Customer Churn Prediction

Raunak Agrawal

14 November 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this Case is to predict customer behaviour in the telecom industry. We will be predicting the churn of a customer based on certain parameters.

1.2 Data

Our task is to build classification models which will classify the churn depending on multiple factors. Given below is a sample of the data set that we are using to predict the churn:

Telecom user Sample Data (Columns: 1-21)

account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
128	no	yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10	3	2.7	1	False.
107	no	yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7	1	False.
137	no	no	0	243.4	114	41.38	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29	0	False.
84	yes	no	0	299.4	71	50.9	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False.
75	yes	no	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False.
118	yes	no	0	223.4	98	37.98	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7	0	False.
121	no	yes	24	218.2	88	37.09	348.5	108	29.62	212.6	118	9.57	7.5	7	2.03	3	False.

As you can see in the table below we have the following 17 variables, using which we have to correctly predict the churn:
Predictor Variables

1. account length
2. international plan
3. voicemail plan
4. number of voicemail messages
5. total day minutes used
6. day calls made
7. total day charge
8. total evening minutes
9. total evening calls
10. total evening charge
11. total night minutes
12. total night calls
13. total night charge
14. total international minutes used
15. total international calls made
16. total international charge
17. number of customer service calls made

Target Variable - churn: if the customer has moved

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process we will first try and look at all the probability distributions of the variables. Most analysis, require the data to be normally distributed. We can visualize that in a glance by looking at the histograms of the variable.

Variable Reduction

First we remove variables like **State, Area Code** and **Phone Number** because we are predicting Churn based on usage pattern.

Assign Levels to Factor/Categorical Variables

In the given dataset we have 3 Categorical Variables:
1) International Plan
2) Voice Mail Plan
3) Churn

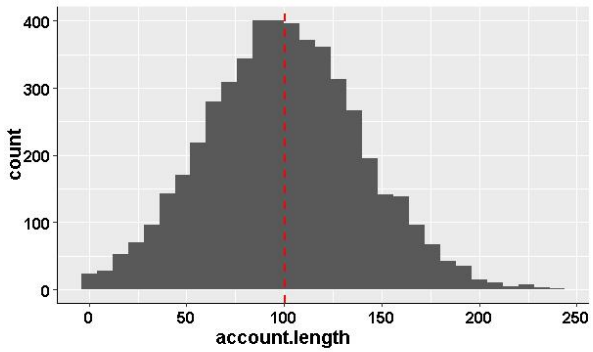
We have assigned 0 and 1 to these variables explained as below:
1) International Plan
No = 0 Yes = 1

