the outliers with NA and impute the missing terms with imputation technique.

Procedure perform form finding the outliers:

1. Deleting the outliers:
   Following code is used to delete the outlier from the data set:
   Python:

```
        Process_data = Train_data.copy()
        for i in numerical_col:
           print(i)
           q75, q25 = np.percentile(Process_data.loc[:,i], [75 ,25])
           iqr = q75 - q25
           min, max = q25 - (iqr*1.5),q75 + (iqr*1.5)
           print('Maximim :{ma}\tMinimum :{mi}'.format( ma= max, mi= min))
           Process_data = Process_data.drop(Process_data[Process_data.loc[:,i] <
        min].index)
           Process_data = Process_data.drop(Process_data[Process_data.loc[:,i] >
        max].index)
    #print(i, '--> DONE')
```

R code:

```
Process1 =  new_dataset
for(i in cnames)
{
  val = Process1[,i][Process1[,i] %in% boxplot.stats(Process1[,i])$out]
  Process1 = Process1[which(!Process1[,i] %in% val),]
} # all the outliear got removed after applying this loop
```

2. Replacing the outliers with NA and imputation technique is used the replace the missing values.
   Following are the code the replace the outliers with NAN and KNN imputation is used to replace the missing terms.
   Python:

```
          Process_data2 = Train_data.copy()
          for i in numerical_col:
              print(i)
              q75_1, q25_1 = np.percentile(Process_data2.loc[:,i], [75 ,25])
              iqr2 = q75_1 - q25_1
              minimum, maximum = q25_1 - (iqr2*1.5),q75_1 + (iqr2*1.5)
              print('Maximim:', maximum)
              print('Minimum:', minimum)
              Process_data2[i] = np.where(Process_data2[i] < minimum , np.nan,
          Process_data2[i])
    Process_data2[i] = np.where(Process_data2[i] > maximum , np.nan,
Process_data2[i])



#Imputation
from fancyimpute import KNN
Process_data2 = pd.DataFrame(KNN(k = 3).complete(Process_data2),
columns=Process_data2.columns)
```

R code:

```
          Process2 = new_dataset
          for(i in cnames)
          {
            val = Process2[,i][Process2[,i] %in% boxplot.stats(Process2[,i])$out]
            Process2[,i][Process2[,i] %in% val] = NA
          }

          anyNA(Process2)
          #Imputation
```

```
        library(DMwR)
        Process2 = knnImputation(Process2,k=5)

anyNA(Process2)
```

KNN imputation method checks all the nearest variable with respect to the missing value. KNN algorithm used for the matching point with the closest k neighbour in the multi-dimension. It can be used for data that are categorical, discrete and continuous which makes it more efficient with all kind of missing data.
KNN imputation uses the distance formula to replace the missing value.
Following are the formula used by KNN imputation to replace the missing value.
Euclidean distance:

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \,.$$

Reason for not performing the statistical imputation technique:
Statistical imputation technique involves mean, median and mode method to replace the missing value with the respective dataset and mean, median and mode are considered to be central tendency of a data set. On other hand outliers are the extreme low and extreme high values of a data set. The difference will be fare greater if statistical imputation will be performed as compare to the imputation method.
Moreover, 12%(Python) of a data set has outliers. So rather than performing on falsely imputed data set it better to remove all the outliers and perform the operation on original data set.

2.1.3 Feature Selection

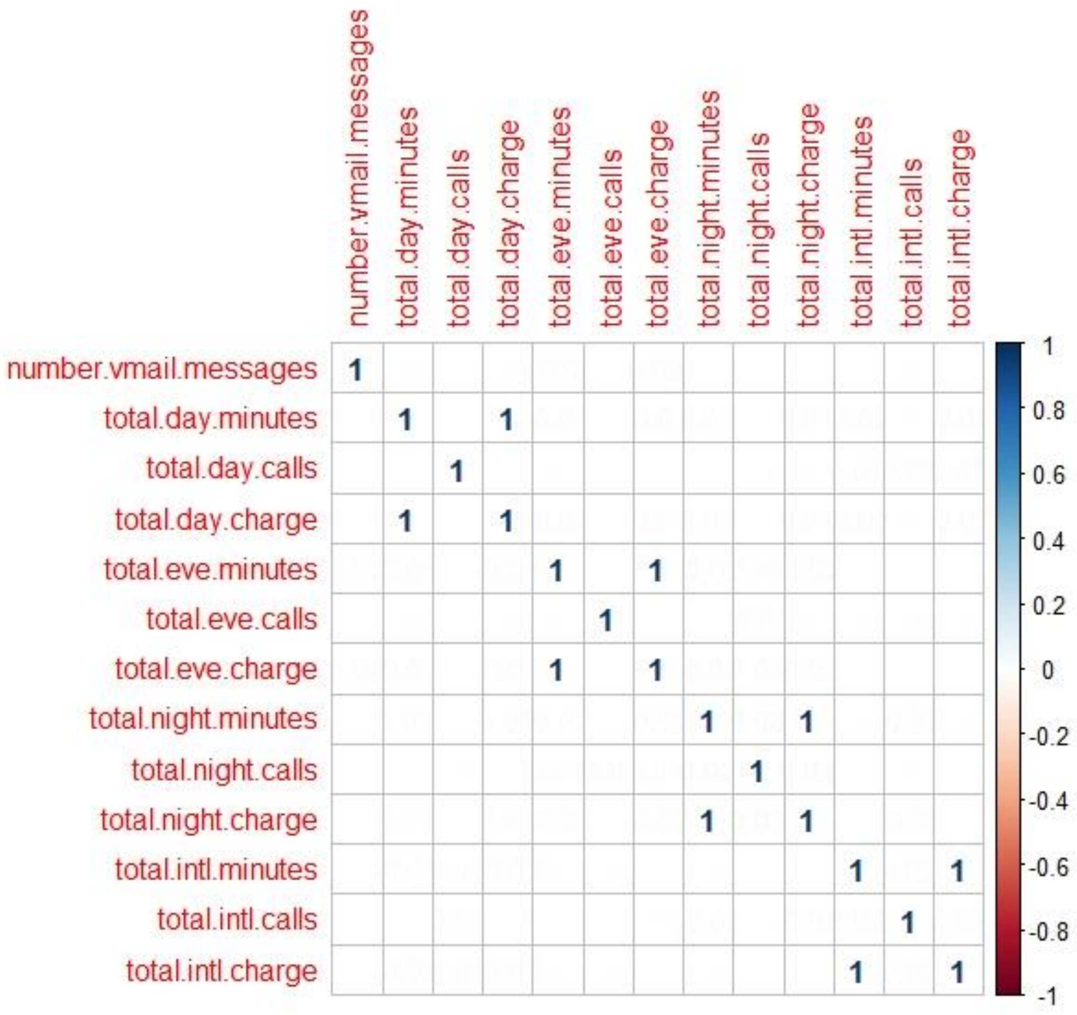
Feature selection is another pre-processing technique which decreases the load over machine learning algorithm checking the correlation between other feature and check which feature is highly correlated to another feature. If two feature carries the same data, this will create a bias output. Feature selection is a process where you can select those features which contribute most to the prediction output. Moreover, having a relevant feature in the data set can decrease the accuracy of the model.
As we have numerical feature and categorical feature, two types of feature selection method will be performed to select the relevant feature and will reduce the redundancy in the respective data set.
Following are method applied over the data set for the feature selection:

1. **Pearson correlation:**
   Pearson correlation apply on two numerical data set to check the correlation between the feature. IT measure the linear correlation between the two variable, it has the value between +1 and -1, where 1 is total positive correlation, 0 is no linear correlation and -1 is total negative linear correlation.
   Following is the respective formula to calculate the Pearson correlation:

$$r = \frac{N\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{[N\Sigma x^2 - (\Sigma x)^2][N\Sigma y^2 - (\Sigma y)^2]}}$$

Where:

N	=	number of pairs of scores
Σxy	=	sum of the products of paired scores
Σx	=	sum of x scores
Σy	=	sum of y scores
Σx^2	=	sum of squared x scores
Σy^2	=	sum of squared y scores



Python:

```
plt.figure(figsize=(20,20))
corr = Process_data.corr()
sns.heatmap(corr, annot= True,mask = np.zeros_like(corr, dtype = np.bool), vmin = -1,
vmax = 1 )
plt.show()
```

Removal of Feature

```
Process_data =
Process_data.drop(['total_day_minutes','total_eve_minutes','total_night_minutes','total_intl
_minutes'], axis=1)

Process_data2 =
Process_data2.drop(['total_day_minutes','total_eve_minutes','total_night_minutes','total_in
tl_minutes'], axis=1)
```

R code:

```
library(corrplot)

#Feature selection for numerical type data set

corrplot(corr = cor(new_dataset[,numerical_columns]),method = 'number')

#Observation:
#As expected, with repect to the above figure following feature are highly correlated with
each other :

# 1. total_day_charge and total_day_minutes (Correlation : 1)
# 2. total_eve_charge and total_eve_minutes (Correlation : 1)
# 3. total_night_charge and total_night_minutes (Correlation : 1)
# 4. total_intl_charge and total_intl_minutes (Correlation : 1)
#It's obvisous, as the amount charges by the operators are directly proportaional to total
minutes

# Deleting the feature
Process1 = subset(Process1, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,total.intl.minutes))

Process2 = subset(Process2, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,total.intl.minutes))
```

'total_day_minutes','total_eve_minutes','total_night_minutes','total_intl_minutes' where removed

2. **Chi-Square test of independence:**
   Compares two variables in a contingency table to see if they are related to each other. Hypothesis testing phenomena where null hypothesis represents that the two variables are independent and alternate hypothesis represents that two variables are not independent.
   Chi-Square test can be calculated as follows:

$$c^{2} = \sum_{i=1}^{k} \left[\frac{(O_i - E_i)^2}{E_i} \right]$$

We will check the p-value of chi-square test, as p-value is a probability representation of overall test and if p-value is less than 0.05 we will accept the null hypothesis (variables are independent) and if the value is greater than 0.05 then we can’t accept the null hypothesis as there will be some degree of dependency in the two variable.

Python:

```
for col in categorical_col:
    if col != 'Churn':
        chi2, p, dof, ex = chi2_contingency(pd.crosstab(Process_data['Churn'],
Process_data[col]))
        if p<0.05: print('{}:{}'.format(col,p))
        else: print('{}:{}\t**Feature should be remove'.format(col,p))
```

Removal of Feature

```
Process_data = Process_data.drop(['area_code'], axis=1)
Process_data2 = Process_data2.drop(['area_code'], axis=1)
```

R code:

```
# Feature Selection for categorical data
factor_index = sapply(Process1, is.factor)
factor_data = Process1[, factor_index]

for (i in 1:length(colnames(factor_data)))
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}



#1] "area.code"
#Pearson's Chi-squared test
#data:  table(factor_data$Churn, factor_data[, i])
#X-squared = 0.013014, df = 2, p-value = 0.9935
# As p-value is greater than 0.05, Feature should be remove

Process1 = subset(Process1, select = -area.code)
Process2 = subset(Process2, select = -area.code)
```

'area_code' was removed.

2.1.4 Feature Scaling

Feature scaling is a method to standardize/normalized the variable or the feature of the data set, it also knows as standardization/normalization of a data set. Some feature has

