Project Report

Churn reduction

By

Sonnet Padayatty Reju

22nd July 2018

Contents

Chapter 1

Introduction

1.1 Project 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.
Data Sets –
1) Test_data.csv
2) Train_data.csv

1.2 Problem Statement

The objective of this Case is to predict customer behavior. 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.

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

- number of customer service calls made

Target Variable
Churn : move: if the customer has moved (1=yes; 0 = no)

Chapter 2

Methodology

2.1 Pre-Processing

Pre-processing of data is an indispensable stage in predictive analysis. Since the predictive model needs to handle a big data set, it is always necessary to eliminate unwanted data. There may be many variables whose data type is incorrect and may create complexity while training the predictive model with train dataset. In order to minimize such issues in modeling stage, we conduct data pre-processing and extract important insights from the raw data. We can extract such information by analyzing the independent variables using probability density function or by visualizing how the data points have been distributed in each variable. It can be easily achieved by checking the normality using Tableau or other normality checking functions.

Following are main pre-processing methods used in predictive analysis

2.1.1 Missing Value Analysis

Once we are done with pre-processing steps like "Renaming the variables" and "Converting into proper Data types", we can conduct the missing value analysis. You may use the combination of both train and test data for the imputation of missing values as it makes the model to predict the values more accurately. Mainly, there are 3 methods for imputation of missing values.

1. Mean method
2. Median method
3. KNN imputation

We are not supposed to do imputation if the percentage of missing values in a variable is more than 30%. Fortunately, in the given data set, there are no missing values under any variables as shown below.

Variable	number of missing values
State	0
account.length	0
area.code	0
phone.number	0
international.plan	0
voice.mail.plan	0
number.vmail.messages	0
total.day.minutes	0
total.day.calls	0
total.day.charge	0
total.eve.minutes	0
total.eve.calls	0
total.eve.charge	0
total.night.minutes	0
total.night.calls	0
total.night.charge	0
total.intl.minutes	0
total.intl.calls	0
total.intl.charge	0
number.customer.service.calls	0
Churn	0

2.1.2 Outlier Analysis

By definition, outliers are points that are distant from remaining observations. As a result, they can potentially skew or bias any analysis performed on the dataset. It is therefore very important to detect and adequately deal with outliers. In order to show the impact of outliers, we use a technique called box plot in which the distribution of data points is visualized. The box plots and histograms of each independent variable before and after outlier analysis are shown in the Appendix. Here, I would like to share the box plots and histograms of the variables " Number.Customer.Service Calls" and "Total intl. calls " in order to display the **effect of outliers** in each variable.

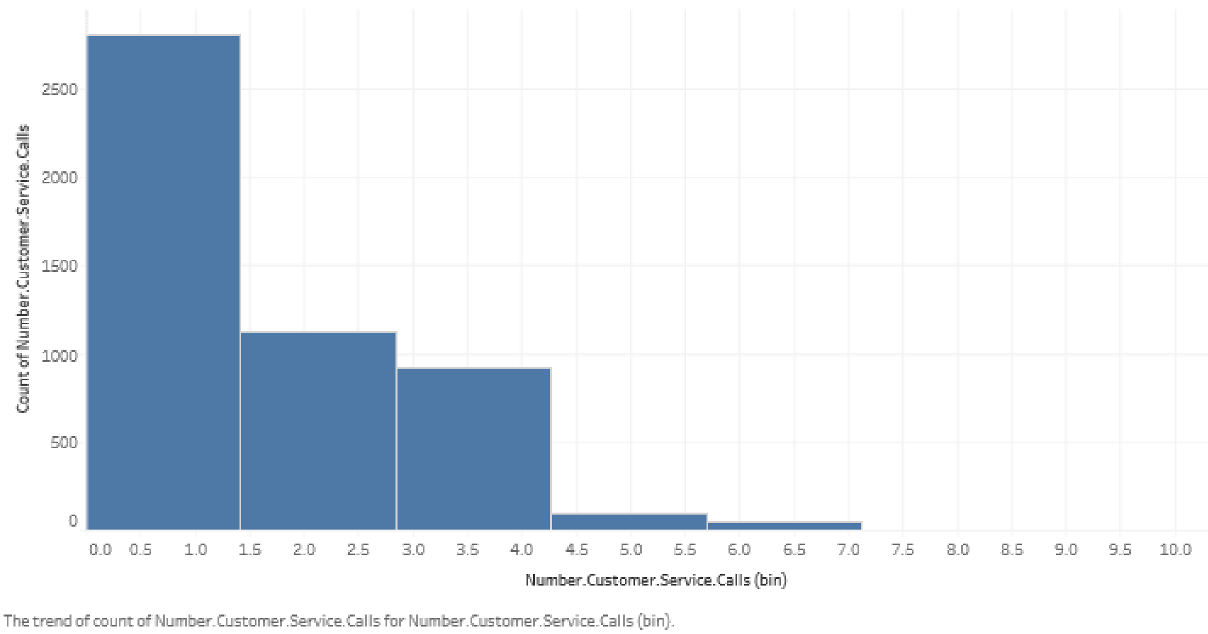


Fig:1 - Normality check of variable –"Number.Customer.Service Calls" before Outlier Analysis.

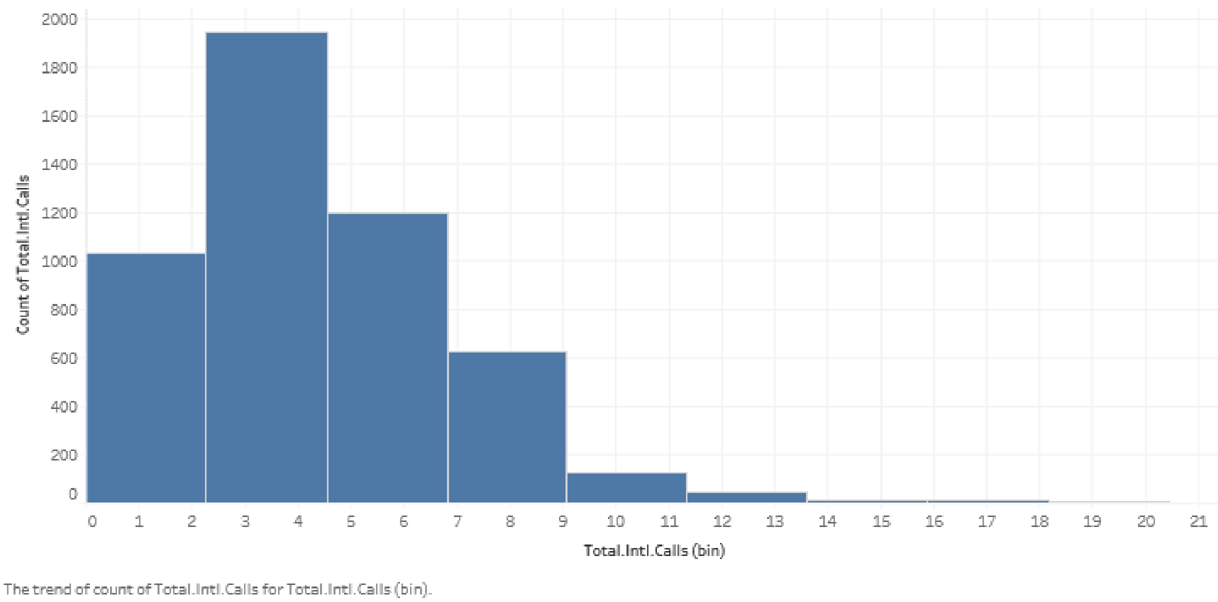


Fig: 2 - Normality check of variable –"Total Intl. Calls" before Outlier Analysis.

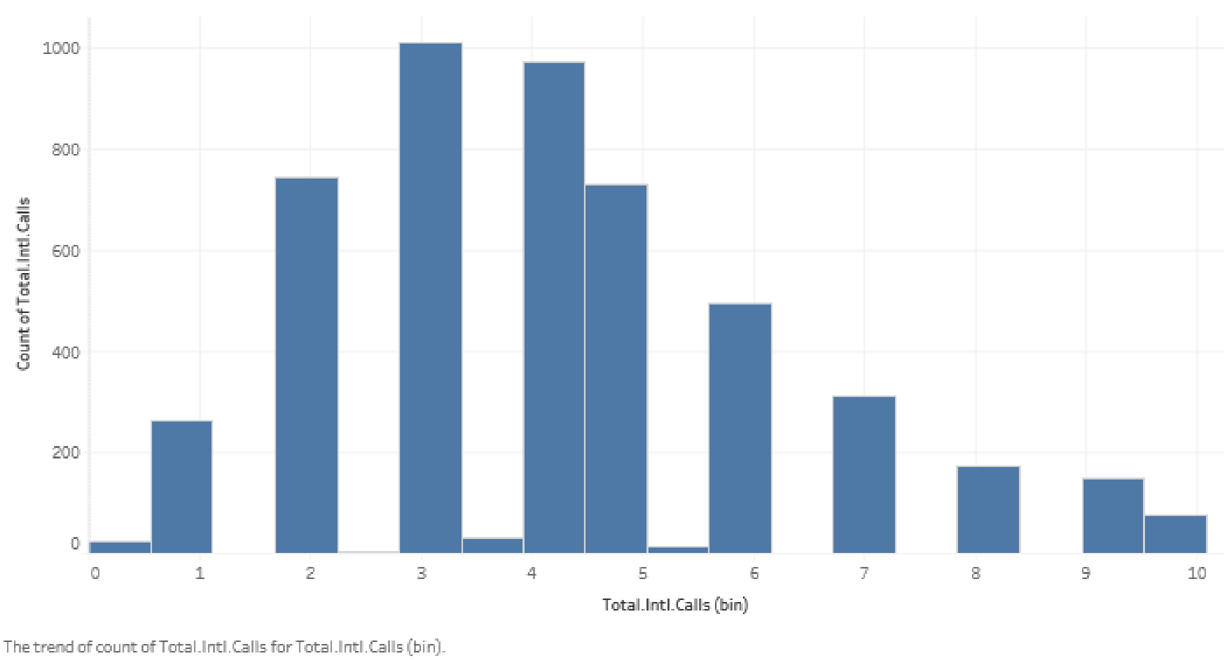


Fig:3- Normality check of variable –"Total Intl. Calls" after Outlier Analysis.