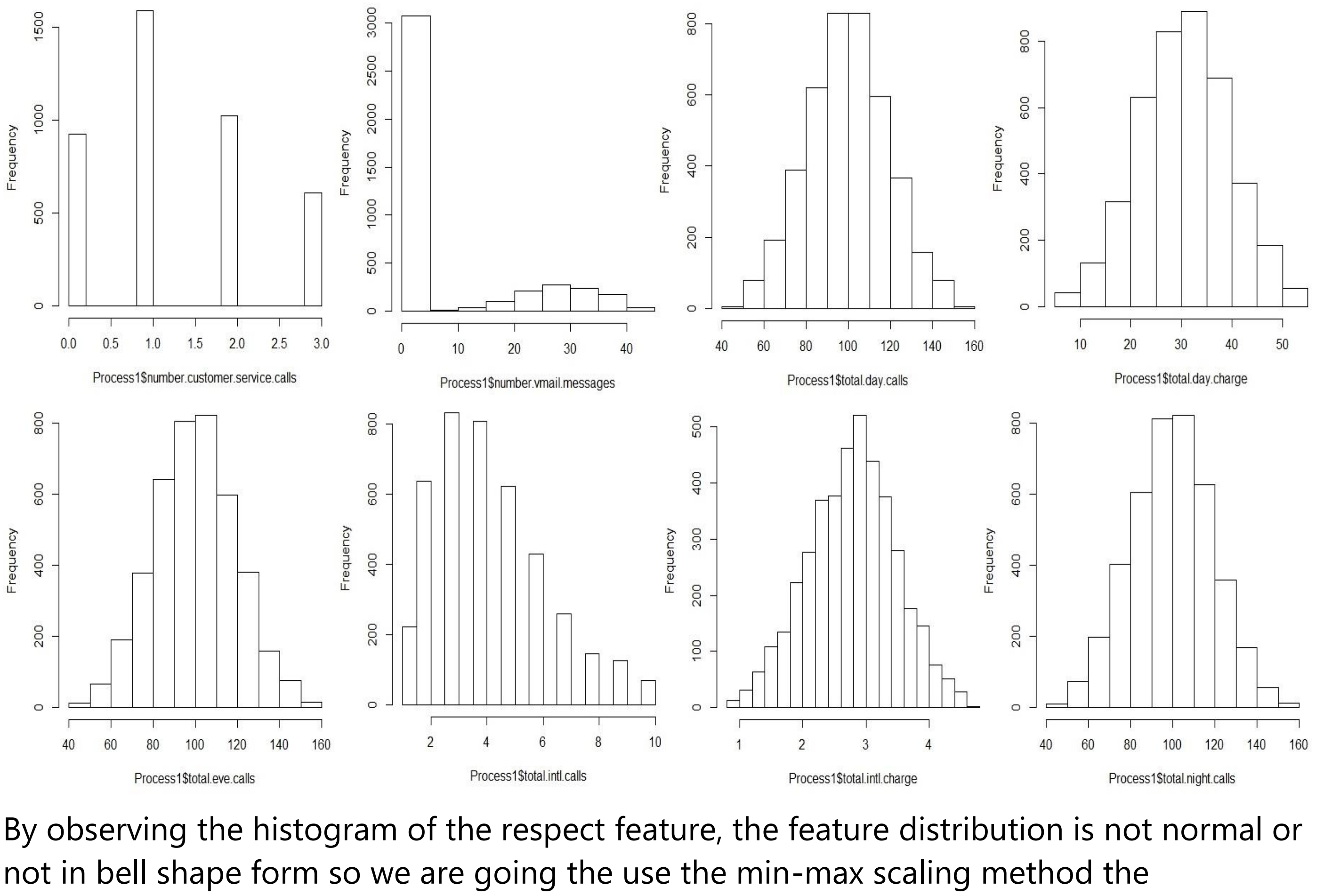
magnitudes, unit or range and normalization should be performed when the scaling of the feature is irrelevant or have a high magnitude which will lead to the bases in the machine learning algorithm which will lean towards the higher magnitude dataset. Z-scoring/standardization is one technique in which the value of a feature is subtracted with the mean of the feature and their subtraction is divided by the standard deviation of the feature which will also set the feature in normalized bell shape form whereas another method involves the value of a feature is subtracted by the minimum value of the respective feature which is divided by the subtraction of the minimum value and maximum value of the respective feature.
Following are the formula to perform normalization and z-score of a variable:

Min-Max scaling:

$Y_i = [X_i - \min(X)]/[\max(X) - \min(X)]$

Z-score scaling:(For normally distributed data)

$Z = (X_i – \text{Mean}(X))/ \text{Standard Deviation}(X)$



By observing the histogram of the respect feature, the feature distribution is not normal or not in bell shape form so we are going the use the min-max scaling method the normalized the respective features of our data set.
Following feature are normalized:
Python:

```
def normalize(data,columns):
    for i in columns:
        if i in list(data.columns):
```

```
        print(i)
        minimum , maximum = data[i].min(), data[i].max()
        data[i] = (data[i] - minimum)/(maximum - minimum)
    return data

normalized_list = []
for col in numerical_col: normalized_list.append(col)
normalized_list.append('account_length')

Process_data = normalize(Process_data, normalized_list)
Process_data2 = normalize(Process_data2, normalized_list)
```

R code:

```
#Checking the distribution of the numerical feature
#hist(Process1$number.vmail.messages) # HISTOGRAM_number.vmail.messages
#hist(Process1$total.day.calls) # HISTOGRAM_total.day.calls
#hist(Process1$total.day.charge) # HISTOGRAM_total.day.charge
#hist(Process1$total.eve.calls) # HISTOGRAM_total.eve.calls
#hist(Process1$total.night.calls) # HISTOGRAM_total.night.calls
#hist(Process1$total.night.charge) # HISTOGRAM_total.night.charge
#hist(Process1$total.intl.calls) # HISTOGRAM_total.intl.calls
#hist(Process1$total.intl.charge) # HISTOGRAM_total.intl.charge
#hist(Process1$number.customer.service.calls)                              #
HISTOGRAM_number.customer.service.calls

# INFERENCE:
# As THE DATA IS NOT NORMALIZED SO WE WILL USE Min-Max scaling METHOD

numerical_columns = 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")
#Normalisation
for (i in numerical_columns)
{
   print(i)
   Process1[,i] = (Process1[,i] - min(Process1[,i]))/(max(Process1[,i] - min(Process1[,i])))
}

for (i in numerical_columns)
{
  print(i)
  Process2[,i] = (Process2[,i] - min(Process2[,i]))/(max(Process2[,i] - min(Process2[,i])))
}
```

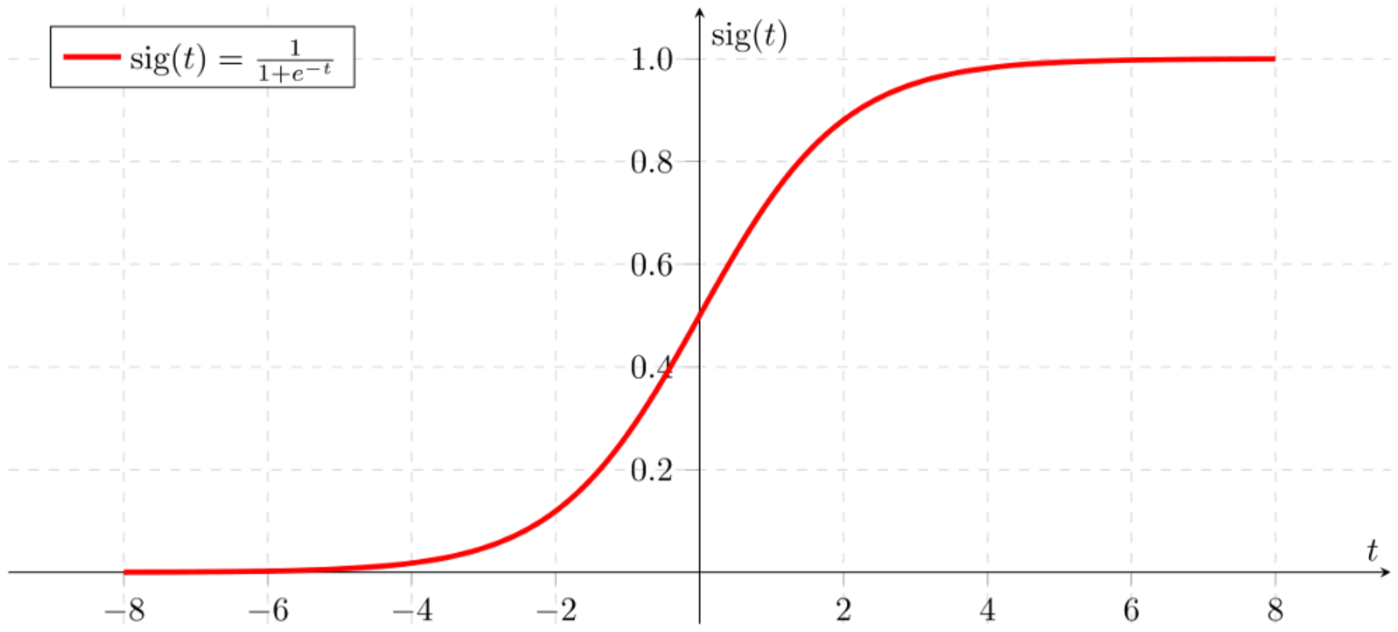
2.2 Modelling

2.1.1 Model Selection

In our earlier stage of analysis, we came to know that our dependent variable in our case is Churn is categorical, so we are using a classification analysis over our data set.
Following are the machine learning algorithm are performing on our data set to predict the future data.

2.1.2 Logistic Regression

Logistic regression used when the target variable is categorical as in our case. We are using binary logistic regression as our target variable has two unique parameters True or False.
Logistic regression uses the sigmoid function to determine the target value with respect to the probability. As in our case we have target variable which has two unique variables the probability factor will be dividend in to two sector, and the thresh hold value will be 0.5. If the probability value is greater than 0.5 then the final churn will True/1 and if the probability value is smaller than 0.5 then the churn will be false/0.
Following is the diagram represent the logistic diagram:



Following are the code for generating the Logistic Regression ML:
Python:

```
#Outlier Deleted
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression().fit(x_train,y_train)
logistic_estimated = logistic.predict(x_test)
logistic_result = model_acc(y_test,logistic_estimated)

#Outlier Imputed
```

```
from sklearn.linear_model import LogisticRegression
logistic2 = LogisticRegression().fit(x_train2,y_train2)
logistic_estimated2 = logistic2.predict(x_test2)
logistic_result2 = model_acc(y_test2,logistic_estimated2)
```

Output:

Outlier Deleted

```
=============================================
Accuracy :85.64%             Error Rate :14.36%
Specificity :96.77%   Recall :18.39%
False Positive :3.23% Fasle Negative :81.61%
=============================================
[[509  17]
 [ 71  16]]
```

Outlier Imputed

```
=============================================
Accuracy :85.91%             Error Rate :14.09%
Specificity :97.54%   Recall :17.53%
False Positive :2.46% Fasle Negative :82.47%
=============================================
[[556  14]
 [ 80  17]]
```

R code:

```
#Process1
logistic_model = glm(Churn~.,train, family = 'binomial')
summary(logistic_model)
logistic_predict1 = predict(logistic_model, test[,-15], type = 'response')
predicted_logit = as.factor(ifelse(logistic_predict1 > 0.5, 1,0))
caret::confusionMatrix(test$Churn, as.factor(predicted_logit))

#Accuracy : 0.9066

#Process2
logistic_model2 = glm(Churn~.,train2, family = 'binomial')
summary(logistic_model2)
logistic_predict2 = predict(logistic_model2, test2[,-15], type = 'response')
predicted_logit2 = as.factor(ifelse(logistic_predict2 > 0.5, 1,0))
caret::confusionMatrix(test2$Churn, as.factor(predicted_logit2))

#Accuracy : 0.8666

# With respect to the confusion matrix Process1 tent to perform far better as compare to process2
```

Output:

```
#Process1
> caret::confusionMatrix(test2$Churn, as.factor(predicted_logit2))
Confusion Matrix and Statistics
```

```
          Reference
Prediction    0    1
         0 1259   28
         1  173   39

               Accuracy : 0.8659
                 95% CI : (0.8476, 0.8828)
    No Information Rate : 0.9553
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2271
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8792
            Specificity : 0.5821
         Pos Pred Value : 0.9782
         Neg Pred Value : 0.1840
             Prevalence : 0.9553
         Detection Rate : 0.8399
   Detection Prevalence : 0.8586
      Balanced Accuracy : 0.7306

       'Positive' Class : 0


#Process2
> caret::confusionMatrix(test$Churn, c50_predict1)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1100   13
         1   41   88

               Accuracy : 0.9565
                 95% CI : (0.9436, 0.9672)
    No Information Rate : 0.9187
    P-Value [Acc > NIR] : 8.039e-08

                  Kappa : 0.7417
 Mcnemar's Test P-Value : 0.0002386

            Sensitivity : 0.9641
            Specificity : 0.8713
         Pos Pred Value : 0.9883
         Neg Pred Value : 0.6822
             Prevalence : 0.9187
         Detection Rate : 0.8857
   Detection Prevalence : 0.8961
      Balanced Accuracy : 0.9177

       'Positive' Class : 0
```

2.2.3. Decision Trees

Decision Tree algorithm creates a flow chat like tree structure where end nodes denote the outcome and the sub-nodes denote the decision flow of the tree.
Following are the code for generating the Decision Trees ML:
Python:

```
#Outlier Deleted
from sklearn import tree
DTree_model = tree.DecisionTreeClassifier().fit(x_train,y_train)
DTree_estimated = DTree_model.predict(x_test)
print('Decision Tree')
```

```
Decision_result = model_acc(y_test,DTree_estimated)

print('Feature Importance')
for col, per in zip(x_train,DTree_model.feature_importances_):print('{c} :{p}%'.format(c=col,
p = round(per*100,2)  ))

#Outlier Imputed
from sklearn import tree
DTree_model2 = tree.DecisionTreeClassifier().fit(x_train2,y_train2)
DTree_estimated2 = DTree_model2.predict(x_test2)
print('Decision Tree')
Decision_result2 = model_acc(y_test2,DTree_estimated2)
```

Output:

Outlier Deleted

```
Decision Tree
=============================================
Accuracy :90.7%              Error Rate :9.3%
Specificity :95.06%   Recall :64.37%
False Positive :4.94% Fasle Negative :35.63%
=============================================
[[500  26]
 [ 31  56]]
```

Outlier Imputed

```
Decision Tree
=============================================
Accuracy :91.45%             Error Rate :8.55%
Specificity :94.04%   Recall :76.29%
False Positive :5.96% Fasle Negative :23.71%
=============================================
[[536  34]
 [ 23  74]]
```

R code:

```
library(C50)
#Process1
c50model = C5.0(Churn~., train, trails = 100, rules = TRUE)
summary(c50model)
c50_predict1 = predict(c50model, test[,-15])

caret::confusionMatrix(test$Churn, c50_predict1)
#Accuracy : 0.9565
#Process2
c50model2 = C5.0(Churn~., train2, trails = 100, rules = TRUE)
summary(c50model2)
```

```
c50_predict2 = predict(c50model2, test2[,-15])

caret::confusionMatrix(test2$Churn, c50_predict2)

#Accuracy : 0.9213
# Similar phenomena observed in Decision tree modeling
# Process1 produce better respect to Process2
```

Output:

```
#Process1
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1100   13
         1   41   88

               Accuracy : 0.9565
                 95% CI : (0.9436, 0.9672)
    No Information Rate : 0.9187
    P-Value [Acc > NIR] : 8.039e-08

                  Kappa : 0.7417
 Mcnemar's Test P-Value : 0.0002386

            Sensitivity : 0.9641
            Specificity : 0.8713
         Pos Pred Value : 0.9883
         Neg Pred Value : 0.6822
             Prevalence : 0.9187
         Detection Rate : 0.8857
   Detection Prevalence : 0.8961
      Balanced Accuracy : 0.9177

       'Positive' Class : 0
#Process2
> caret::confusionMatrix(test2$Churn, c50_predict2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1261   26
         1   80  132

               Accuracy : 0.9293
                 95% CI : (0.9151, 0.9417)
    No Information Rate : 0.8946
    P-Value [Acc > NIR] : 2.515e-06

                  Kappa : 0.6742
 Mcnemar's Test P-Value : 2.635e-07

            Sensitivity : 0.9403
            Specificity : 0.8354
         Pos Pred Value : 0.9798
         Neg Pred Value : 0.6226
             Prevalence : 0.8946
         Detection Rate : 0.8412
   Detection Prevalence : 0.8586
      Balanced Accuracy : 0.8879

       'Positive' Class : 0
```

2.2.4. Random Forest

Similar to the decision tree, we also applied a random forest algorithm to check the prediction rate of a data set. Random forest works the same as decision tree but rather than creating a single tree, random forest creates multiple trees corresponding to the data set. Moreover, it fully depends on developer, the number of tree to be grown. More tree will be equal to the increase in accuracy of a model until the error rate stops decreasing. So we have generated multiple tree and checked the optimum number of tree required until error stops decreasing.
Following are the code for generating the Random Forest ML:
Python:

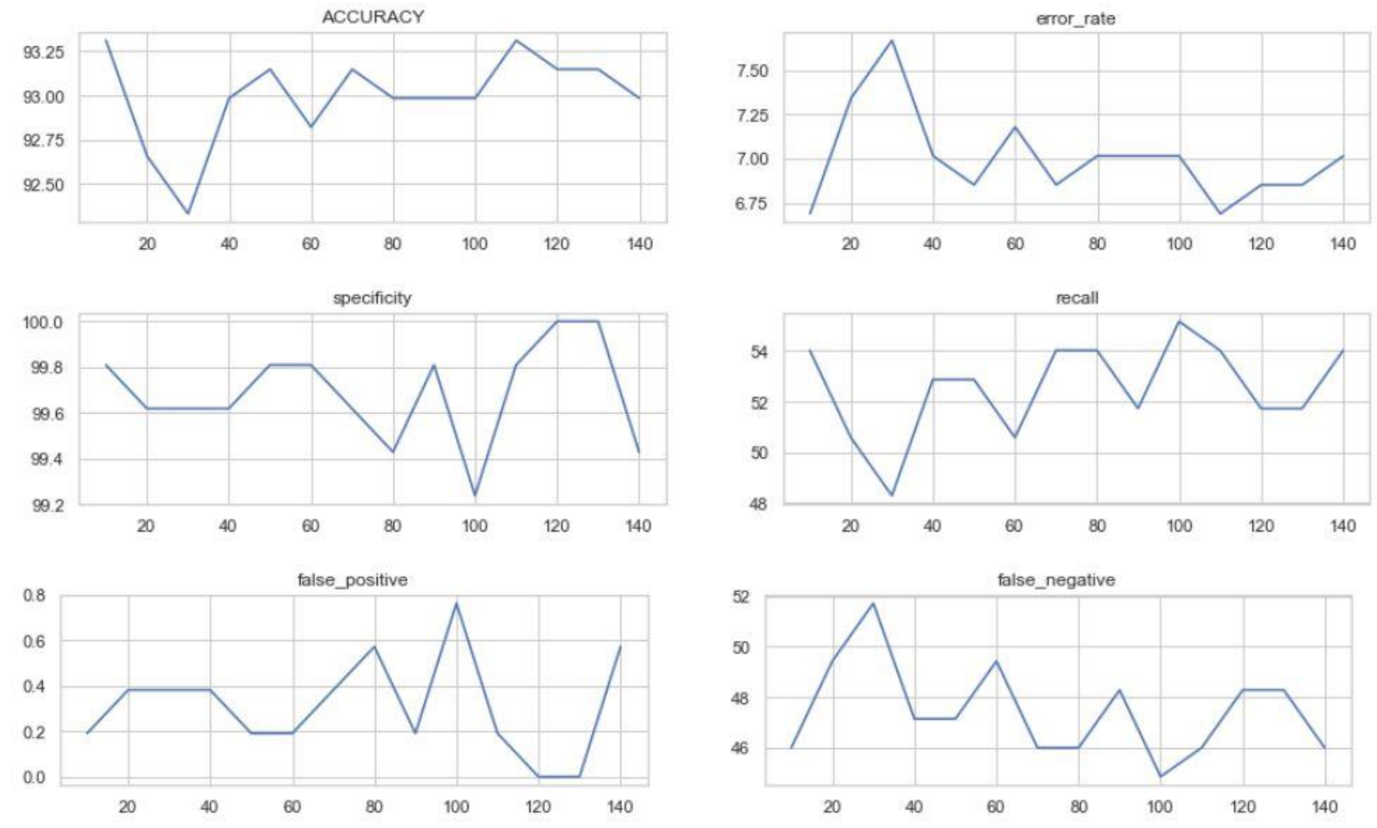
Outlier Deleted

```
from sklearn.ensemble import RandomForestClassifier
RF_model = RandomForestClassifier(n_estimators= 50).fit(x_train,y_train)
RF_estimated = RF_model.predict(x_test)
rf_result = model_acc(y_test,RF_estimated)
```

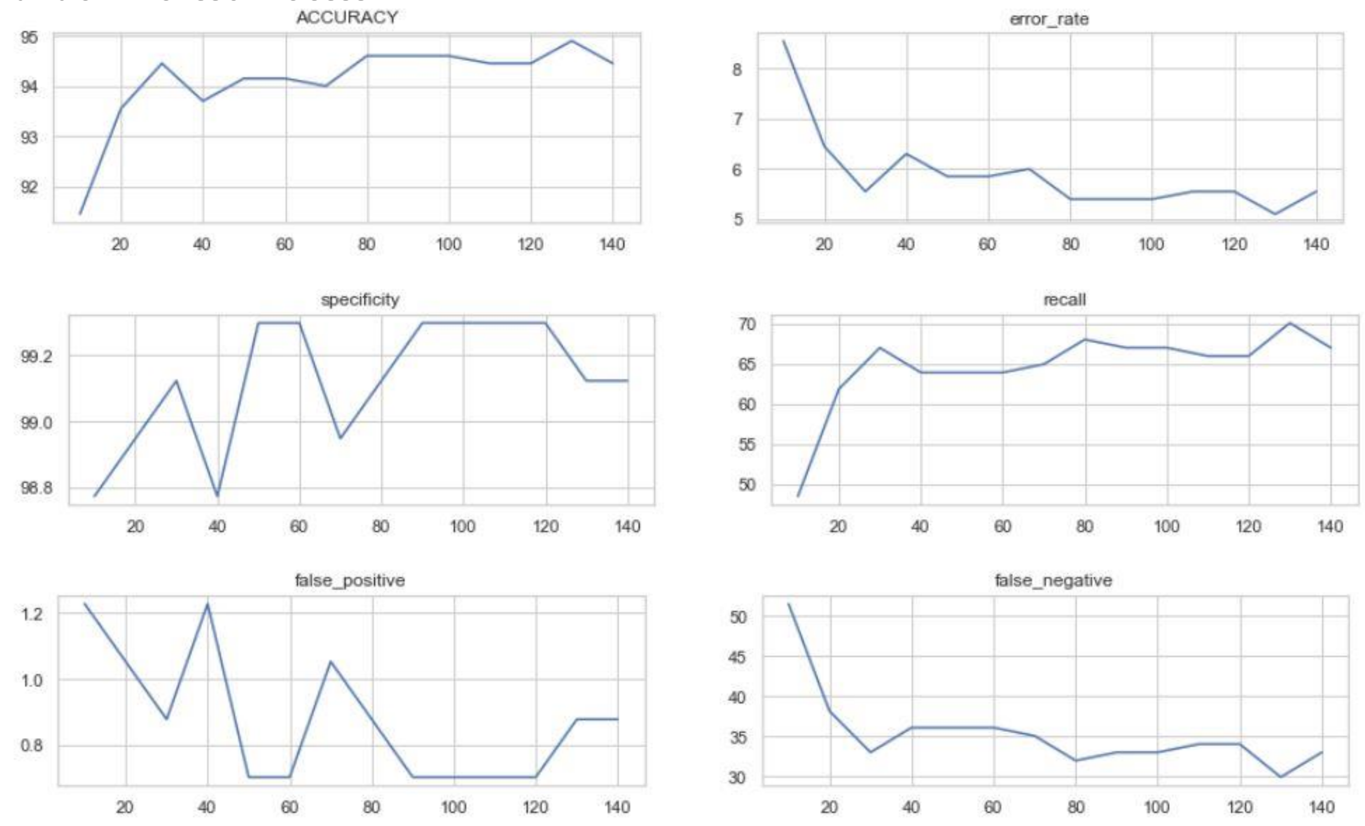
Outlier Imputed

```
from sklearn.ensemble import RandomForestClassifier
RF_model2 = RandomForestClassifier(n_estimators= 50).fit(x_train2,y_train2)
RF_estimated2 = RF_model2.predict(x_test2)
rf_result2 = model_acc(y_test2,RF_estimated2)
```

Random Forest Process1



Random Forest Process2



R code:

```
#Process1
RF_model = randomForest::randomForest(x= train[,-15], y = train$Churn, importance =
TRUE, ntree = 500)
summary(RF_model)
rf_predict1 = predict(RF_model, test[,-15])
caret::confusionMatrix(test$Churn, rf_predict1)

#Accuracy : 0.9597

#Process2
RF_model2 = randomForest::randomForest(x= train2[,-15], y = train2$Churn, importance
= TRUE, ntree = 500)
summary(RF_model2)
rf_predict2 = predict(RF_model2, test2[,-15])
caret::confusionMatrix(test2$Churn, rf_predict2)

#Accuracy : 0.9213

# Similar phenomena observed in random forest modeling
# Process1 produce better respect to Process2
```

Output:

```
#Process1
```

```
> caret::confusionMatrix(test$Churn, rf_predict1)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1110    3
         1   47   82

               Accuracy : 0.9597
                 95% CI : (0.9473, 0.97)
    No Information Rate : 0.9316
    P-Value [Acc > NIR] : 1.611e-05

                  Kappa : 0.7453
 Mcnemar's Test P-Value : 1.193e-09

            Sensitivity : 0.9594
            Specificity : 0.9647
         Pos Pred Value : 0.9973
         Neg Pred Value : 0.6357
             Prevalence : 0.9316
         Detection Rate : 0.8937
   Detection Prevalence : 0.8961
      Balanced Accuracy : 0.9620

       'Positive' Class : 0
#Process2
> caret::confusionMatrix(test2$Churn, rf_predict2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1277   10
         1  109  103

               Accuracy : 0.9206
                 95% CI : (0.9058, 0.9338)
    No Information Rate : 0.9246
    P-Value [Acc > NIR] : 0.7403

                  Kappa : 0.5939
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9214
            Specificity : 0.9115
         Pos Pred Value : 0.9922
         Neg Pred Value : 0.4858
             Prevalence : 0.9246
         Detection Rate : 0.8519
   Detection Prevalence : 0.8586
      Balanced Accuracy : 0.9164

       'Positive' Class : 0
```

2.2.5 K-Nearest Neighbours

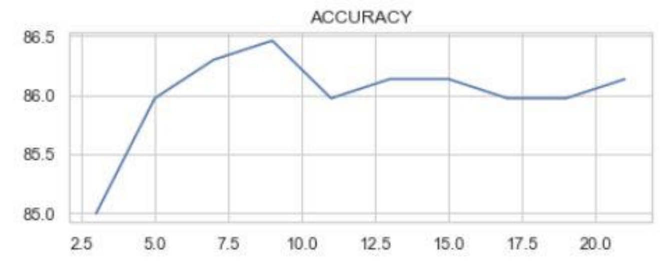
Similar to the imputation technique, KNN uses distance formula to predict the dependent variable.