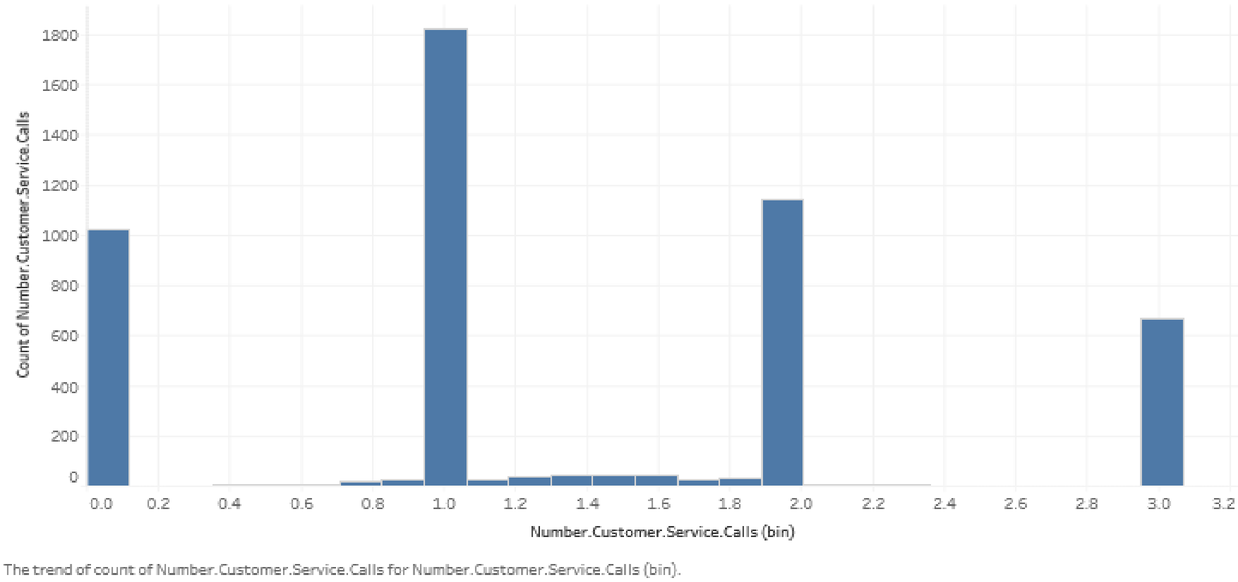


Fig: 4- Normality check of variable –"Number.Customer.Service Calls" after Outlier Analysis.

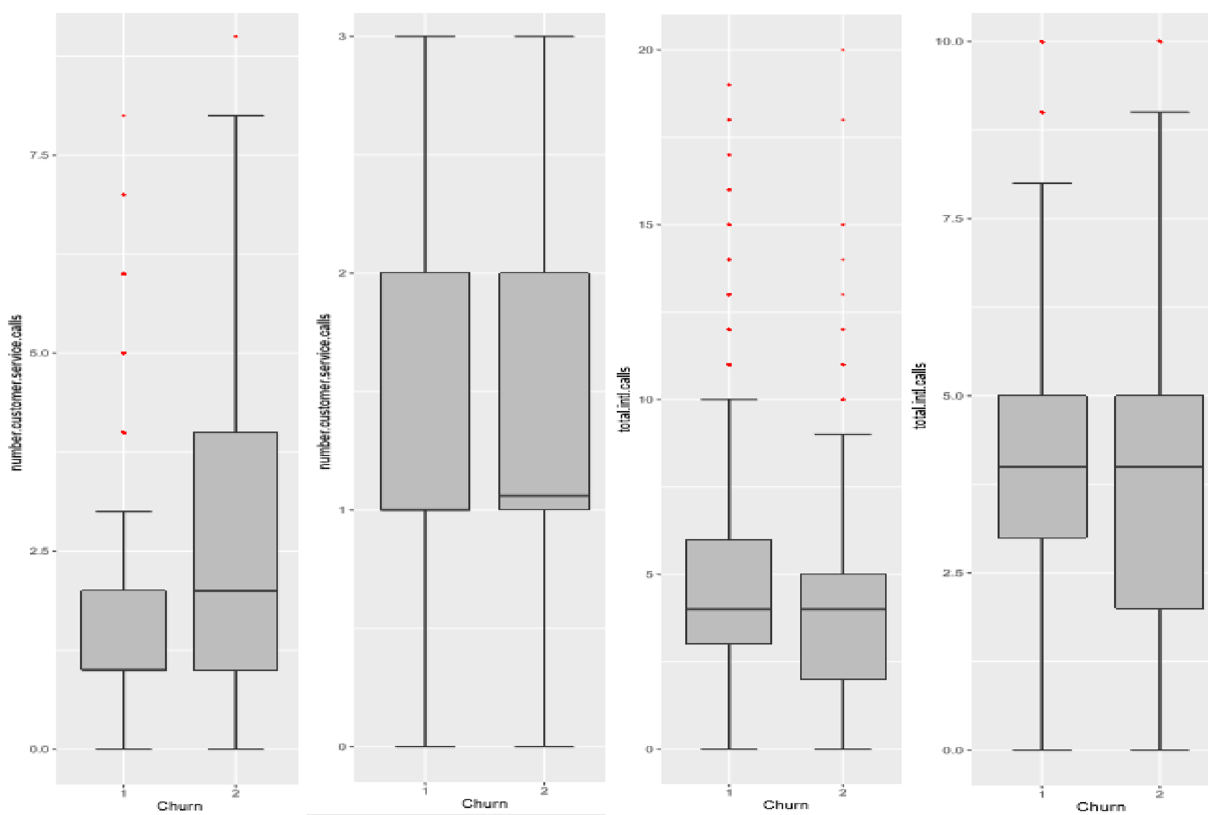


Fig:3 - Box plot of Total Intl calls and Number.Customer.Service Calls Before and After Outlier Analysis.

From the above figures, it is evident that the distribution of data became more normal / less skewed after the removal of outliers. In the box plots of variables , we can see red spots- ‘Outliers’, which were more dominant in numbers before outlier analysis, has become very less and negligible after outlier analysis. This illustrates that removal of outliers from the data can improve the accuracy of the predictive model.

2.1.3 Feature Selection

Feature selection is extremely important in machine learning primarily because it serves as a fundamental technique to direct the use of variables to what's most efficient and effective for a given machine learning system. It helps to minimize the curse of dimensionality or help deal with over fitting feature selection helps to give developers the tools to use only the most relevant and useful data in machine learning training sets, which dramatically reduces costs and data volume. There are many ways to do feature selection, but in this project we use Chi-Square test and Correlation Analysis for the feature selection of Categorical and Continuous variables respectively.

The correlation plot of numerical variables is shown below. From the figures, it is clear that the following variables are highly correlated to each other.

1. Total day minutes and Total day charge
2. Total eve minutes and Total eve charge
3. Total Intl minutes and Total Intl charge
4. Total night minutes and Total night charge

As these variables are highly correlated, we can eliminate the following numerical variables, so that it helps to minimize the curse of dimensionality.

1. Total day charge
2. Total eve charge
3. Total Intl charge
4. Total night charge

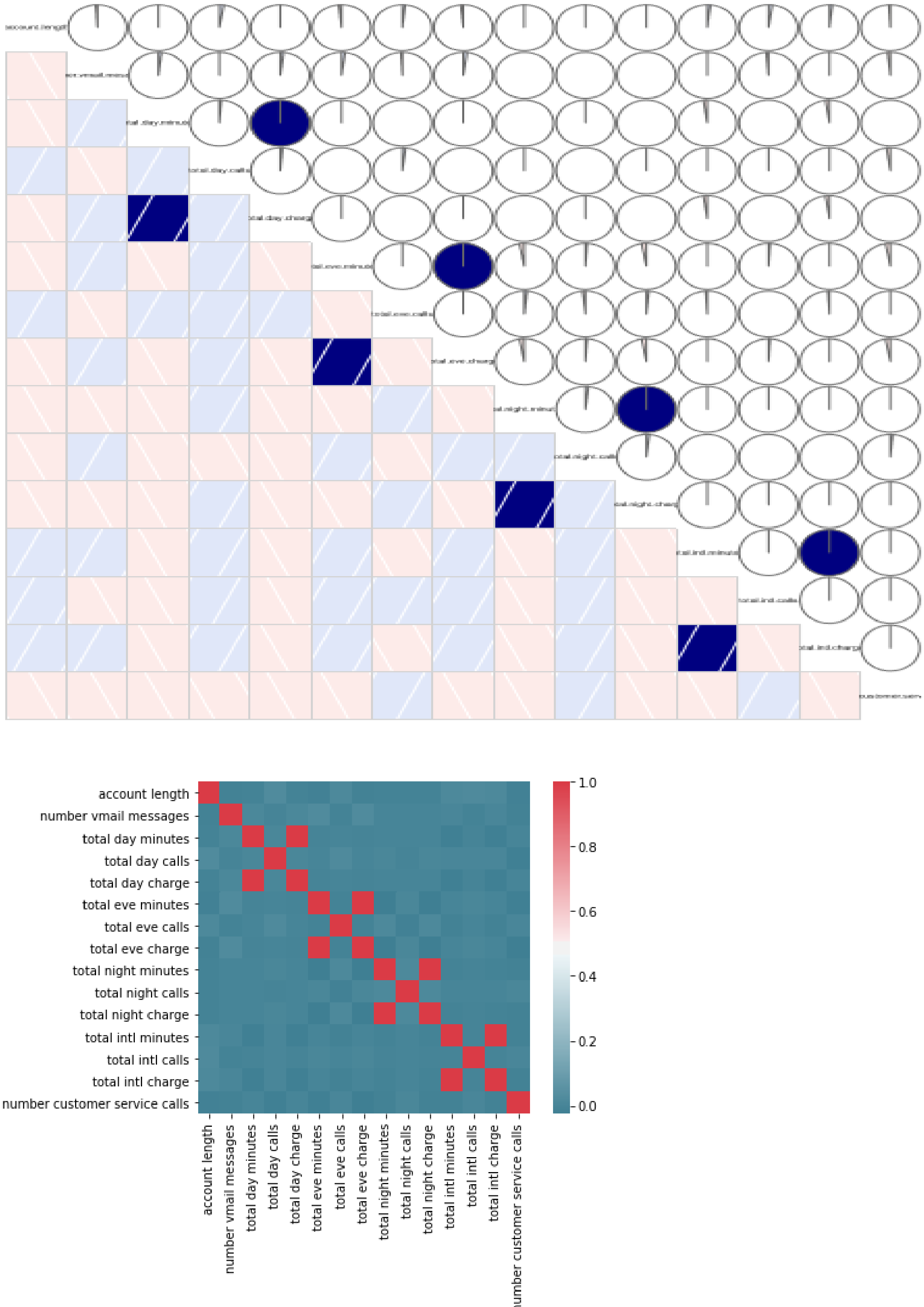


Fig : 7 – Correlation plot used in R & Python programming

When it comes to categorical variables, we need to perform Chi-Square test of Independence in order to extract unwanted variables from the data set. The result of Chi-Square test is shown below.

```
[1] "state"

	Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 96.899, df = 50, p-value = 7.851e-05

[1] "area.code"

	Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 0.56298, df = 2, p-value = 0.7547

[1] "phone.number"

	Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 5000, df = 4999, p-value = 0.4934

[1] "international.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 333.19, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 60.552, df = 1, p-value = 7.165e-15
```

Since the p values of the variables- "area.code" and "phone.number" are greater than 0.05, As per the rule of Chi-Square test, it is found that these variables can be eliminated from modeling stage. Therefore, as part of Dimension reduction, the following variables are removed from the data set.