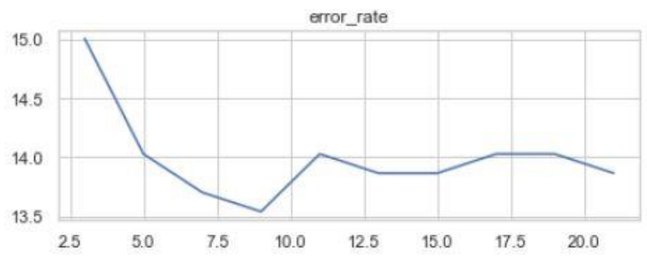
Following are the code for generating the K-Nearest Neighbours ML:
Python:

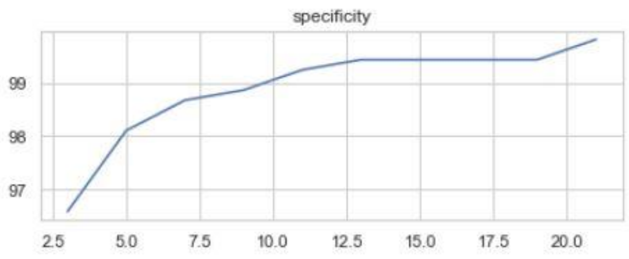
```
from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(x_train,y_train)
knn_estimated = KNN_model.predict(x_test)
```

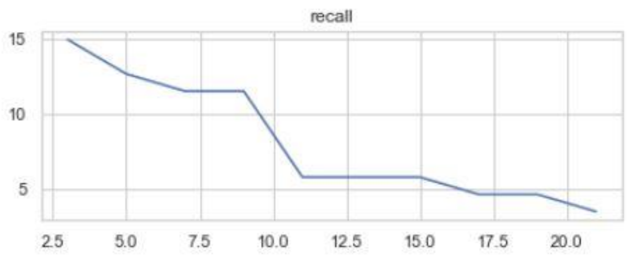
```
knn_result = model_acc(y_test,knn_estimated)

from sklearn.neighbors import KNeighborsClassifier
KNN_model2 = KNeighborsClassifier(n_neighbors = 9).fit(x_train2,y_train2)
knn_estimated2 = KNN_model2.predict(x_test2)
knn_result2 = model_acc(y_test2,knn_estimated2)
```

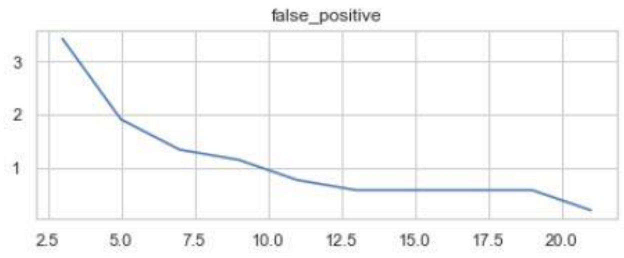
KNN Process 1

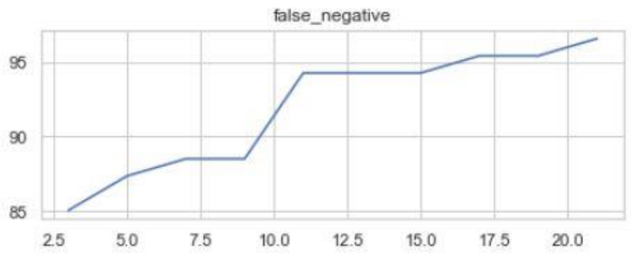




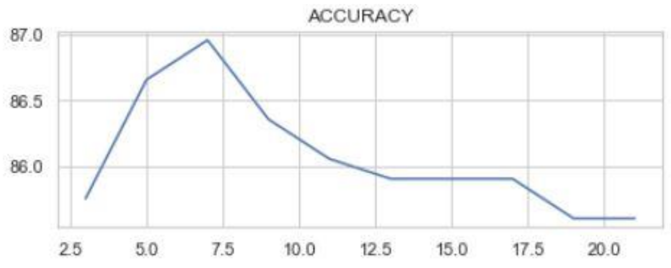


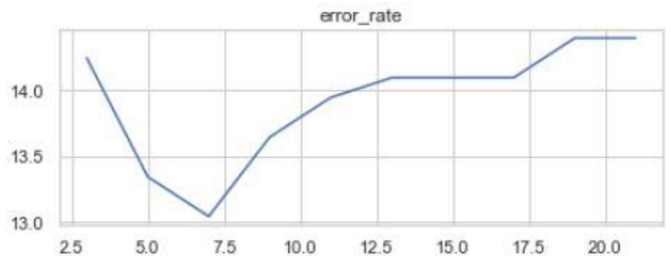


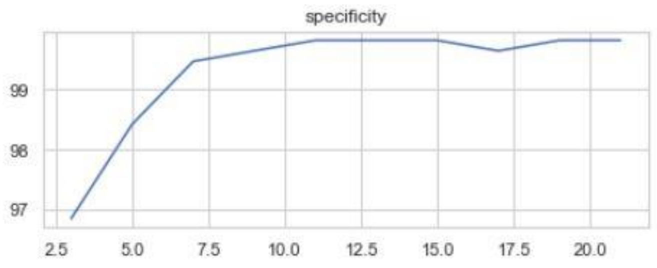


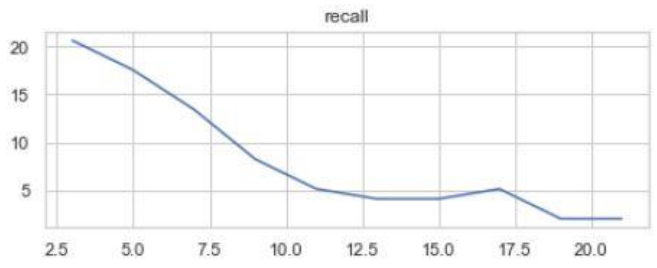


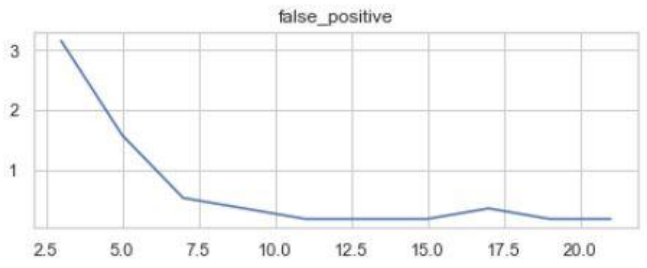
KNN Process 2

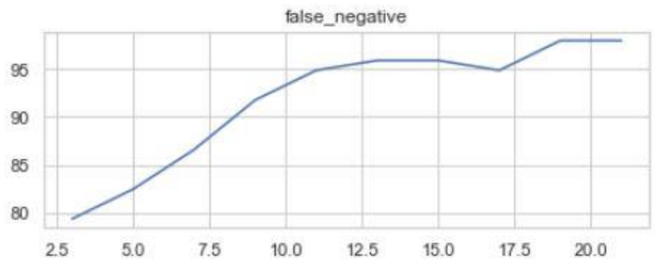












Output:

```
=============================================
Accuracy :86.46%             Error Rate :13.54%
Specificity :98.86%   Recall :11.49%
False Positive :1.14% Fasle Negative :88.51%
=============================================
[[520   6]
 [ 77  10]]


=============================================
Accuracy :86.36%             Error Rate :13.64%
Specificity :99.65%   Recall :8.25%
False Positive :0.35% Fasle Negative :91.75%
=============================================
[[568   2]
 [ 89   8]]
```

R code:

```
#Process1
for (i in c('state', 'international.plan','voice.mail.plan' ))
{
  train[,i] = as.factor(as.numeric(train[,i]))
  test[,i] = as.factor((as.numeric(test[,i])))
}
knn_predict1 = class::knn(train[,1:14], test[,1:14], train$Churn, k = 7)
caret::confusionMatrix(test$Churn, knn_predict1)

#Accuracy : 0.8969

#Process2
for (i in c('state', 'international.plan','voice.mail.plan' ))
{
  train2[,i] = as.factor(as.numeric(train2[,i]))
  test2[,i] = as.factor((as.numeric(test2[,i])))
}
knn_predict2 = class::knn(train2[,1:14], test2[,1:14], train2$Churn, k = 7)
caret::confusionMatrix(test2$Churn, knn_predict2)

#Accuracy : 0.8592

# Similar phenomena observed in KNN modeling
# Process1 produce better respect to Process2
```

Output:

```
#Process1
> caret::confusionMatrix(test$Churn, knn_predict1)
Confusion Matrix and Statistics

          Reference
```

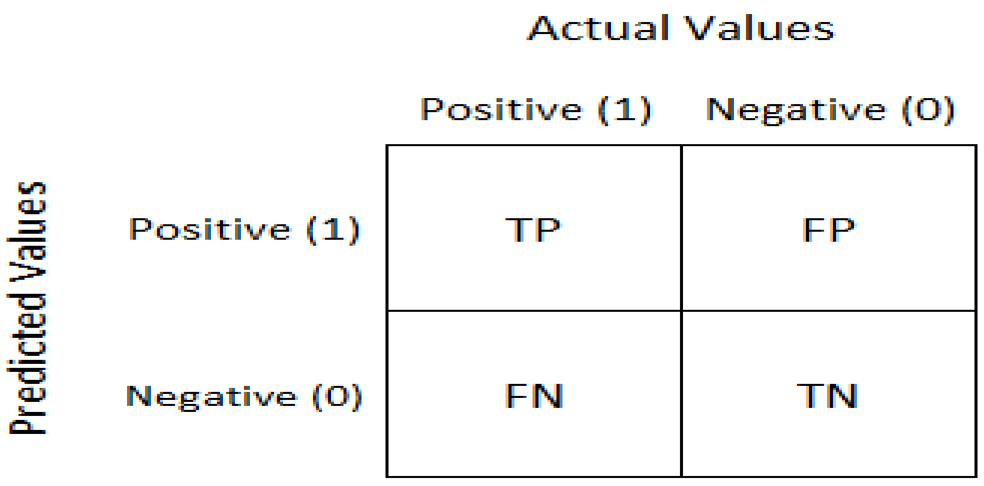
```
Prediction    0    1
         0 1110    3
         1  125    4

               Accuracy : 0.8969
                 95% CI : (0.8787, 0.9133)
    No Information Rate : 0.9944
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0487
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.89879
            Specificity : 0.57143
         Pos Pred Value : 0.99730
         Neg Pred Value : 0.03101
             Prevalence : 0.99436
         Detection Rate : 0.89372
   Detection Prevalence : 0.89614
      Balanced Accuracy : 0.73511

       'Positive' Class : 0
#Process2
> caret::confusionMatrix(test2$Churn, knn_predict2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1273   14
         1  196   16

               Accuracy : 0.8599
                 95% CI : (0.8413, 0.8771)
    No Information Rate : 0.98
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1007
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.86658
            Specificity : 0.53333
         Pos Pred Value : 0.98912
         Neg Pred Value : 0.07547
             Prevalence : 0.97999
         Detection Rate : 0.84923
   Detection Prevalence : 0.85857
      Balanced Accuracy : 0.69995

       'Positive' Class : 0
```

3. Conclusion

3.1. Model Evaluation

As This is classification problem we are using following parameter to evaluate the performance of the model and how well the value is predicted with respect to the machine learning algorithm.

Confusion Matrix: Confusion matrix is a table that used to describe the performance of the classification model



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

- Accuracy = TP+TN/ Total observations = 1 –Error rate
  (How accurately model can able to classify)

- Error rate = Miss classifieds/Total observations
  (classifying a record as belonging to one class when it belongs to another class.)

- Specificity = TN/TN+FP
  (The proportion of actual negative cases which are correctly identified.)

- Recall = TP/TP+FN
  (The proportion of actual positive cases which are correctly identified.)

- False Positive rate/Type-1 Error = FP/FP+TN
  (The proportion of negative which yield positive test outcomes with the test.)

- False Negative rate/ Type-2 Error = FN/FN+TP
  (The proportion of positives which yield negative test outcomes with the test.)

3.2 Model Selection

As per the output of all the algorithm, random forest tends to perform better as compare to the other algorithms. The accuracy of the model is 95%.

Python Ecosystem:

As per the respective python 3 ecosystem, The Random forest algorithm performance is quite high as compare to the other algorithms. Moreover, Outlier Imputation technique tends to produce slight high accuracy as compare to the outlier deletion technique.
So, with respect to python ecosystem, Random forest with outlier imputation technique is selected for the respective data set.

R Ecosystem:

As per the R ecosystem, The Random forest algorithm performance is quite high as compare to the other algorithms. Moreover, Outlier deletion technique tends to produce high accuracy as compare to the outlier imputation technique. All the algorithm performance parameters of outlier deletion model were quite high/better as compared to the outlier imputation models. Difference of approx. 2% was observed with respect to the algorithms.
So, with respect to R ecosystem, Random forest with outlier deletion technique is selected for the respective data set.

Appendix A – Python Code

```
#!/usr/bin/env python
# coding: utf-8

# # PROJECT 3 : CHURN REDUCTION

# ### Introduction
# ___The objective of this Case is to predict customer behaviour and to
develop an algorithm to predict the churn score based on usage pattern.___

# ## Library Importing

# In[1]:


import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import chi2_contingency
os.chdir('D:/Data Science/EDWISOR/2_PORTFOLIO/project 3')


# ## Self-Defined Function

# In[2]:


#Creating the function
def model_acc(actual,predict):
    from sklearn.metrics import confusion_matrix
    tn, fp, fn, tp = confusion_matrix(actual, predict).ravel()
    total= tn+fp+fn+tp
    accuracy = ((tp+tn)/total)*100
    error_rate = ((fp+fn)/total)*100
    specificity = (tn/(tn+fp))*100
    recall = (tp/(tp+fn))*100
    false_positive = (fp/(fp+tn))*100
    false_negative = (fn/(fn+tp))*100
    print('============================================\nAccuracy
:{acc}%\t\tError Rate :{error}%\nSpecificity :{spe}%\tRecall :{rec}%\nFalse
Positive :{falp}%\tFasle Negative
:{faln}%\n============================================'.          format(acc
= round(accuracy,2), error=
round(error_rate,2),spe=round(specificity,2),rec=round(recall,2),
falp=round(false_positive,2),faln=round(false_negative,2)))

    print(confusion_matrix(actual, predict))
    return accuracy, error_rate, specificity, recall, false_positive,
false_negative
#############################################################################
##############
# Normalized the the data set with respect to the columns porvided
def normalize(data,columns):
    for i in columns:
        if i in list(data.columns):
            print(i)
            minimum , maximum = data[i].min(), data[i].max()
            data[i] = (data[i] - minimum)/(maximum - minimum)
    return data


# ## IMPORTING THE DATA SET
# ## 1. EXPLORATORY DATA ANALYSIS

# In[3]:


Train_data = pd.read_csv( 'Train_data.csv', header = 0)
Train_data.columns = Train_data.columns.str.replace(' ','_')
Test_data = pd.read_csv('Test_data.csv', header = 0)
Test_data.columns = Test_data.columns.str.replace(' ','_')
print('Train data size is', Train_data.shape)
print('Test data size is', Test_data.shape)
Train_data.head(3)


# ### __Observations__
# 1. __Train data set has 3333 observations and 21 feature and, test data set
has 1667 observations and same number of features.__
# 2. __As per the above chunk of an data set, some of the features are
categorical in nature and others are numerical.__
#
#
# Categorical features: State, area_code, international_plan, voice_mail_plan
and Churn.
#
# Numerical
features:'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', and
'total_intl_charge'
#

# In[4]:


categorical_col =
['state','area_code','international_plan','voice_mail_plan', 'Churn']
numerical_col =
['number_vmail_messages','total_day_minutes','total_day_calls','total_day_cha
rge',
'total_eve_minutes','total_eve_calls','total_eve_charge','total_night_minutes
',
'total_night_calls','total_night_charge','total_intl_minutes','total_intl_cal
ls',                    'total_intl_charge' ]


# In[5]:


Train_data = Train_data.drop(['phone_number'], axis = 1)


# In[6]:
```

```
for col in categorical_col:
    Train_data[col] = pd.Categorical(Train_data[col])
# CONVERTING THE DATA TYPE INTO CATEGORICAL


# In[7]:


plt.figure(1)
plt.figure(figsize=(17,7))
plt.subplot(211)
sns.countplot(Train_data['state'])

plt.figure(2)
plt.figure(figsize=(17,7))
plt.subplot(221)
sns.countplot(Train_data['area_code'])
plt.subplot(222)
sns.countplot(Train_data['international_plan'])
plt.figure(3)
plt.figure(figsize=(17,7))
plt.subplot(223)
sns.countplot(Train_data['voice_mail_plan'])
plt.subplot(224)
sns.countplot(Train_data['Churn'])
plt.show()


# In[8]:


from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
Train_data['state'] = label.fit_transform(Train_data['state'])
Train_data.Churn = label.fit_transform(Train_data.Churn)
Train_data.voice_mail_plan = label.fit_transform(Train_data.voice_mail_plan)
Train_data.area_code = label.fit_transform(Train_data.area_code)
Train_data.international_plan =
label.fit_transform(Train_data.international_plan)

for col in categorical_col:Train_data[col] = pd.Categorical(Train_data[col])


# In[9]:


Train_data.describe()


# #### Observation:
# Minimum value and maximum value of the recpective features varies too much.
# Data set should be normalized before applying any machine learning
algorithms

# In[10]:


Train_data.hist(figsize=(20,20))
```

```
# #### Observation
# Data distribution of not normal
# total_intli_call and number_customer_service_calls are skewed towards right
(POSITIVELY SKEWED VARAIBLE).

# # DATA PRE-PROCESSING

# ## 1. MISSING VALUE CHECK

# In[11]:


Train_data.isna().sum()


# __OBSERVATION__
# 1. As per the above statement our data set do not have any missing value.

# ## 2. OUTLIER ANALYSIS

# In[12]:


count=0
for i in numerical_col:
    q75,q25 = np.percentile(Train_data.loc[:,i],[75,25])
    iqr=q75-q25
    l_fence,u_fence = round(q25-(iqr*1.5),4),round(q75+(iqr*1.5),4)
    total_outlier = len([num for num in Train_data[i] if num > u_fence or num
< l_fence ])
    if total_outlier != 0:

print('+++=============================================================================
======================+++')
        print ('OUTLIER PRESENT IN ',i)
        plt.figure(figsize=(13,0.8))
        sns.boxplot(Train_data[i])
        plt.show()
        print('25% :{}   Median :{}   75% :{}   IQR :{}'.
format(q25,Train_data[i].median(),q75,iqr) )
        print('Minimum :{}   Lower Fence :{}   Upper Fence :{}   Maximum :{}'
.format(Train_data[i].min(),l_fence,u_fence,Train_data[i].max()))
        print('Number of Outliers:{}   Percentage of Outlier in {} :{}%'
.format(total_outlier,i,round(total_outlier/len(Train_data),2)*100))
        count = count + total_outlier

    else: print('+++++++----NO OUTLIER IN --->',i)


print('%'*40)
print('Total outliers is {}'.format(count))

print('Percentage of outliers is
{}%'.format(round(count/Train_data.shape[0],2)*100))


# ### Process 1 Outlier deletation

# In[13]:
```

```
Process_data = Train_data.copy()
for i in numerical_col:
    print(i)
    q75, q25 = np.percentile(Process_data.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min, max = q25 - (iqr*1.5),q75 + (iqr*1.5)
    print('Maximim :{ma}\tMinimum :{mi}'.format( ma= max, mi= min))
    Process_data = Process_data.drop(Process_data[Process_data.loc[:,i] <
min].index)
    Process_data = Process_data.drop(Process_data[Process_data.loc[:,i] >
max].index)
    #print(i, '--> DONE')


# In[14]:


print(Process_data.shape)
print(((3333-3064)/3333)*100)


# 8% of observation decreased

# ### Process 2 Outlier imputation

# In[15]:


Process_data2 = Train_data.copy()
for i in numerical_col:
    print(i)
    q75_1, q25_1 = np.percentile(Process_data2.loc[:,i], [75 ,25])
    iqr2 = q75_1 - q25_1
    minimum, maximum = q25_1 - (iqr2*1.5),q75_1 + (iqr2*1.5)
    print('Maximim:', maximum)
    print('Minimum:', minimum)
    Process_data2[i] = np.where(Process_data2[i] < minimum , np.nan,
Process_data2[i])
    Process_data2[i] = np.where(Process_data2[i] > maximum , np.nan,
Process_data2[i])


# In[16]:


Process_data2.isna().sum()


# In[17]:


from fancyimpute import KNN


# In[18]:


Process_data2 = pd.DataFrame(KNN(k = 3).complete(Process_data2),
columns=Process_data2.columns)
```

```
# In[19]:


Process_data2.isna().sum()


# In[ ]:





# ## 3. FEATURE SELECTION

# #### 1. Feature selection on numerical varaibles

# In[20]:


plt.figure(figsize=(20,20))
corr = Process_data.corr()
sns.heatmap(corr, annot= True,mask = np.zeros_like(corr, dtype = np.bool),
vmin = -1, vmax = 1 )
plt.show()


# ### Observation:
# __As expected, with repect to the above figure following feature are highly
correlated with each other :__
# 1. total_day_charge and total_day_minutes     (Correlation : 1)
# 2. total_eve_charge and total_eve_minutes     (Correlation : 1)
# 3. total_night_charge and total_night_minutes  (Correlation : 1)
# 4. total_intl_charge and total_intl_minutes    (Correlation : 1)
#
# __It's obvisous, as the amount charges by the operators are directly
proportaional to total minutes__

# In[21]:


Process_data =
Process_data.drop(['total_day_minutes','total_eve_minutes','total_night_minut
es','total_intl_minutes'], axis=1)


Process_data2 =
Process_data2.drop(['total_day_minutes','total_eve_minutes','total_night_minu
tes','total_intl_minutes'], axis=1)



# #### 2. Feature selection on Categorical varaibles

# In[22]:


for col in categorical_col:
    if col != 'Churn':
        chi2, p, dof, ex =
chi2_contingency(pd.crosstab(Process_data['Churn'], Process_data[col]))
        if p<0.05: print('{}:{}'.format(col,p))
        else: print('{}:{}\t**Feature should be remove'.format(col,p))


# In[23]:


Process_data = Process_data.drop(['area_code'], axis=1)


Process_data2 = Process_data2.drop(['area_code'], axis=1)


# In[24]:


Process_data.head(5)


# ### 4. NORMALISATION

# In[25]:


normalized_list = []
for col in numerical_col: normalized_list.append(col)
normalized_list.append('account_length')


# In[26]:


Process_data = normalize(Process_data, normalized_list)


# In[27]:


Process_data.describe()


# In[28]:


Process_data2 = normalize(Process_data2, normalized_list)

# ### Sampling
# Stratified sampling

# In[29]:


from sklearn.model_selection import train_test_split


# In[30]:


independent_data = Process_data.drop(['Churn'], axis=1)
dependent_data = Process_data['Churn']
x_train,x_test,y_train,y_test =
train_test_split(independent_data,dependent_data,
                                                 test_size = 0.2,
random_state = 0, stratify = dependent_data  )


# In[31]:


independent_data2 = Process_data2.drop(['Churn'], axis=1)
dependent_data2 = Process_data2['Churn']
x_train2,x_test2,y_train2,y_test2 =
train_test_split(independent_data2,dependent_data2,
                                                 test_size = 0.2,
random_state = 0, stratify = dependent_data2  )


# ## MODEL DEVELOPMENT AND MODEL EVALUATION (CLASSIFICATION MODEL)

# As the dependent varaible is categorical varaibles, model selection will be
done based on the accuracy of classification models.
# Following Machine Learning algorithm will be use to develop the
classfication model:
# 1. Logistic Regression Algorithm.
# 2. Decision Tree Algorithm.
# 3. Random Forest Algorithm.
# 4. k-Nearest Neighbors Algorithm

# ## We have two data set
# #### 1. Process1 - process1 involves outlier deletation technique
# #### 2. Process2 - process2 involves outlier imputation technique
# #### Other parameter remains same

# # __1. Logistic Regression Algorithm__

# ### Process 1

# In[32]:


#Outlier Deleted
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression().fit(x_train,y_train)
logistic_estimated = logistic.predict(x_test)
logistic_result = model_acc(y_test,logistic_estimated)


# ### Process 2

# In[33]:


#Outlier Imputed
from sklearn.linear_model import LogisticRegression
logistic2 = LogisticRegression().fit(x_train2,y_train2)
logistic_estimated2 = logistic2.predict(x_test2)
logistic_result2 = model_acc(y_test2,logistic_estimated2)


# # As per the above parameters
# ### Process1 and Process2 accurary is quite similar
# ### Process2 tend to perform better at some extent as compare to process1
in logistic regression
```

```
# In[34]:


#Experiment check

import statsmodels.api as sm
logit = sm.Logit(y_train , x_train.astype(float)).fit()
print(logit.summary2())
logit_predict = logit.predict(x_test.astype(float))
logit_value = logit_predict.copy()
logit_value['value'] = np.where(logit_value > 0.5 ,1 ,0)
logit_result = model_acc(y_test,logit_value['value'])


# #  __2. Decision Tree Algorithm__

# ### Process 1

# In[35]:


#Outlier Deleted
from sklearn import tree
DTree_model = tree.DecisionTreeClassifier().fit(x_train,y_train)
DTree_estimated = DTree_model.predict(x_test)
print('Decision Tree')
Decision_result = model_acc(y_test,DTree_estimated)


# In[36]:


print('Feature Importance')
for col, per in zip(x_train,DTree_model.feature_importances_):print('{c}
:{p}%'.format(c=col, p = round(per*100,2)  ))


# ### Process 2

# In[37]:


from sklearn import tree
DTree_model2 = tree.DecisionTreeClassifier().fit(x_train2,y_train2)
DTree_estimated2 = DTree_model2.predict(x_test2)
print('Decision Tree')
Decision_result2 = model_acc(y_test2,DTree_estimated2)


# In[38]:


print('Feature Importance')
for col, per in zip(x_train,DTree_model2.feature_importances_):print('{c}
:{p}%'.format(c=col, p = round(per*100,2)  ))


# # As per the above parameters
# ### Process1 and Process2 accurary is quite similar
# ### Process2 tend to perform better at some extent as compare to process1
in Decision Tree

# # __3. Random Forest Algorithm.__

# ### Process 1

# In[39]:



from sklearn.ensemble import RandomForestClassifier
RF_model = RandomForestClassifier(n_estimators= 50).fit(x_train,y_train)
RF_estimated = RF_model.predict(x_test)
rf_result = model_acc(y_test,RF_estimated)


# CHECKING THE OPKTIMUM NUMBER OF TREES BASED ON THE ACCUARACY PARAMETERS

# In[40]:


tree_numbers = [i*10 for i in range(1,15)]


# In[41]:


accuracy, error_rate, specificity, recall, false_positive, false_negative =
[],[],[],[],[],[]

for number in tree_numbers:
    from sklearn.ensemble import RandomForestClassifier
    model_build = RandomForestClassifier(n_estimators=
number).fit(x_train,y_train)
    model_estimated = model_build.predict(x_test)
    rf_result = model_acc(y_test,model_estimated)
    accuracy.append(rf_result[0])
    error_rate.append(rf_result[1])
    specificity.append(rf_result[2])
    recall.append(rf_result[3])
    false_positive.append(rf_result[4])
    false_negative.append(rf_result[5])


# In[42]:


np.array(accuracy).max()


# In[43]:


sns.set(style="whitegrid")
plt.figure(1)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('ACCURACY')
sns.lineplot(x = tree_numbers, y = accuracy)
plt.subplot(222)
plt.title('error_rate')
sns.lineplot(x = tree_numbers, y = error_rate )

plt.figure(2)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('specificity')
sns.lineplot(x = tree_numbers, y = specificity)
plt.subplot(222)
plt.title('recall')
sns.lineplot(x = tree_numbers, y = recall )

plt.figure(3)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('false_positive')
sns.lineplot(x = tree_numbers, y = false_positive)
plt.subplot(222)
plt.title('false_negative')
sns.lineplot(x = tree_numbers, y = false_negative )




# ### Process 2

# In[44]:


from sklearn.ensemble import RandomForestClassifier
RF_model2 = RandomForestClassifier(n_estimators= 50).fit(x_train2,y_train2)
RF_estimated2 = RF_model2.predict(x_test2)
rf_result2 = model_acc(y_test2,RF_estimated2)


# In[45]:


accuracy1, error_rate1, specificity1, recall1, false_positive1,
false_negative1 = [],[],[],[],[],[]

for number in tree_numbers:
    from sklearn.ensemble import RandomForestClassifier
    model_build = RandomForestClassifier(n_estimators=
number).fit(x_train2,y_train2)
    model_estimated = model_build.predict(x_test2)
    rf_result = model_acc(y_test2,model_estimated)
    accuracy1.append(rf_result[0])
    error_rate1.append(rf_result[1])
    specificity1.append(rf_result[2])
    recall1.append(rf_result[3])
    false_positive1.append(rf_result[4])
    false_negative1.append(rf_result[5])


# In[46]:


np.array(accuracy1).max()
```

```
# In[47]:


sns.set(style="whitegrid")
plt.figure(1)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('ACCURACY')
sns.lineplot(x = tree_numbers, y = accuracy1)
plt.subplot(222)
plt.title('error_rate')
sns.lineplot(x = tree_numbers, y = error_rate1 )

plt.figure(2)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('specificity')
sns.lineplot(x = tree_numbers, y = specificity1)
plt.subplot(222)
plt.title('recall')
sns.lineplot(x = tree_numbers, y = recall1 )

plt.figure(3)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('false_positive')
sns.lineplot(x = tree_numbers, y = false_positive1)
plt.subplot(222)
plt.title('false_negative')
sns.lineplot(x = tree_numbers, y = false_negative1 )



# # As per the above parameters
# ### Process1 and Process2 accurary is quite similar (2% higher accraracy)
# ### Process2 tend to perform better at some extent as compare to process1
in Random Forest

# ### 4. k-Nearest Neighbors Algorithm

# ### Process 1

# In[48]:


from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(x_train,y_train)
knn_estimated = KNN_model.predict(x_test)
knn_result = model_acc(y_test,knn_estimated)


# In[49]:


accuracy1, error_rate1, specificity1, recall1, false_positive1,
false_negative1 = [],[],[],[],[],[]
cluster_number = [3,5,7,9,11,13,15,17,19,21]
for number in cluster_number:
    from sklearn.neighbors import KNeighborsClassifier
    print('Number of cluster :{}'.format(number) )
    model_build = KNeighborsClassifier(n_neighbors =
number).fit(x_train,y_train)

    model_estimated = model_build.predict(x_test)

    result = model_acc(y_test,model_estimated)

    accuracy1.append(result[0])
    error_rate1.append(result[1])
    specificity1.append(result[2])
    recall1.append(result[3])
    false_positive1.append(result[4])
    false_negative1.append(result[5])


# In[50]:


print(np.array(accuracy1).max())


# In[51]:


sns.set(style="whitegrid")
plt.figure(1)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('ACCURACY')
sns.lineplot(x = cluster_number, y = accuracy1)
plt.subplot(222)
plt.title('error_rate')
sns.lineplot(x = cluster_number, y = error_rate1 )

plt.figure(2)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('specificity')
sns.lineplot(x = cluster_number, y = specificity1)
plt.subplot(222)
plt.title('recall')
sns.lineplot(x = cluster_number, y = recall1 )

plt.figure(3)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('false_positive')
sns.lineplot(x = cluster_number, y = false_positive1)
plt.subplot(222)
plt.title('false_negative')
sns.lineplot(x = cluster_number, y = false_negative1 )


# ### Process 2

# In[52]:


from sklearn.neighbors import KNeighborsClassifier
KNN_model2 = KNeighborsClassifier(n_neighbors = 9).fit(x_train2,y_train2)
knn_estimated2 = KNN_model2.predict(x_test2)
```

```
knn_result2 = model_acc(y_test2,knn_estimated2)


# In[53]:


accuracy1, error_rate1, specificity1, recall1, false_positive1,
false_negative1 = [],[],[],[],[],[]
cluster_number = [3,5,7,9,11,13,15,17,19,21]
for number in cluster_number:
    from sklearn.neighbors import KNeighborsClassifier
    model_build = KNeighborsClassifier(n_neighbors =
number).fit(x_train2,y_train2)
    print('Number of cluster :{}'.format(number) )
    model_estimated = model_build.predict(x_test2)

    result = model_acc(y_test2,model_estimated)

    accuracy1.append(result[0])
    error_rate1.append(result[1])
    specificity1.append(result[2])
    recall1.append(result[3])
    false_positive1.append(result[4])
    false_negative1.append(result[5])


# In[54]:


print(np.array(accuracy1).max())


# In[55]:


sns.set(style="whitegrid")
plt.figure(1)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('ACCURACY')
sns.lineplot(x = cluster_number, y = accuracy1)
plt.subplot(222)
plt.title('error_rate')
sns.lineplot(x = cluster_number, y = error_rate1 )

plt.figure(2)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('specificity')
sns.lineplot(x = cluster_number, y = specificity1)
plt.subplot(222)
plt.title('recall')
sns.lineplot(x = cluster_number, y = recall1 )

plt.figure(3)
plt.figure(figsize=(15,5))
plt.subplot(221)
plt.title('false_positive')
sns.lineplot(x = cluster_number, y = false_positive1)
plt.subplot(222)
plt.title('false_negative')
```

```
sns.lineplot(x = cluster_number, y = false_negative1 )


# # As per the above parameters
# ### Process1 and Process2 accurary is quite similar
# ### Process2 tend to perform better at some extent as compare to process1
in KNN

# # Final Inference:
#
# # As per the above analysis and result genrated.
# # Random Forest algorithm tend to provide the better result in respect to
this data set
#
# # Moreover, after the imption of outlier with KNN imputation the accuracy
of all the algorithm increases at some extent and in respect to random forest
the accraracy in 2% more as compare to outlier deletation technique
#
#
#
# # So the final model selection with respect to the data set will be Random
Forest and Outlier technique will be KNN imputation
```