1. Total day charge
2. Total eve charge
3. Total Intl charge
4. Total night charge

5. Area Code
6. Phone number

2.1.4 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 preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

In this project, we apply feature scaling methods like Normalization and Standardization to the independent variables. Standardization is usually done on the variables whose distribution of data is normal, whereas the Normalization is applied on the variables whose data distribution is not uniform/ Normal. Normality check functions are used for the same. Once we analyze the distribution of data/ Normality check, Standardization and Normalization is executed based on the following formulas.

Normalization Formula:-

$$Value_{new} = \frac{Value - minValue}{maxValue - minValue}$$

Range is 0 to 1

Standardization Formula:-

$$z = \frac{x - \mu}{\sigma}$$

Where x = raw value, σ = standard deviation, μ = mean

Z can be either positive or negative. It will be negative if the raw value is less than mean. On the other hand it will be positive when raw value is greater than mean.

Chapter 3

Modeling

3.1 Model Selection

Model selection is purely based on the type of Machine learning algorithm that we opt. In this case, we follow supervised machine learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

Y = f(X)

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. Supervised learning problems can be further grouped into regression and classification problems.

- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

As the problem statement is about churn reduction and the target variable is Categorical, The predictive model should be a Classification model. We need to follow Trial and Error method in order to find out the most suitable predictive model. The classification models that we used in this project are followings:-

1. Decision tree
2. Random forest
3. Logistic regression
4. KNN implementation
5. Naive Bayes

3.2 Model Evaluation

The evaluation of model can be done by Error metrics. There are two types of error metrics.

1. Classification metrics
   a. Confusion matrix
   b. Accuracy
   c. False positive rate
   d. False negative rate
2. Regression metrics
   a. RMSE
   b. MSE

As we selected Classification model for Predictive Analysis, The error metrics that we opted for model evaluation is Classification metrics. First of all, we need to develop a confusion matrix where both predicted class and actual class are compared. From this confusion matrix, we can derive the followings which may play a crucial role in model evaluation.

1. Accuracy
2. False negative rate
3. False positive rate
4. True positive rate
5. True negative rate

Generally, we consider Accuracy and False negative rate of the model. So it is possible to determine the efficiency of the model from the developed confusion matrix. The results of confusion matrixes of each model that we tried in this predictive analysis are shown below.

I. Decision tree

```
Confusion Matrix and Statistics

   C50_Predictions
       1    2
  1 1441    2
  2   90  134

               Accuracy : 0.9448
                 95% CI : (0.9327, 0.9553)
    No Information Rate : 0.9184
    P-Value [Acc > NIR] : 2.07e-05

                  Kappa : 0.7156
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9412
            Specificity : 0.9853
         Pos Pred Value : 0.9986
         Neg Pred Value : 0.5982
             Prevalence : 0.9184
         Detection Rate : 0.8644
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9633

       'Positive' Class : 1
```

Accuracy: 94.48%

FNR: 40.2%

II. Random Forest

```
Confusion Matrix and Statistics

   RF_Predictions
       1    2
  1 1428   15
  2  116  108

               Accuracy : 0.9214
                 95% CI : (0.9074, 0.9339)
    No Information Rate : 0.9262
    P-Value [Acc > NIR] : 0.7885

                  Kappa : 0.5827
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9249
            Specificity : 0.8780
         Pos Pred Value : 0.9896
         Neg Pred Value : 0.4821
             Prevalence : 0.9262
         Detection Rate : 0.8566
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9015

       'Positive' Class : 1
```

Accuracy=92.3%

FNR=51%

III. Logistic Regression

```
   logit_Predictions
       0    1
  1 1414   29
  2  181   43
```

Accuracy=87.4%

FNR=80.8%

IV. KNN implementation

```
KNN_Predictions    1    2
              1 1436  216
              2    7    8
```

Accuracy = 86.6%

FNR = 46.6%

V. Naïve Bayes

```
Confusion Matrix and Statistics

        predicted
observed    1    2
       1 1416   27
       2  177   47

               Accuracy : 0.8776
                 95% CI : (0.8609, 0.893)
    No Information Rate : 0.9556
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2665
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8889
            Specificity : 0.6351
         Pos Pred Value : 0.9813
         Neg Pred Value : 0.2098
             Prevalence : 0.9556
         Detection Rate : 0.8494
```

```
Detection Prevalence : 0.8656
   Balanced Accuracy : 0.7620

    'Positive' Class : 1
```

Accuracy: 88%
FNR: 79%

From the above results, it is clear that the model based on Decision tree works efficiently with an Accuracy of 95% and False negative rate of 40%.

Some of the rules used for this model creation are mentioned in the Appendix A.

The decision tree flow chart is shown in Appendix C.

Attribute usage:

100.00%	state
100.00%	account.length
100.00%	international.plan
100.00%	voice.mail.plan
100.00%	number.vmail.messages
100.00%	total.day.minutes
100.00%	total.day.calls
100.00%	total.eve.minutes
100.00%	total.night.minutes
100.00%	total.intl.minutes
100.00%	total.intl.calls
100.00%	number.customer.service.calls
99.97%	total.night.calls
99.91%	total.eve.calls

Appendix A

Call:

C5.0.formula(formula = Churn ~ ., data = churn, trials = 100, rules = TRUE)

C5.0 [Release 2.07 GPL Edition]   Fri Jul 20 10:30:43 2018

Class specified by attribute `outcome'

Read 3333 cases (15 attributes) from undefined.data

----- Trial 0: -----

Rules:

Rule 0/1: (160/4, lift 1.1)

international.plan = 1

voice.mail.plan = 2

total.day.minutes > 0.8202674

-> class 1 [0.969]

Rule 0/2: (870/31, lift 1.1)

international.plan = 1

total.day.minutes <= 0.8202674

number.customer.service.calls > 0.6450831

-> class 1 [0.963]

Rule 0/3: (1413/60, lift 1.1)

international.plan = 1

total.day.minutes <= 0.8202674

number.customer.service.calls <= 0.3378751

-> class 1 [0.957]

Rule 0/4: (1045/53, lift 1.1)

international.plan = 1

total.day.minutes > -0.8124499

total.day.minutes <= 0.8202674

total.eve.minutes > -0.212824

-> class 1 [0.948]

Rule 0/5: (2226/171, lift 1.1)

international.plan = 1

total.day.minutes <= 1.607948

total.eve.minutes <= 0.8270235

-> class 1 [0.923]

Rule 0/6: (2162/180, lift 1.1)

total.day.minutes <= 1.607948

total.intl.minutes <= 1.100018

total.intl.calls > -0.6341546

-> class 1 [0.916]

Rule 0/7: (782/88, lift 1.0)

total.eve.minutes <= -0.2354293

total.night.minutes <= 0.251038

-> class 1 [0.886]

Rule 0/8: (62, lift 6.8)

international.plan = 2

total.intl.calls <= -0.6341546

-> class 2 [0.984]

Rule 0/9: (52, lift 6.8)

international.plan = 2

total.intl.minutes > 1.100018

-> class 2 [0.981]

Rule 0/10: (47, lift 6.8)

total.day.minutes <= -0.1146258

total.eve.minutes <= -0.212824

number.customer.service.calls > 0.3378751

number.customer.service.calls <= 0.6450831

-> class 2 [0.980]

Rule 0/11: (43/1, lift 6.6)

total.day.minutes <= -0.8124499

number.customer.service.calls > 0.3378751

number.customer.service.calls <= 0.6450831

-> class 2 [0.956]

Rule 0/12: (60/3, lift 6.5)

voice.mail.plan = 1

total.day.minutes > 1.607948

total.eve.minutes > -1.562982

total.night.minutes > 0.251038

-> class 2 [0.935]

Rule 0/13: (92/6, lift 6.4)

voice.mail.plan = 1

total.day.minutes > 1.607948

total.eve.minutes > -0.2354293

-> class 2 [0.926]

Rule 0/14: (9, lift 6.3)

voice.mail.plan = 1

total.day.minutes > 2.38607

total.night.minutes <= 0.251038

-> class 2 [0.909]

Rule 0/15: (19/1, lift 6.2)

total.day.minutes <= 0.8202674

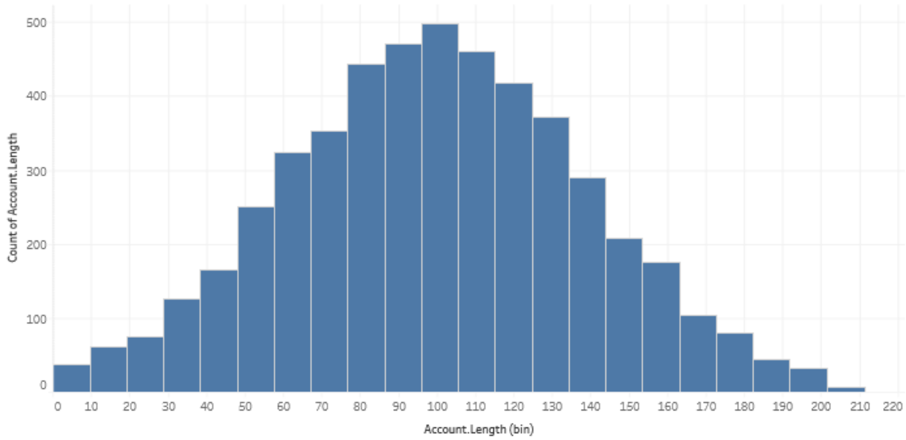
total.eve.minutes <= -1.351313

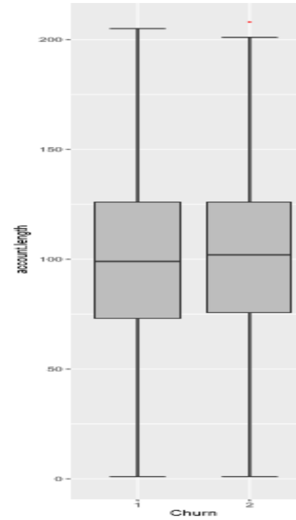
number.customer.service.calls > 0.3378751

number.customer.service.calls <= 0.6450831

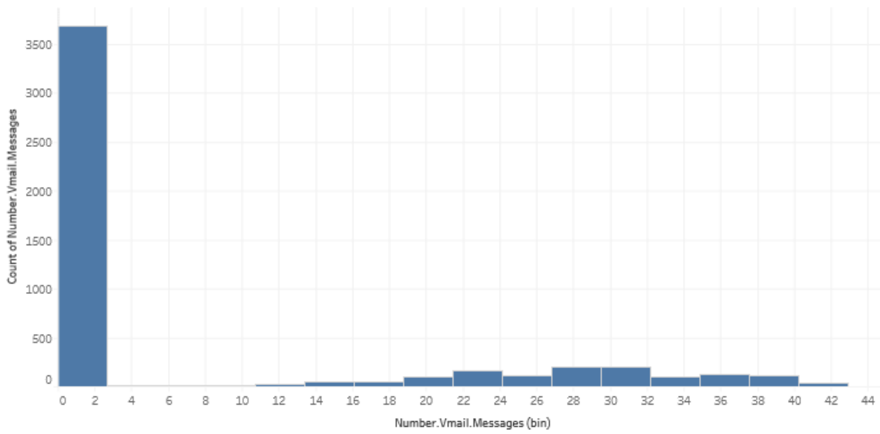
-> class 2 [0.905]