Appendix B – R Code

```
####################CHURN REDUCTION####################
#Introduction
#The objective of this Case is to predict customer behaviour
#and to develop an algorithm to predict the churn score based on usage
pattern.

####################IMPORTING THE DATA SET####################
setwd('D:/Data Science/EDWISOR/2_PORTFOLIO/project 3')
Train_data = read.csv(file = 'Train_data.csv', header = TRUE)
Test_data = read.csv(file = 'Test_data.csv', header = TRUE)
data = rbind(Train_data, Test_data)
summary(data)
phone_numbers = data$phone.number
new_dataset  = subset(x = data, select = -phone.number)
DataCombine::rmExcept(c('new_dataset', 'phone_numbers'))
new_dataset = as.data.frame(new_dataset)
str(new_dataset)

new_dataset$Churn = as.factor(new_dataset$Churn)
new_dataset$Churn = as.numeric(new_dataset$Churn)
new_dataset$Churn[new_dataset$Churn == 1] = 0
new_dataset$Churn[new_dataset$Churn == 2] = 1

categorical_columns = c('state','area.code',
                        'international.plan','voice.mail.plan', 'Churn')
numerical_columns =
c('number.vmail.messages','total.day.minutes','total.day.calls','total.day.ch
arge',

'total.eve.minutes','total.eve.calls','total.eve.charge','total.night.minutes
',

'total.night.calls','total.night.charge','total.intl.minutes','total.intl.cal
ls','total.intl.charge')
#CONVERTING THE DATASET
for (col in categorical_columns){new_dataset[,col] =
as.factor(new_dataset[,col])}
str(new_dataset)
summary(new_dataset)
########################## EDA ##########################
############# MISSING VALUE CHECK #############
anyNA(new_dataset)
# RESULT IS FALSE NO MISSING VALUE PRESENT IN THE RESPECTIVE DATA SET


############# OUTLIER ANALYSIS #############
library(ggplot2)

numeric_index = sapply(new_dataset,is.numeric)
numeric_data = new_dataset[,numeric_index] # creating the dataset with
numerical varaible only
cnames = colnames(numeric_data) # fetching column names
for (i in 1:length(cnames)) # Ploting the boxplot
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"),
data = subset(new_dataset))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey"
,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames[i],x="Churn")+
           ggtitle(paste("Box plot of Churn for",cnames[i])))
}
rm(numeric_data,i)

#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,gn12, ncol = 3)
#gridExtra::grid.arrange(gn13,gn14,gn15, ncol = 3)

rm(gn1,gn2,gn3,gn4,gn5,gn6,gn7,gn8,gn9,gn10,gn11,gn12,gn13,gn14,gn15)

#Mostly all the feature has outliers
# SO WE HAVE TWO CHOICE TO DEAL WITH THE OUTLIERS
# 1. DELETE THE OUTLIERS FROM THE DATA SET AND PERFORM THE FURTHER ANALYSIS
# 2. IMPUTE THE OUTLIER WITH IMPTATION TECHNIQUE OR ANY STATISTICAL TECHNIQUE

# Process1 will have the deletation technique
Process1 =  new_dataset
for(i in cnames)
{
  val = Process1[,i][Process1[,i] %in% boxplot.stats(Process1[,i])$out]
  Process1 = Process1[which(!Process1[,i] %in% val),]
} # all the outliear got removed after applying this loop
#Around 17% of the outliear are removed


#Process2  will have the outlier imputation technique
Process2 = new_dataset
for(i in cnames)
{
  val = Process2[,i][Process2[,i] %in% boxplot.stats(Process2[,i])$out]
  Process2[,i][Process2[,i] %in% val] = NA
}

anyNA(Process2)
library(DMwR)
Process2 = knnImputation(Process2,k=5)

anyNA(Process2)


############# FEATURE SELECTION #############
library(corrplot)

#Feature selection for numerical type data set

corrplot(corr = cor(new_dataset[,numerical_columns]),method = 'number')

#Observation:
#As expected, with repect to the above figure following feature are highly
correlated with each other :

# 1. total_day_charge and total_day_minutes (Correlation : 1)
# 2. total_eve_charge and total_eve_minutes (Correlation : 1)
# 3. total_night_charge and total_night_minutes (Correlation : 1)
# 4. total_intl_charge and total_intl_minutes (Correlation : 1)
```

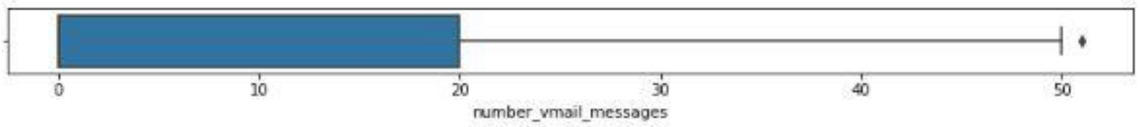
```
#It's obvisous, as the amount charges by the operators are directly
proportaional to total minutes

Process1 = subset(Process1, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,total.intl.minutes)
)

Process2 = subset(Process2, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,total.intl.minutes)
)

# Feature Selection for categorical data
factor_index = sapply(Process1, is.factor)
factor_data = Process1[, factor_index]

for (i in 1:length(colnames(factor_data)))
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}


#1] "area.code"
#Pearson's Chi-squared test
#data:  table(factor_data$Churn, factor_data[, i])
#X-squared = 0.013014, df = 2, p-value = 0.9935
# As p-value is greater than 0.05, Feature should be remove

Process1 = subset(Process1, select = -area.code)
Process2 = subset(Process2, select = -area.code)

############# FEATURE SCALING #############
#Checking the distribution of the numerical feature
#hist(Process1$number.vmail.messages) # HISTOGRAM_number.vmail.messages
#hist(Process1$total.day.calls) # HISTOGRAM_total.day.calls
#hist(Process1$total.day.charge) # HISTOGRAM_total.day.charge
#hist(Process1$total.eve.calls) # HISTOGRAM_total.eve.calls
#hist(Process1$total.night.calls) # HISTOGRAM_total.night.calls
#hist(Process1$total.night.charge) # HISTOGRAM_total.night.charge
#hist(Process1$total.intl.calls) # HISTOGRAM_total.intl.calls
#hist(Process1$total.intl.charge) # HISTOGRAM_total.intl.charge
#hist(Process1$number.customer.service.calls) #
HISTOGRAM_number.customer.service.calls

# INFERENCE:
# As THE DATA IS NOT NORMALIZED SO WE WILL USE Min-Max scaling METHOD


numerical_columns = 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")
#Normalisation
for (i in numerical_columns)
{
    print(i)
    Process1[,i] = (Process1[,i] - min(Process1[,i]))/(max(Process1[,i] -
min(Process1[,i])))
}
```

```
for (i in numerical_columns)
{
  print(i)
  Process2[,i] = (Process2[,i] - min(Process2[,i]))/(max(Process2[,i] -
min(Process2[,i])))
}






######################### MODEL DEVELOPMENT #########################

############# Spliting the dataset in test and train data #############
set.seed(123)
#Process1
train_index = caret::createDataPartition(Process1$Churn, p = 0.7, list =
FALSE)
train = Process1[train_index,]
test = Process1[-train_index,]
#Process2
train_index2 = caret::createDataPartition(Process2$Churn, p = 0.7, list =
FALSE)
train2 = Process2[train_index2,]
test2 = Process2[-train_index2,]


#MODEL DEVELOPMENT AND MODEL EVALUATION (CLASSIFICATION MODEL)
#As the dependent varaible is categorical varaibles, model selection will be
done based on the accuracy of classification models. Following Machine
Learning algorithm will be use to develop the classfication model:

#1 Logistic Regression Algorithm.
#2 Decision Tree Algorithm.
#3 Random Forest Algorithm.
#4 k-Nearest Neighbors Algorithm


############# 1 Logistic Regression Algorithm.#############
#Process1
logistic_model = glm(Churn~.,train, family = 'binomial')
summary(logistic_model)
logistic_predict1 = predict(logistic_model, test[,-15], type = 'response')
predicted_logit = as.factor(ifelse(logistic_predict1 > 0.5, 1,0))
caret::confusionMatrix(test$Churn, as.factor(predicted_logit))

#Accuracy : 0.9066

#Process2
logistic_model2 = glm(Churn~.,train2, family = 'binomial')
summary(logistic_model2)
logistic_predict2 = predict(logistic_model2, test2[,-15], type = 'response')
predicted_logit2 = as.factor(ifelse(logistic_predict2 > 0.5, 1,0))
caret::confusionMatrix(test2$Churn, as.factor(predicted_logit2))

#Accuracy : 0.8666

# With respect to the confusion matrix Process1 tent to perform far better as
compare to process2
```