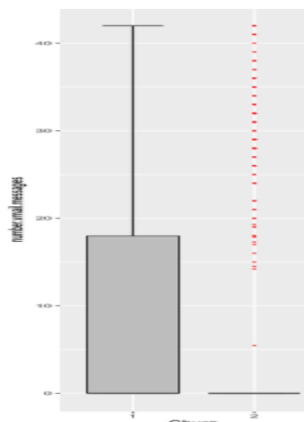
Appendix B



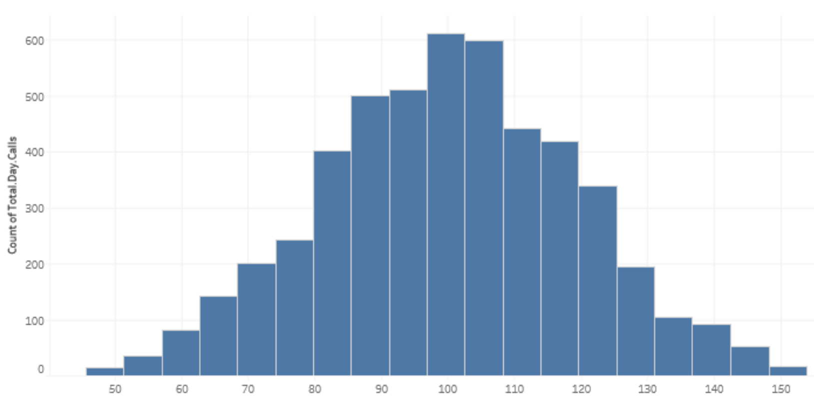


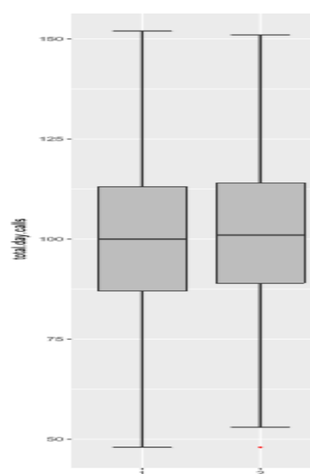
The trend of count of Account.Length for Account.Length (bin).



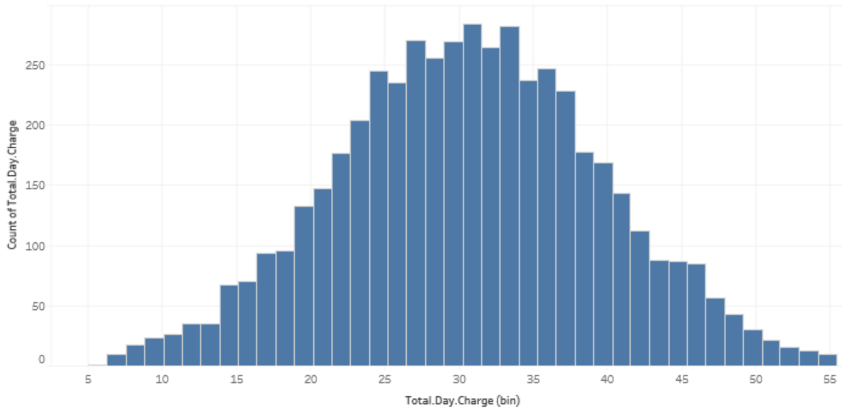


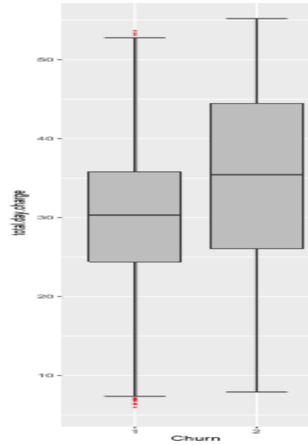
The trend of count of Number.Vmail.Messages for Number.Vmail.Messages (bin).



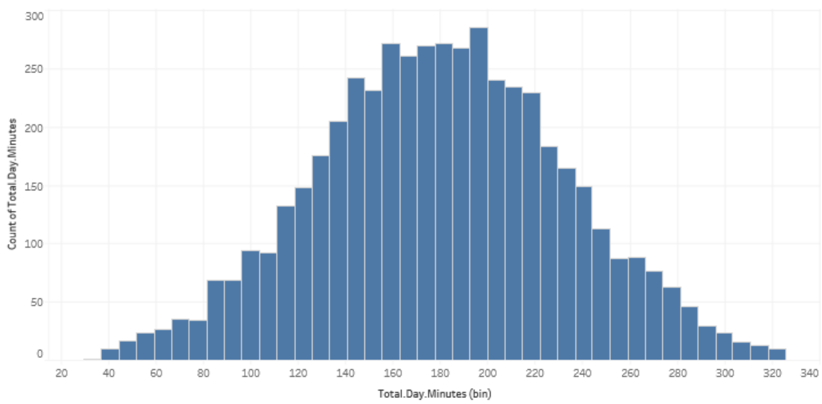


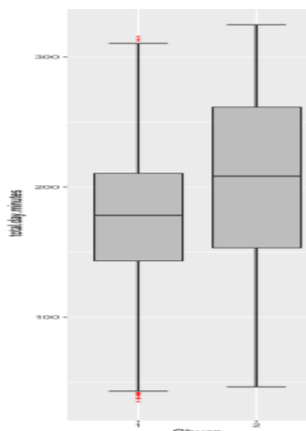
The trend of count of Total.Day.Calls for Total.Day.Calls (bin).



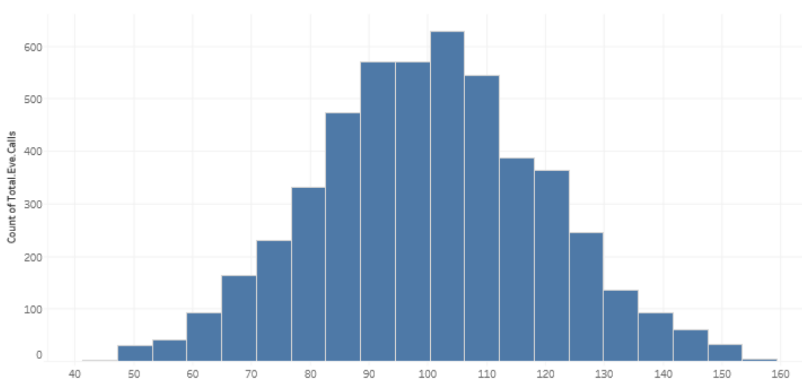


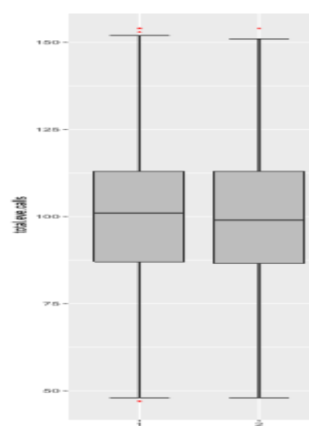
The trend of count of Total.Day.Charge for Total.Day.Charge (bin).



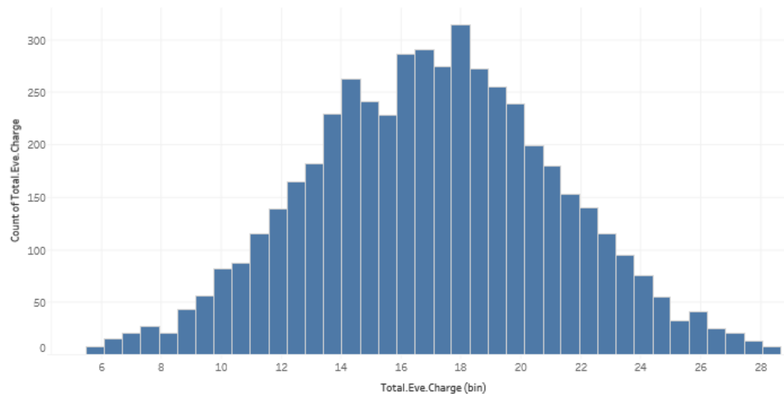


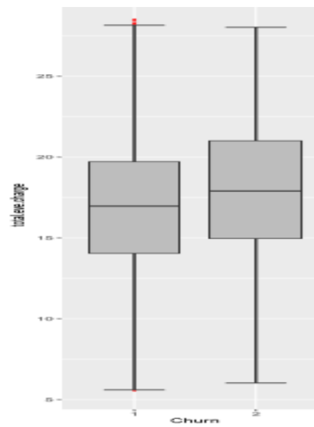
The trend of count of Total.Day.Minutes for Total.Day.Minutes (bin).