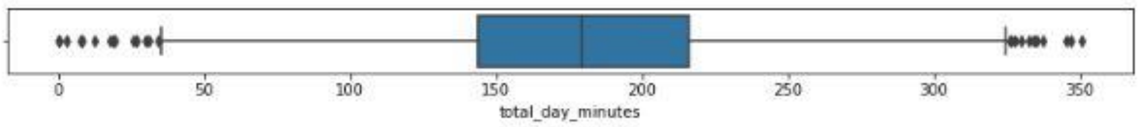
```
############# 2 Decision Tree Algorithm.#############
library(C50)
#Process1
c50model = C5.0(Churn~., train, trails = 100, rules = TRUE)
summary(c50model)
c50_predict1 = predict(c50model, test[,-15])

caret::confusionMatrix(test$Churn, c50_predict1)

#Accuracy : 0.9565
#Process2
c50model2 = C5.0(Churn~., train2, trails = 100, rules = TRUE)
summary(c50model2)
c50_predict2 = predict(c50model2, test2[,-15])

caret::confusionMatrix(test2$Churn, c50_predict2)

#Accuracy : 0.9213

# Similar phenomena observed in Decision tree modeling
# Process1 produce better respect to Process2

############# 3 Random Forest Algorithm.#############
#Process1
RF_model = randomForest::randomForest(x= train[,-15], y = train$Churn,
importance = TRUE, ntree = 500)
summary(RF_model)
rf_predict1 = predict(RF_model, test[,-15])
caret::confusionMatrix(test$Churn, rf_predict1)

#Accuracy : 0.9597

#Process2
RF_model2 = randomForest::randomForest(x= train2[,-15], y = train2$Churn,
importance = TRUE, ntree = 500)
summary(RF_model2)
rf_predict2 = predict(RF_model2, test2[,-15])
caret::confusionMatrix(test2$Churn, rf_predict2)

#Accuracy : 0.9213

# Similar phenomena observed in random forest modeling
# Process1 produce better respect to Process2

############# 4 k-Nearest Neighbors Algorithm############
#Process1
for (i in c('state', 'international.plan','voice.mail.plan' ))
{
  train[,i] = as.factor(as.numeric(train[,i]))
  test[,i] = as.factor((as.numeric(test[,i])))
}
knn_predict1 = class::knn(train[,1:14], test[,1:14], train$Churn, k = 7)
caret::confusionMatrix(test$Churn, knn_predict1)

#Accuracy : 0.8969

#Process2
for (i in c('state', 'international.plan','voice.mail.plan' ))
{
  train2[,i] = as.factor(as.numeric(train2[,i]))
  test2[,i] = as.factor((as.numeric(test2[,i])))
}
knn_predict2 = class::knn(train2[,1:14], test2[,1:14], train2$Churn, k = 7)
caret::confusionMatrix(test2$Churn, knn_predict2)

#Accuracy : 0.8592

# Similar phenomena observed in KNN modeling
# Process1 produce better respect to Process2


#Final Inference:

#As per the above analysis and result genrated.
#Random Forest algorithm tend to provide the better result in respect to this
data set
#Moreover, outlier imputed dataset preformance is quite low as compare to the
outlier deleted dataset
#In this scenario We are going to select the Random Forest Algorithm and we
will finalize the outlier deletation technique
```

Appendix C – Extra Figures

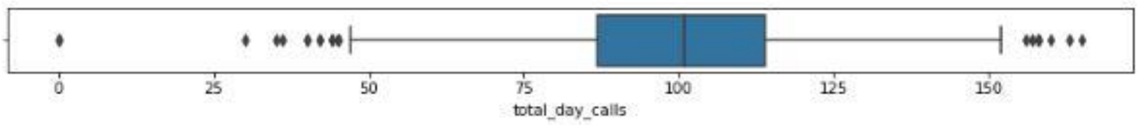
```
+++==========================================================================================+++
OUTLIER PRESENT IN  number_vmail_messages
```



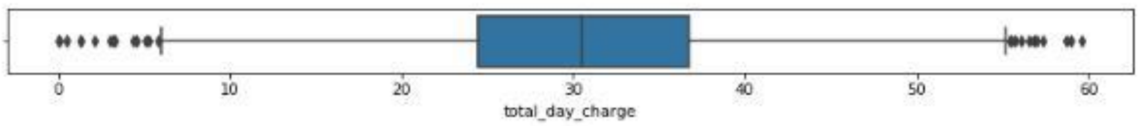
```
25% :0.0   Median :0.0   75% :20.0   IQR :20.0
Minimum :0   Lower Fence :-30.0   Upper Fence :50.0   Maximum :51
Number of Outliers:1   Percentage of Outlier in number_vmail_messages :0.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_day_minutes
```



```
25% :143.7   Median :179.4   75% :216.4   IQR :72.70000000000002
Minimum :0.0   Lower Fence :34.65   Upper Fence :325.45   Maximum :350.8
Number of Outliers:25   Percentage of Outlier in total_day_minutes :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_day_calls
```

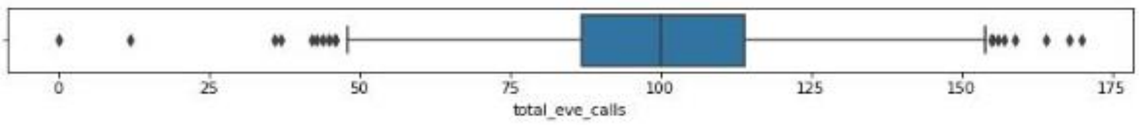


```
25% :87.0   Median :101.0   75% :114.0   IQR :27.0
Minimum :0   Lower Fence :46.5   Upper Fence :154.5   Maximum :165
Number of Outliers:23   Percentage of Outlier in total_day_calls :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_day_charge
```

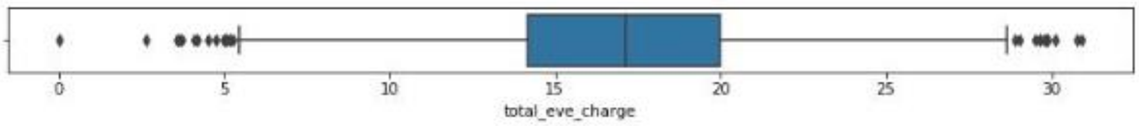


```
25% :24.43   Median :30.5   75% :36.79   IQR :12.36
Minimum :0.0   Lower Fence :5.89   Upper Fence :55.33   Maximum :59.64
Number of Outliers:25   Percentage of Outlier in total_day_charge :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_eve_minutes
```

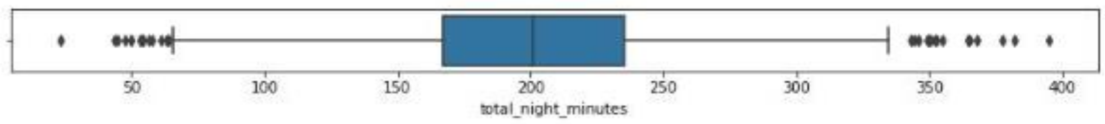
```
25% :166.6   Median :201.4   75% :235.3   IQR :68.70000000000002
Minimum :0.0   Lower Fence :63.55   Upper Fence :338.35   Maximum :363.7
Number of Outliers:24   Percentage of Outlier in total_eve_minutes :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_eve_calls
```



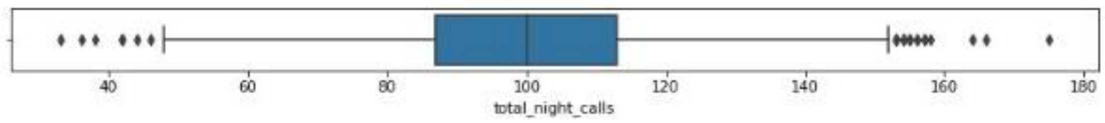
```
25% :87.0   Median :100.0   75% :114.0   IQR :27.0
Minimum :0   Lower Fence :46.5   Upper Fence :154.5   Maximum :170
Number of Outliers:20   Percentage of Outlier in total_eve_calls :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_eve_charge
```



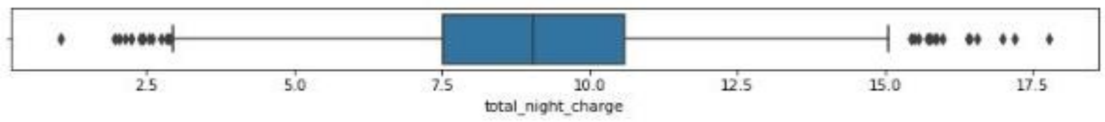
```
25% :14.16   Median :17.12   75% :20.0   IQR :5.84
Minimum :0.0   Lower Fence :5.4   Upper Fence :28.76   Maximum :30.91
Number of Outliers:24   Percentage of Outlier in total_eve_charge :1.0%
+++==========================================================================================+++
25% :14.16   Median :17.12   75% :20.0   IQR :5.84
Minimum :0.0   Lower Fence :5.4   Upper Fence :28.76   Maximum :30.91
Number of Outliers:24   Percentage of Outlier in total_eve_charge :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_night_minutes
```



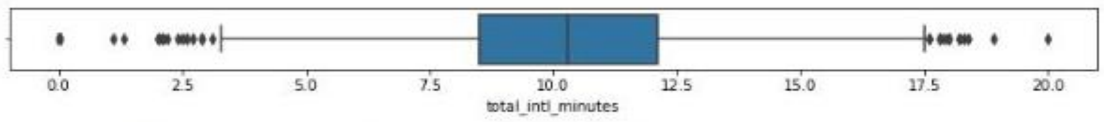
```
25% :167.0   Median :201.2   75% :235.3   IQR :68.30000000000001
Minimum :23.2   Lower Fence :64.55   Upper Fence :337.75   Maximum :395.0
Number of Outliers:30   Percentage of Outlier in total_night_minutes :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_night_calls
```



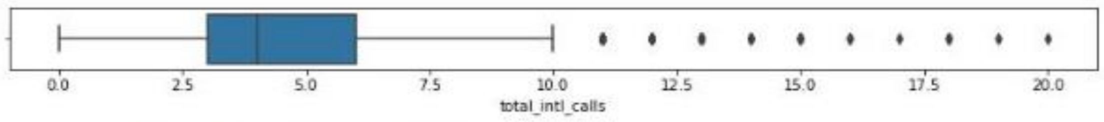
```
25% :87.0   Median :100.0   75% :113.0   IQR :26.0
Minimum :33   Lower Fence :48.0   Upper Fence :152.0   Maximum :175
Number of Outliers:22   Percentage of Outlier in total_night_calls :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_night_charge
```



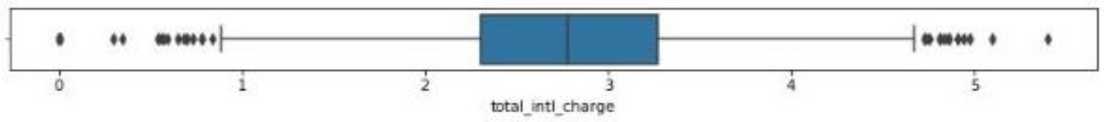
```
25% :7.52   Median :9.05   75% :10.59   IQR :3.0700000000000003
Minimum :1.04   Lower Fence :2.915   Upper Fence :15.195   Maximum :17.77
Number of Outliers:30   Percentage of Outlier in total_night_charge :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_intl_minutes
```



```
25% :8.5   Median :10.3   75% :12.1   IQR :3.5999999999999996
Minimum :0.0   Lower Fence :3.1   Upper Fence :17.5   Maximum :20.0
Number of Outliers:45   Percentage of Outlier in total_intl_minutes :1.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_intl_calls
```

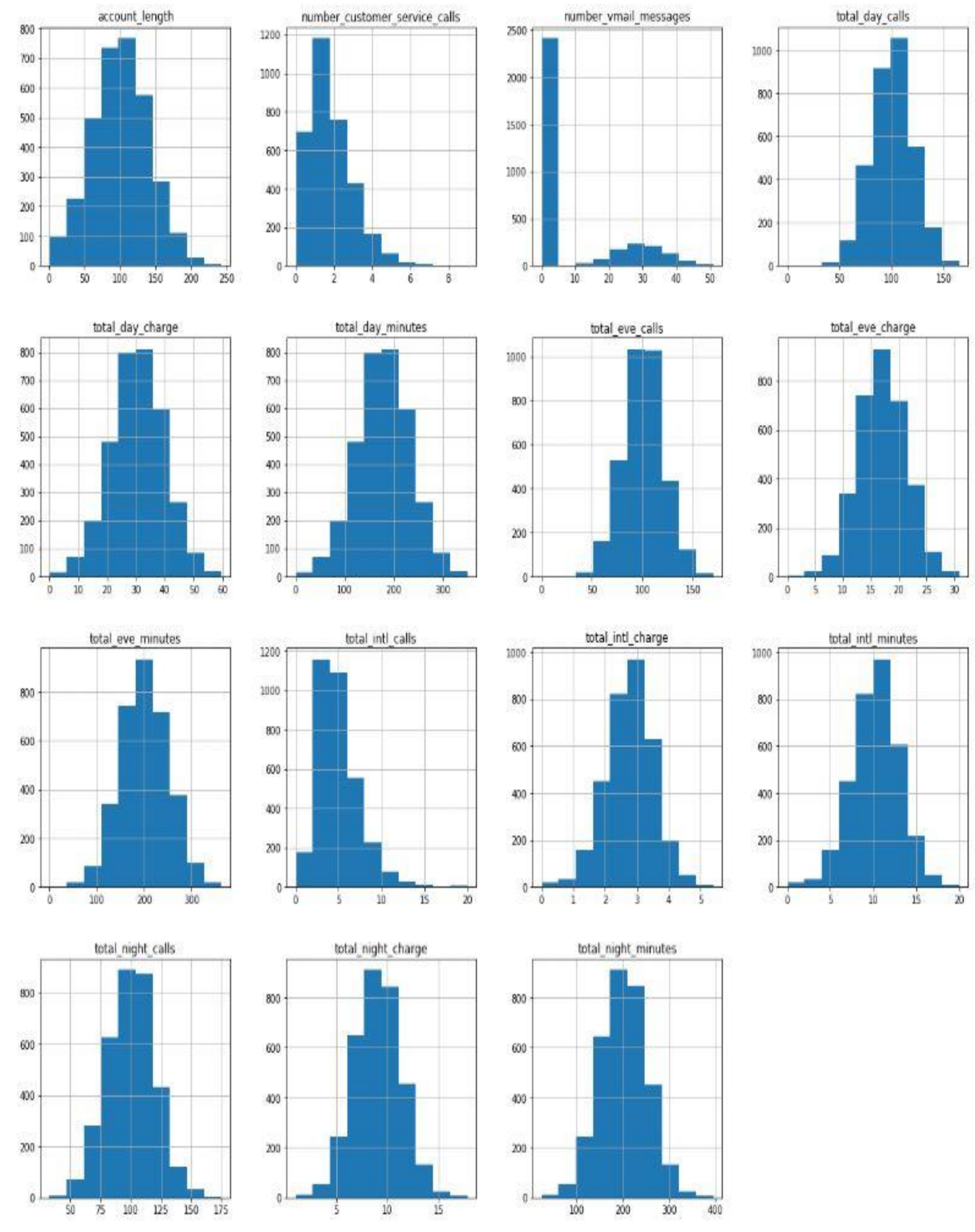


```
25% :3.0   Median :4.0   75% :6.0   IQR :3.0
Minimum :0   Lower Fence :-1.5   Upper Fence :10.5   Maximum :20
Number of Outliers:78   Percentage of Outlier in total_intl_calls :2.0%
+++==========================================================================================+++
OUTLIER PRESENT IN  total_intl_charge
```

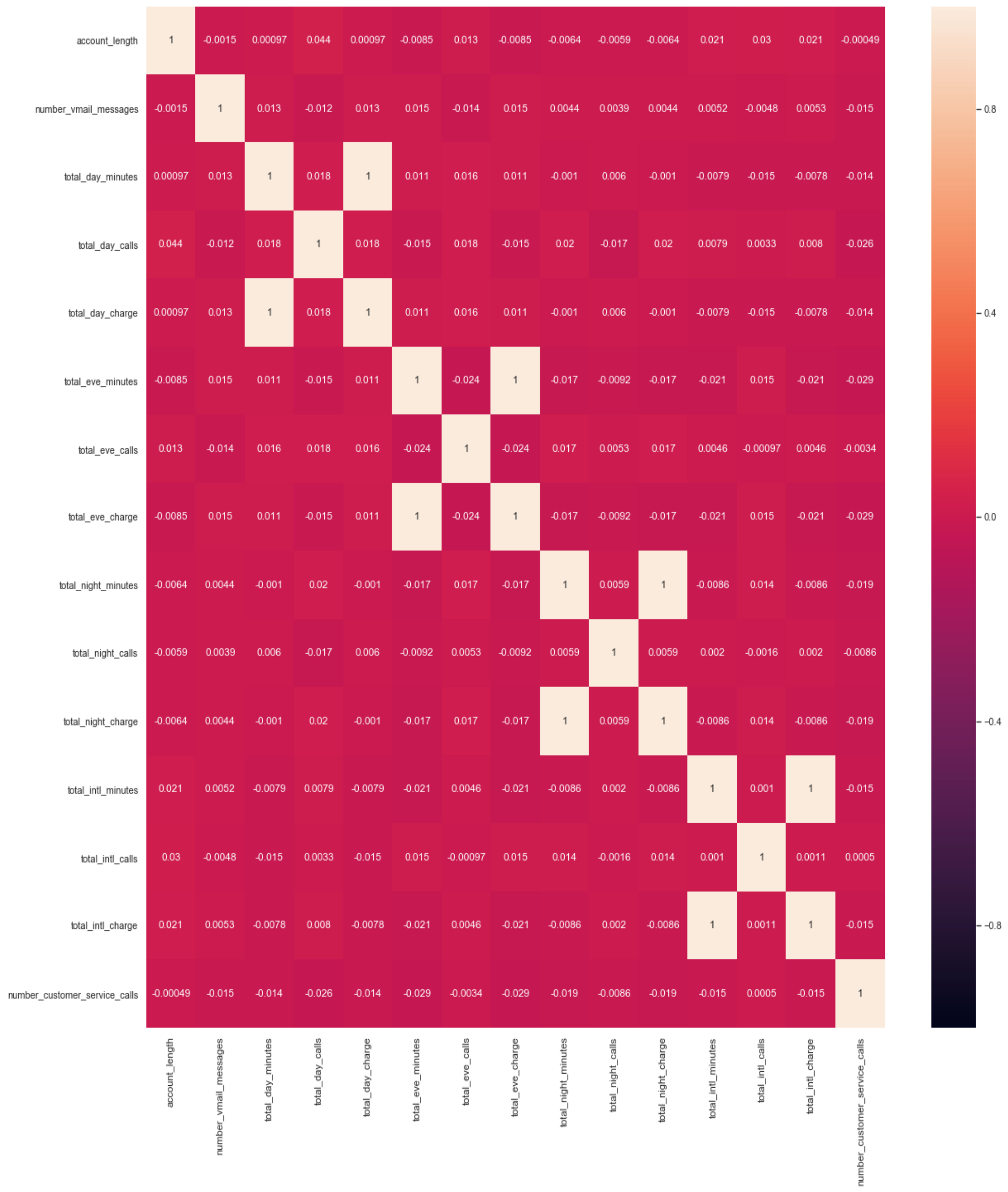


```
25% :2.3   Median :2.78   75% :3.27   IQR :0.9700000000000002
Minimum :0.0   Lower Fence :0.845   Upper Fence :4.725   Maximum :5.4
Number of Outliers:49   Percentage of Outlier in total_intl_charge :1.0%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Total outliers is 396
Percentage of outliers is 12.0%
```

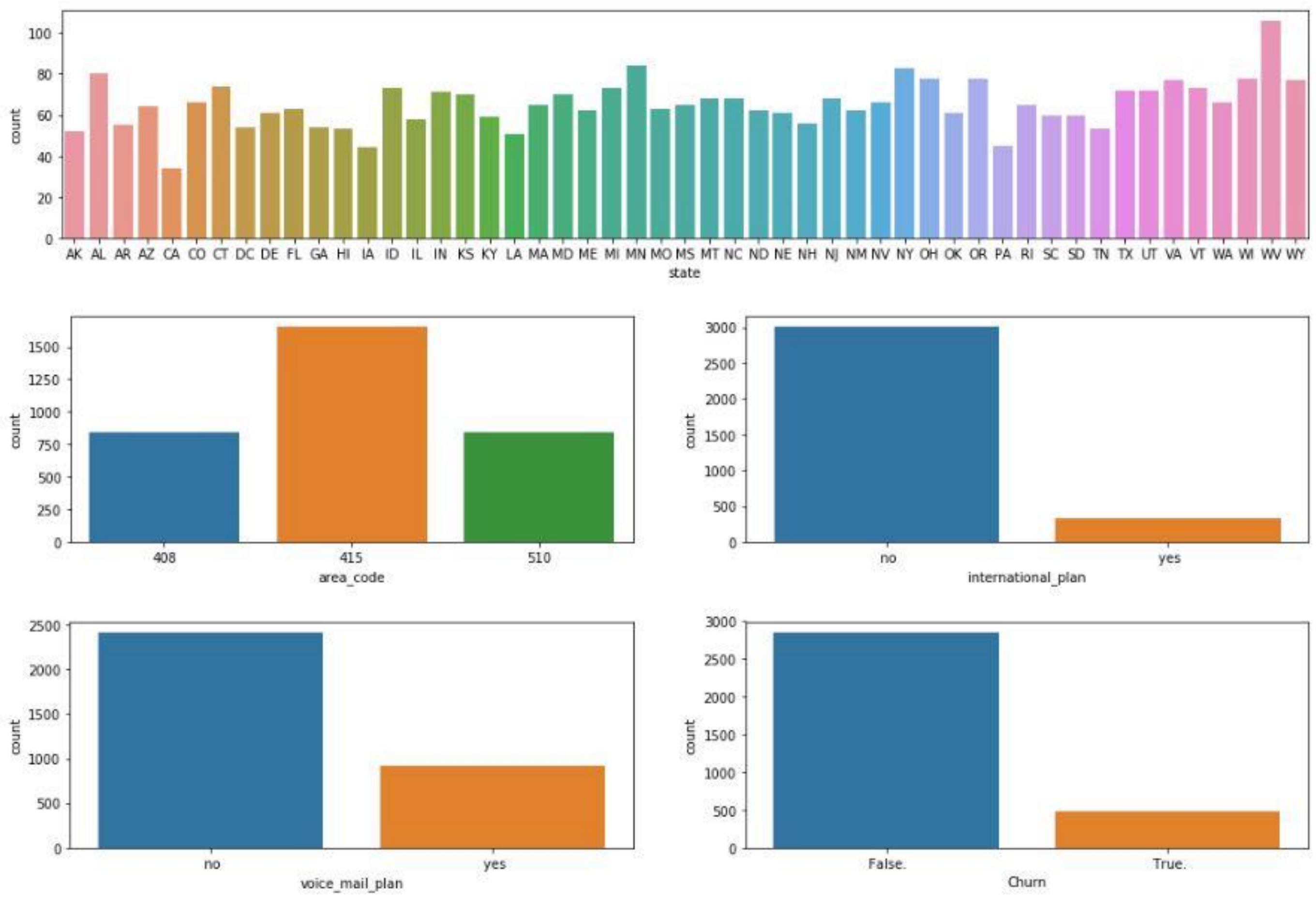
The above figure is the part of outlier analysis, comprises of box-whiskers plot with respect to the feature and also provides the information regarding number of outlier present in features.



The above figure is the histogram with respect to the feature in python. As the data distribution was not normal we used normalization method to feature scaling section.



The above figure is correlation heat map; this graph represents the correlation between the features. As seen in above figure correlation between some variable is high which was removed in feature selection section.



The above graph is the count plot of categorical feature.
This graph shows the distribution of unique variable and the frequency of the variable repeated in the respective features.
In 'State' graph the frequency of the variable, the fluctuation or the frequency is high with respect to 'WV' state and quite low with respect to 'CA' state.
In international plan count plot, we tend to know that the user who has international plan are quite low.
Similar trend was seen in voice mail plan, number of user who has voice mail plan activated are quite low as compare to the others.
'Churn' count plot is highly important as this show the imbalance of the data set, as this is our dependent variable and "False" has a high frequency as compare to "True". Due to this pattern we have used stratify sampling technique which split the data is same ratio as of the dependent variable.