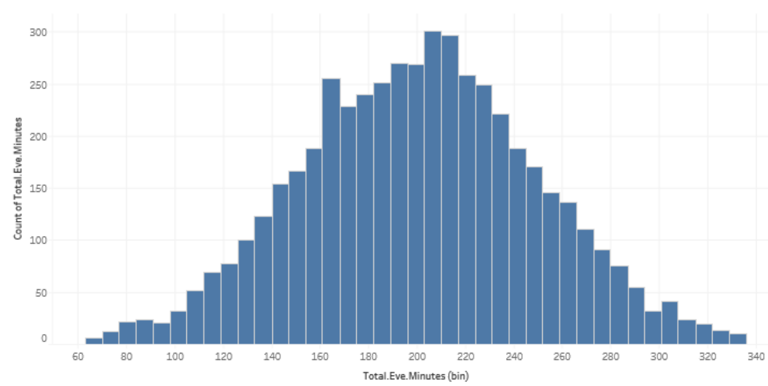


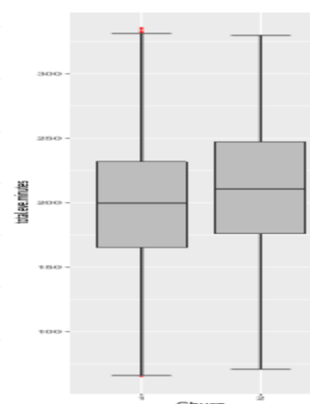
The trend of count of Total.Eve.Calls for Total.Eve.Calls (bin).



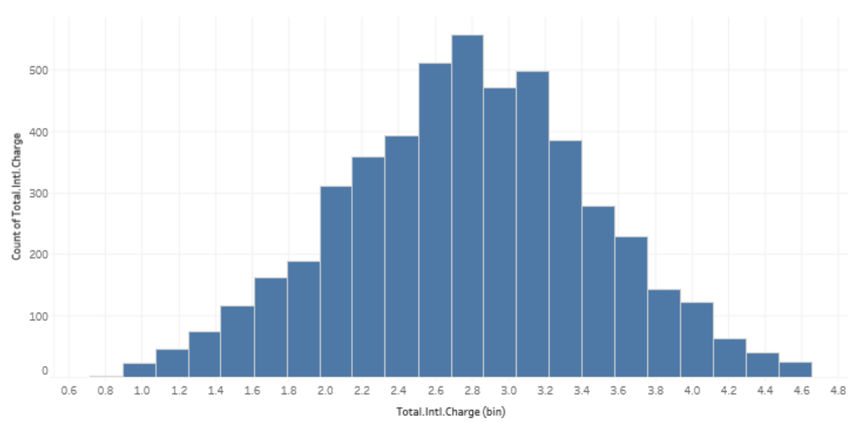


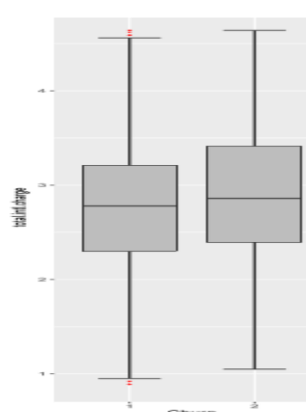
'he trend of count of Total.Eve.Charge for Total.Eve.Charge (bin).



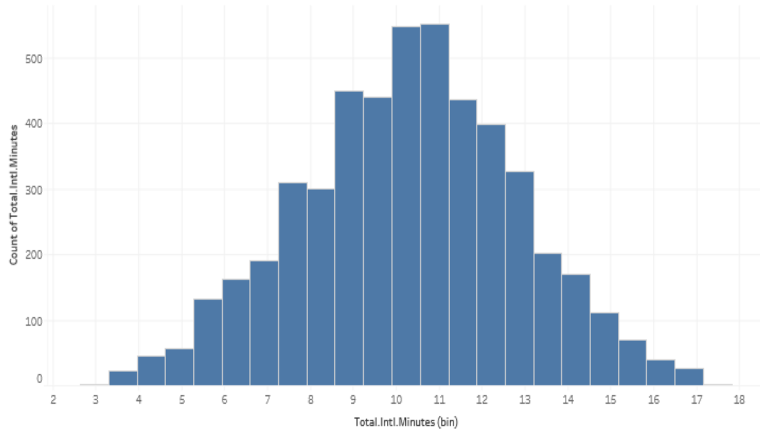


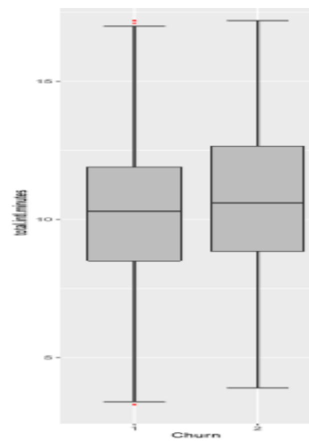
The trend of count of Total.Eve.Minutes for Total.Eve.Minutes (bin).



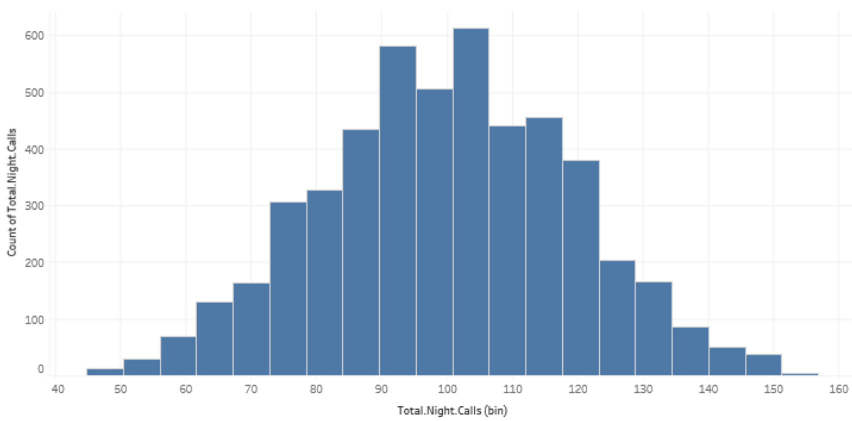


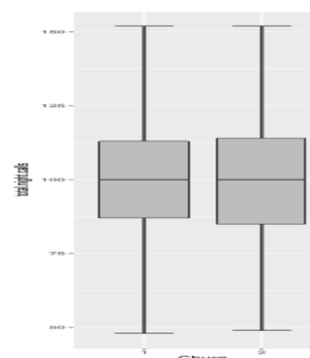
The trend of count of Total.Intl.Charge for Total.Intl.Charge (bin).



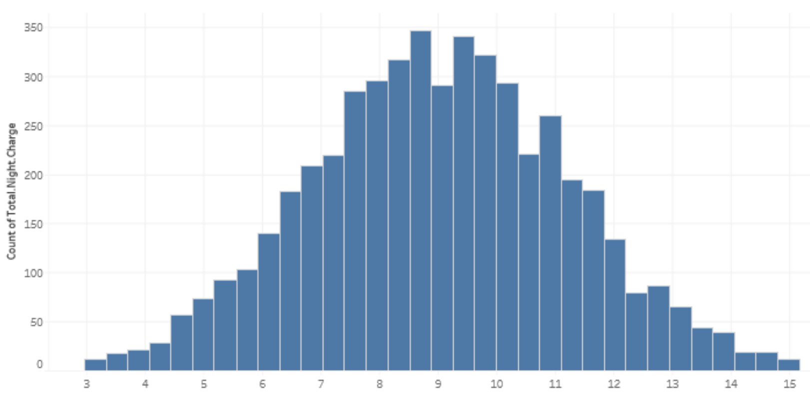


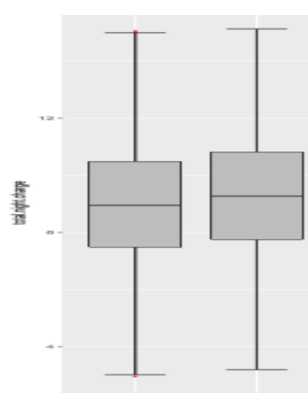
The trend of count of Total.Intl.Minutes for Total.Intl.Minutes (bin).



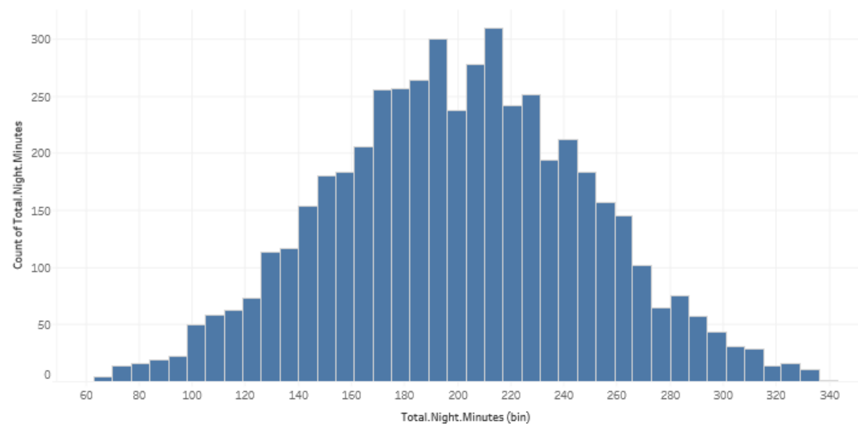


The trend of count of Total.Night.Calls for Total.Night.Calls (bin).

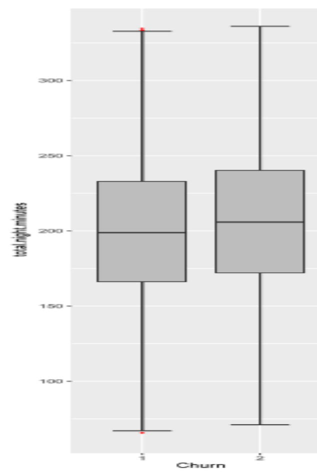




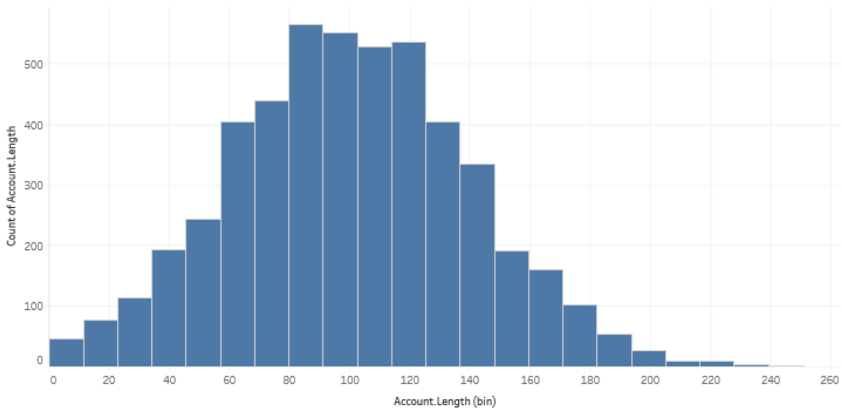
The trend of count of Total.Night.Charge for Total.Night.Charge (bin).

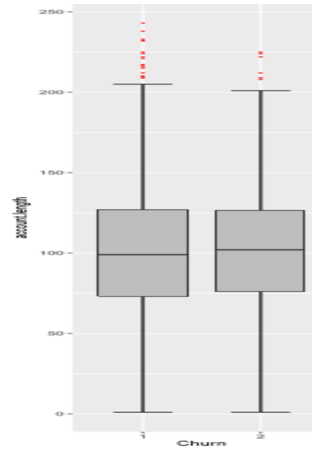


The trend of count of Total.Night.Minutes for Total.Night.Minutes (bin).

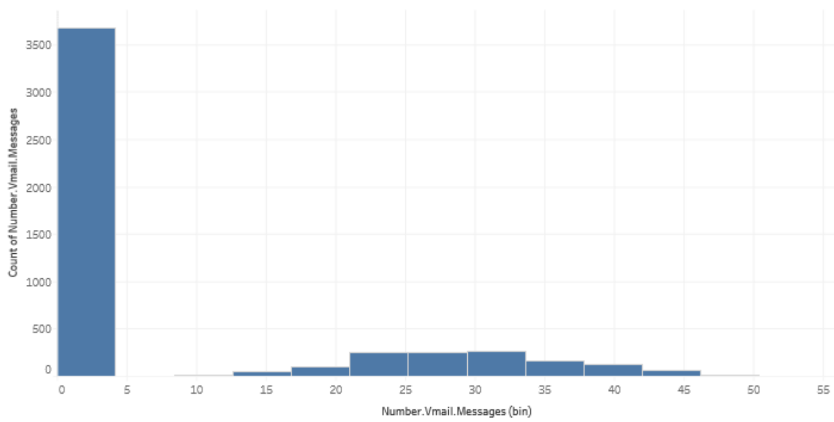


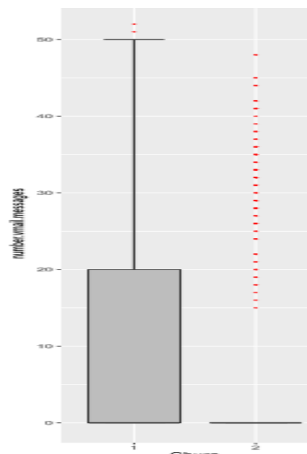
Appendix C



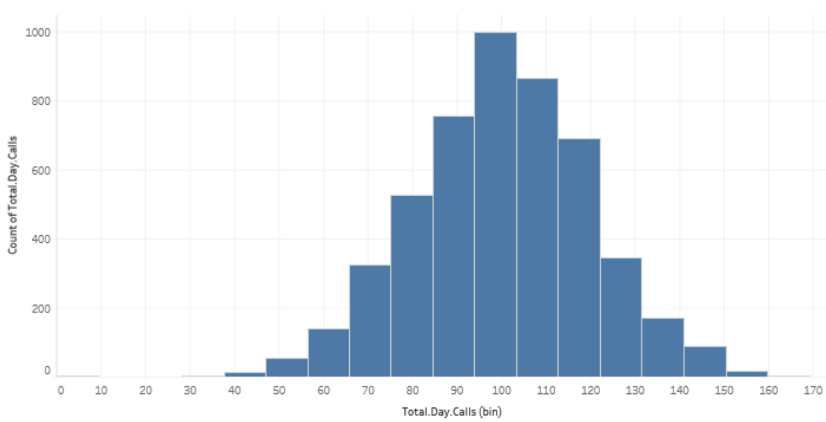


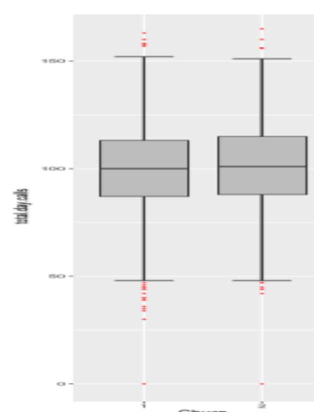
The trend of count of Account.Length for Account.Length (bin).



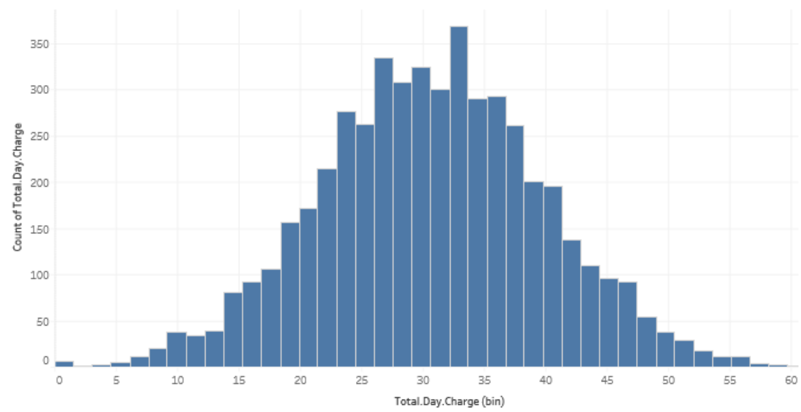


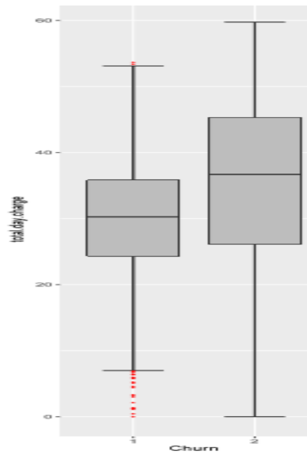
The trend of count of Number.Vmail.Messages for Number.Vmail.Messages (bin).



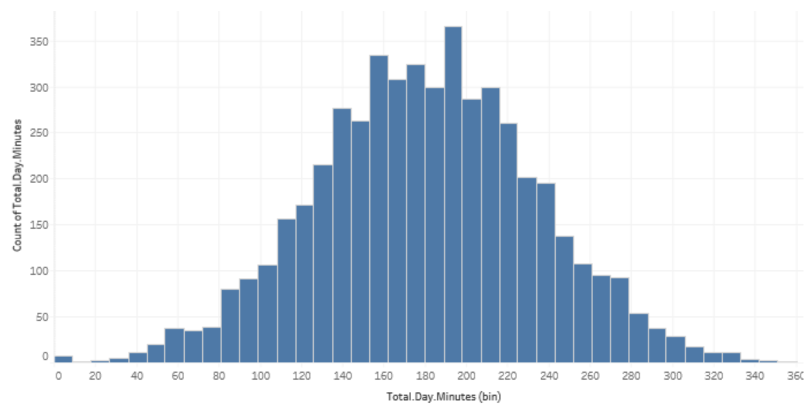


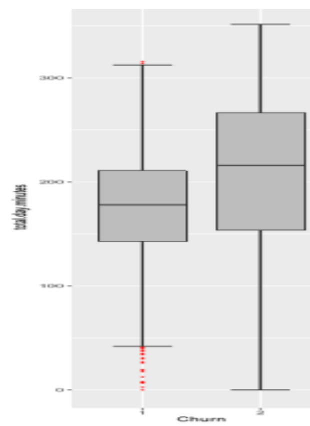
The trend of count of Total.Day.Calls for Total.Day.Calls (bin).



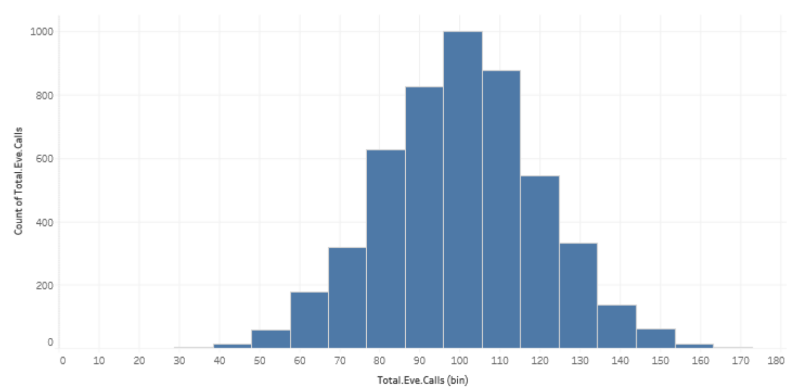


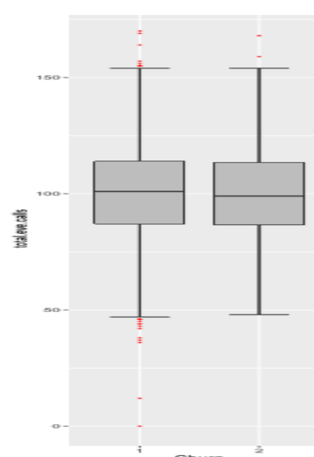
The trend of count of Total.Day.Charge for Total.Day.Charge (bin).



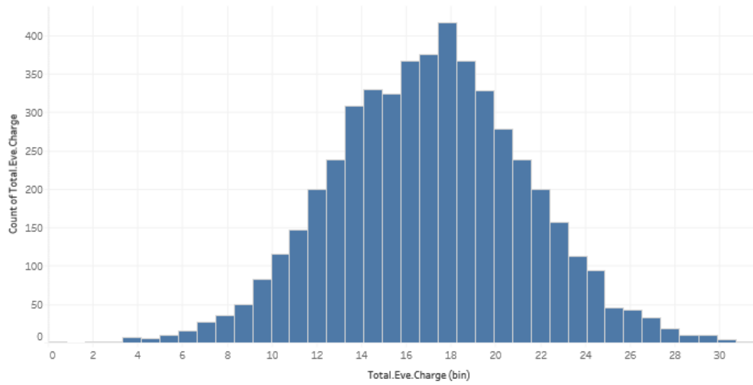


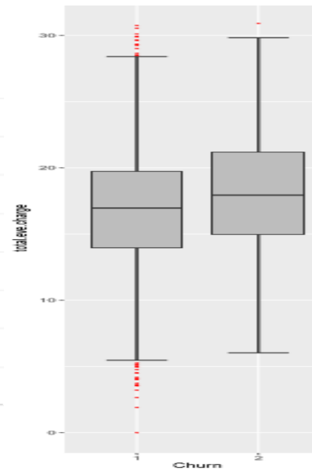
The trend of count of Total.Day.Minutes for Total.Day.Minutes (bin).



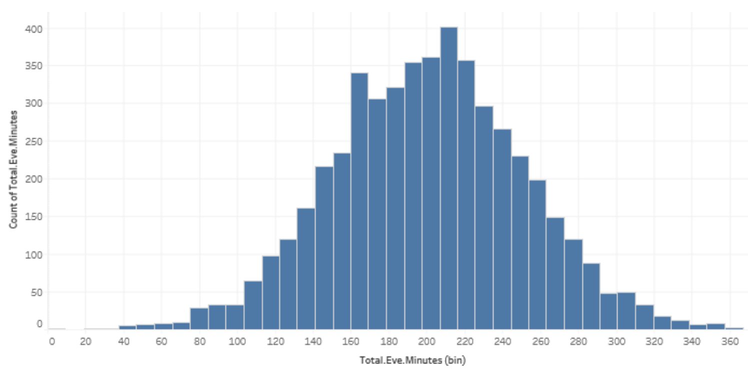


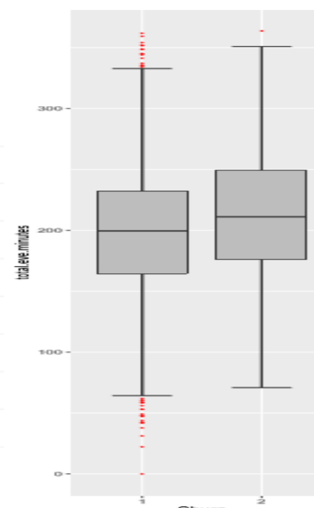
The trend of count of Total.Eve.Calls for Total.Eve.Calls (bin).



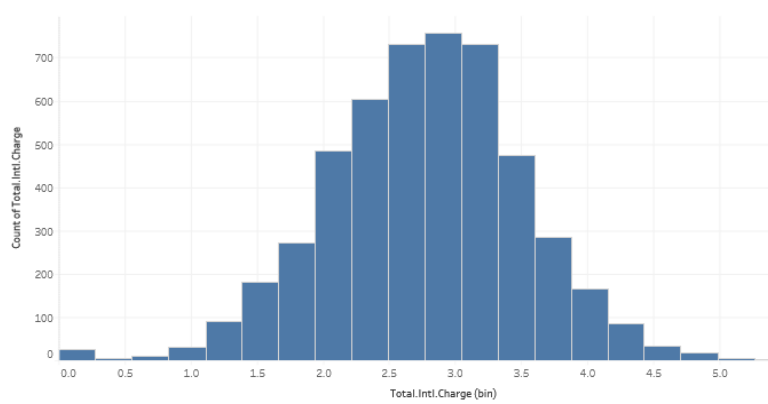


The trend of count of Total.Eve.Charge for Total.Eve.Charge (bin).

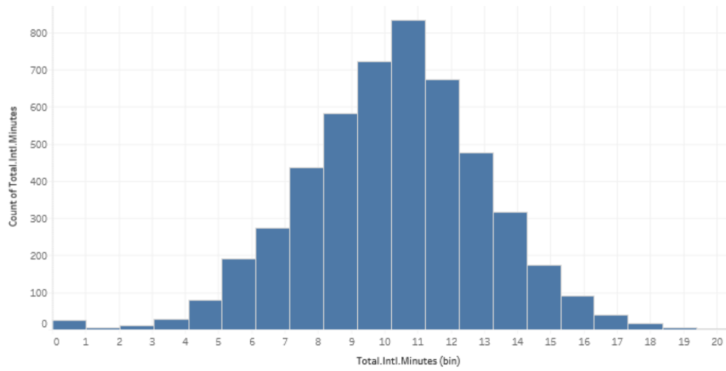


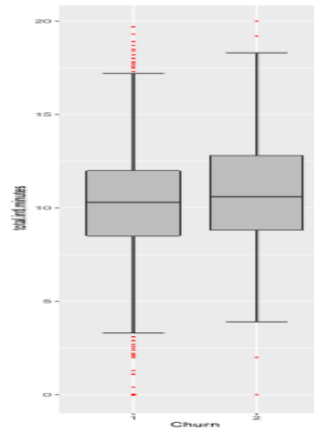


The trend of count of Total.Eve.Minutes for Total.Eve.Minutes (bin).

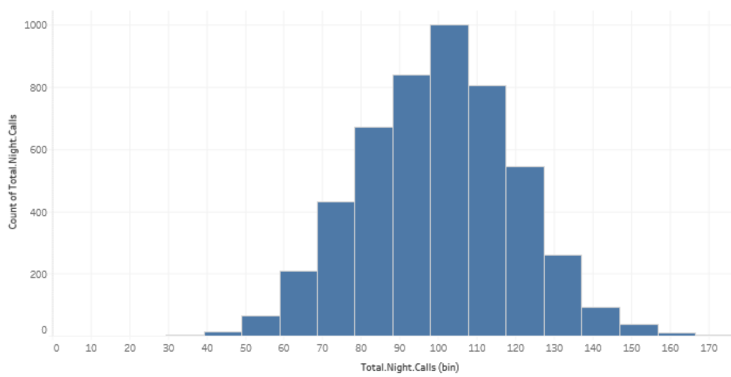


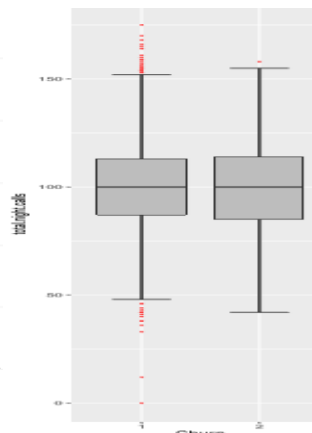
The trend of count of Total.Intl.Charge for Total.Intl.Charge (bin).



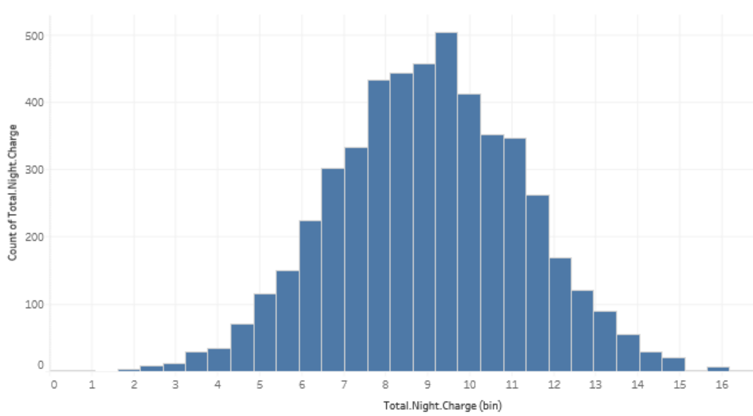


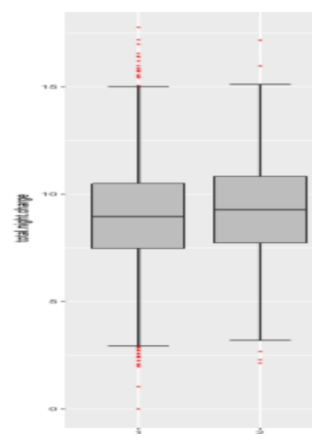
The trend of count of Total.Intl.Minutes for Total.Intl.Minutes (bin).



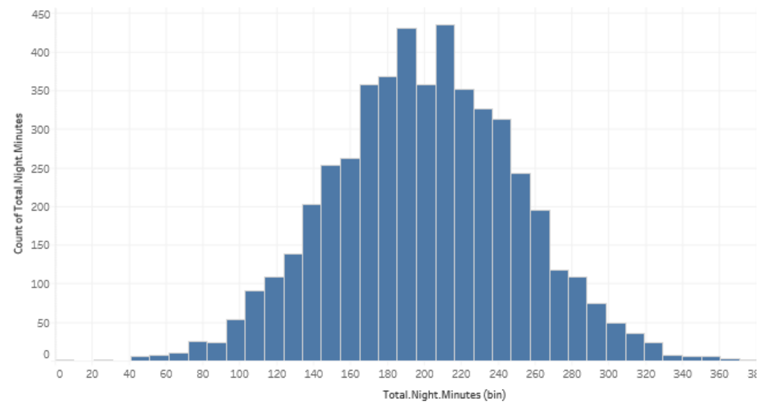


The trend of count of Total.Night.Calls for Total.Night.Calls (bin).

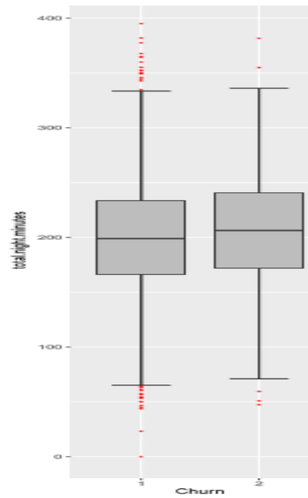




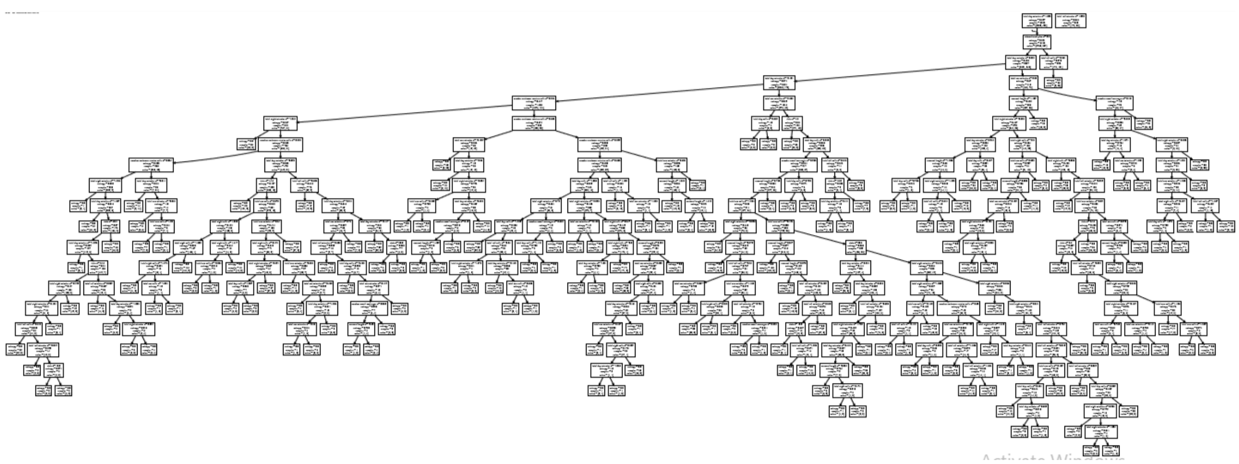
The trend of count of Total.Night.Charge for Total.Night.Charge (bin).



The trend of count of Total.Night.Minutes for Total.Night.Minutes (bin).



Decision Tree Flow Chart



Appendix D

R Code:-

```
rm(list=ls(all=T))

setwd("E:/project")


#Load Libraries

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

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

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

rm(x)


## Read the data

churn = read.csv("train_data.csv", header = T, na.strings = c(" ", "", "NA"))

churn_test=read.csv('test_data.csv',header = T, na.strings = c(" ", "", "NA"))

df=rbind(churn,churn_test)

View(df)

churn=df

str(churn)

churn$area.code=as.factor(churn$area.code)
```

```
###################################Missing Values Analysis#################################################

missing_val = data.frame(apply(churn,2,function(x){sum(is.na(x))}))

missing_val



##Data Manupulation; convert string categories into factor numeric

for(i in 1:ncol(churn)){


  if(class(churn[,i]) == 'factor'){


    churn[,i] = factor(churn[,i], labels=(1:length(levels(factor(churn[,i])))))


  }

}


##############################################Outlier Analysis###############################################

# ## BoxPlots - Distribution and Outlier Check


numeric_index = sapply(churn,is.numeric) #selecting only numeric


numeric_data = churn[,numeric_index]
```

```
cnames = colnames(numeric_data)

cnames

for (i in 1:length(cnames))
  {
    assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"), data =
subset(churn))+
           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])))
  }

### Plotting plots together
 gridExtra::grid.arrange(gn1,gn5,gn2,ncol=3)
 gridExtra::grid.arrange(gn6,gn7,gn3,ncol=3)
 gridExtra::grid.arrange(gn8,gn9,gn4,ncol=3)
 gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
```

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

 View(churn)

 sum(is.na(churn))
# #Replace all outliers with NA and impute

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

 churn = knnImputation(churn, k = 5)

write.csv(df,'merged_with_outliers.csv', row.names = F)

####################################Feature Selection##################################################
## Correlation Plot
corrgram(churn[,numeric_index], order = F,
```

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



## Chi-squared Test of Independence

factor_index = sapply(churn,is.factor)

factor_data = churn[,factor_index]

View(factor_data)

for (i in 1:5)

{

  print(names(factor_data)[i])

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

}


## Dimension Reduction

churn = subset(churn,

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


###################################Feature
Scaling#################################################

#Normality check


qqnorm(churn$number.customer.service.calls)
```

```
hist(churn$number.customer.service.calls)

#Normalisation

cnames1= c("number.vmail.messages","number.customer.service.calls")


for(i in cnames1){

  churn[,i] = (churn[,i] - min(churn[,i]))/

    (max(churn[,i] - min(churn[,i])))

}



 #Standardisation


cnames2=c("account.length","total.day.minutes","total.day.calls",

          "total.intl.minutes","total.intl.calls","total.eve.minutes",

          "total.eve.calls","total.night.minutes","total.night.calls")


for(i in cnames2){

  churn[,i] = (churn[,i] - mean(churn[,i]))/

                                  sd(churn[,i])

}
```

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

#######Import test data #######################

## Read the data

churn_test=churn[3334:5000,]

churn=churn[1:3333,]

View(churn_test)


###################################Model
Development#######################################

#Clean the environment

rmExcept(c("churn","churn_test"))

#Decision tree for classification

#Develop Model on training data

C50_model = C5.0(Churn ~.,churn,trials=100,rules = TRUE)


#Summary of DT model

summary(C50_model)


#write rules into disk

write(capture.output(summary(C50_model)), "c50mergedrules.txt")
```

```
#Lets predict for test cases
C50_Predictions = predict(C50_model, churn_test[,-15], type = "class")


##Evaluate the performance of classification model
ConfMatrix_C50 = table(churn_test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)


#Accuracy: 94.48%
#FNR: 40.2%
##################################################################################
###Random Forest
RF_model = randomForest(Churn ~ ., churn, importance = TRUE, ntree = 500)


#Presdict test data using random forest model
RF_Predictions = predict(RF_model, churn_test[,-15])


##Evaluate the performance of classification model
ConfMatrix_RF = table(churn_test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
#Accuracy=92.3%
#FNR=51%
```

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

#Logistic Regression

logit_model = glm(Churn ~ ., data = churn, family = "binomial")


#summary of the model

summary(logit_model)


#predict using logistic regression

logit_Predictions = predict(logit_model, newdata = churn_test, type = "response")


#convert prob

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



##Evaluate the performance of classification model

ConfMatrix_RF = table(churn_test$Churn, logit_Predictions)

ConfMatrix_RF

#accuracy=87.4%

#False Negative rate=80.8%

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

```
##KNN Implementation
library(class)

#Predict test data
KNN_Predictions = knn(churn[, 1:14], churn_test[, 1:14], churn$Churn, k = 7)

#Confusion matrix
Conf_matrix = table(KNN_Predictions, churn_test$Churn)
Conf_matrix
#Accuracy
sum(diag(Conf_matrix))/nrow(churn_test)

#Accuracy = 86.6%
#FNR = 46.6%
#####################################################################
##############

#naive Bayes
library(e1071)

#Develop model
NB_model = naiveBayes(Churn ~ ., data = churn)
```

```
#predict on test cases #raw
NB_Predictions = predict(NB_model, churn_test[,1:14], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = churn_test[,15], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)

#Accuracy: 88%
#FNR: 79%
```

Python code:

```
#Load libraries
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
#Set working directory
```

```
os.chdir("E:\projectpy")
#Load data
churn = pd.read_csv("train_data.csv")
churn_test=pd.read_csv("test_data.csv")
#combine two datasets
df=churn.append(churn_test,ignore_index=True)
df['area code']=df['area code'].astype(object)
#Assigning levels to the categories
lis = []
for i in range(0, df.shape[1]):
    if(df.iloc[:,i].dtypes == 'object'):
        df.iloc[:,i] = pd.Categorical(df.iloc[:,i])
        df.iloc[:,i] = df.iloc[:,i].cat.codes
        df.iloc[:,i] = df.iloc[:,i].astype('object')
        lis.append(df.columns[i]
#to find any missing values
missing_val = pd.DataFrame(df.isnull().sum())
lis1=[]
for i in range(0, df.shape[1]):
    if(df.iloc[:,i].dtypes != 'object'):
        lis1.append(df.columns[i])
#Detect and replace with NA
for i in lis1:
```

```
    # #Extract quartiles
    q75, q25 = np.percentile(df.loc[:,i], [75,25])

    # #Calculate IQR
    iqr = q75 - q25

    # #Calculate inner and outer fence
    minimum = q25 - (iqr*1.5)
    maximum = q75 + (iqr*1.5)

    # #Replace with NA
    df.loc[df.loc[:,i] < minimum,i] = np.nan
    df.loc[df.loc[:,i] > maximum,i] = np.nan
# #Calculate missing value
missing_val = pd.DataFrame(df.isnull().sum())
missing_val.loc[:,0].sum()
# #Impute with KNN
df = pd.DataFrame(KNN(k = 3).complete(df), columns = df.columns)
##Correlation analysis
#Correlation plot
df_corr = df.loc[:,lis1]
#Set the width and hieght of the plot
```

```
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
#Chisquare test of independence
#Save categorical variables
cat_names = lis[0:5]
cat_names
#loop for chi square values
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(df['Churn'], df[i]))
    print(p)
#dimension reduction
df = df.drop(['area code', 'phone number', 'total day charge', 'total eve charge', 'total intl charge','total night charge'], axis=1)
df2 = df.copy()
#Normality check
```

```
%matplotlib inline

plt.hist(df['number customer service calls'], bins='auto')

#Nomalisation

cnames=["number customer service calls","number vmail messages"]

for i in cnames:

    print(i)

    df[i] = (df[i] - min(df[i]))/(max(df[i]) - min(df[i]))

# #Standarisation

cnames1=['account length',

 'total day minutes',

 'total day calls',

 'total eve minutes',

 'total eve calls',

 'total night minutes',

 'total night calls',

 'total intl minutes',

 'total intl calls']

for i in cnames1:

    print(i)

    df[i] = (df[i] - df[i].mean())/df[i].std()

#replace target categories with Yes or No

df['Churn'] = df['Churn'].replace(0.0, 'No')

df['Churn'] = df['Churn'].replace(1.0, 'Yes')
```

```
train=df.iloc[0:3333,:]

test=df.iloc[3333:,:]

X_train=train.iloc[:,0:14]

y_train=train.iloc[:,14]

X_test=test.iloc[:,0:14]

y_test=test.iloc[:,14]

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

#Import Libraries for decision tree

from sklearn import tree

from sklearn.metrics import accuracy_score

from sklearn.cross_validation import train_test_split

#Decision Tree

C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)


#predict new test cases

C50_Predictions = C50_model.predict(X_test)


#Create dot file to visualise tree  #http://webgraphviz.com/

# dotfile = open("pt.dot", 'w')

# df = tree.export_graphviz(C50_model, out_file=dotfile, feature_names =
marketing_train.columns)

#build confusion matrix

# from sklearn.metrics import confusion_matrix
```

```
# CM = confusion_matrix(y_test, y_pred)
CM = pd.crosstab(y_test, C50_Predictions)

#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]

#check accuracy of model
#accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
#(FN*100)/(FN+TP)

#Results
#Accuracy: 89.5%
#FNR: 40%
#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF_model = RandomForestClassifier(n_estimators = 500).fit(X_train, y_train)
```

```
RF_Predictions = RF_model.predict(X_test)
#build confusion matrix
# from sklearn.metrics import confusion_matrix
# CM = confusion_matrix(y_test, y_pred)
CM = pd.crosstab(y_test, RF_Predictions)

#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]

#check accuracy of model
#accuracy_score(y_test, y_pred)*100
#((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
(FN*100)/(FN+TP)

#Accuracy: 92.6
#FNR: 53.6
#KNN implementation
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(X_train, y_train)
#predict test cases
KNN_Predictions = KNN_model.predict(X_test)
#build confusion matrix
CM = pd.crosstab(y_test, KNN_Predictions)

#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]

#check accuracy of model
#accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
(FN*100)/(FN+TP)

#Accuracy: 86.8
#FNR: 96.8
#Naive Bayes
```

```
from sklearn.naive_bayes import GaussianNB

#Naive Bayes implementation
NB_model = GaussianNB().fit(X_train, y_train)
#predict test cases
NB_Predictions = NB_model.predict(X_test)
#Build confusion matrix
CM = pd.crosstab(y_test, NB_Predictions)

#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]

#check accuracy of model
#accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)

#False Negative rate
#(FN*100)/(FN+TP)
#Accuracy: 85.8
#FNR: 68.3
```

THANK YOU