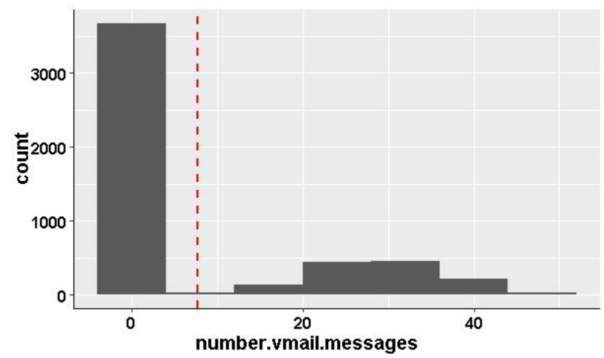
2) Voice Mail Plan
No = 0 Yes = 1

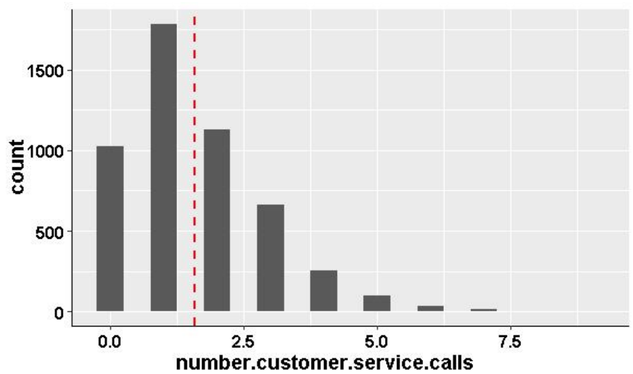
3) Churn
False = 0 True = 1

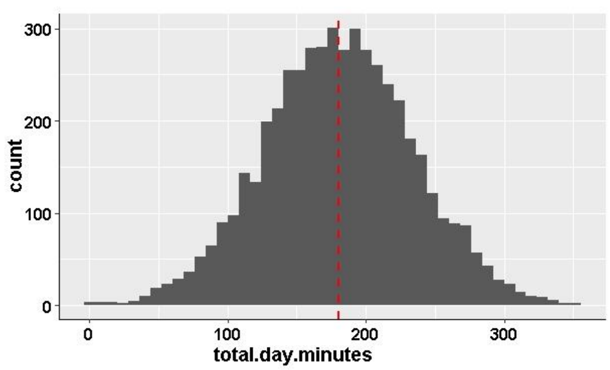
Univariate Analysis

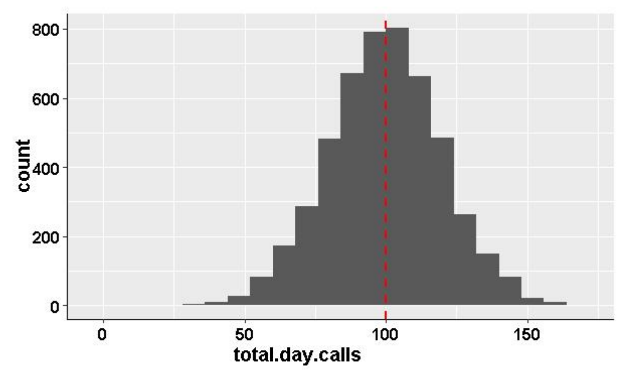
Continuous Variables:- In case of continuous variables, we need to understand the central tendency and spread of the variable. These are measured using histogram Plots as shown below:

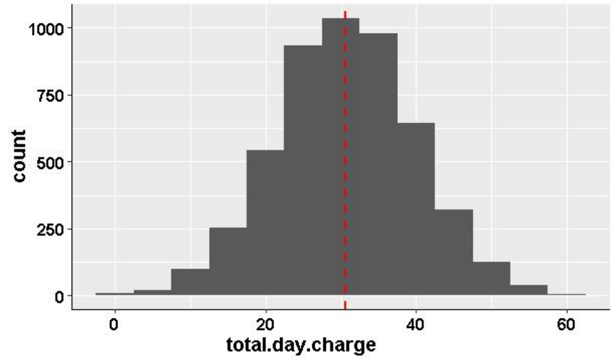

count
account.length

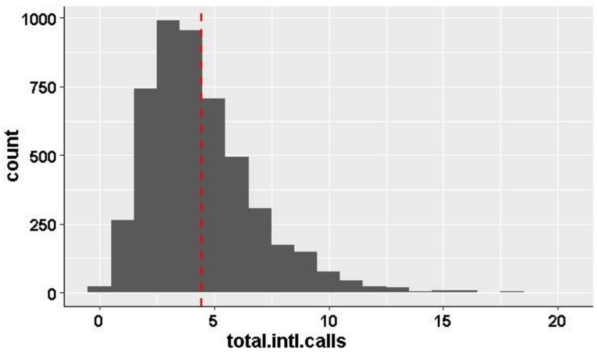

count
number.vmail.messages

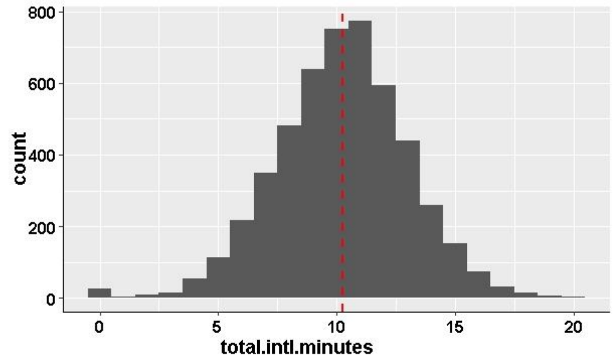

count
number.customer.service.calls

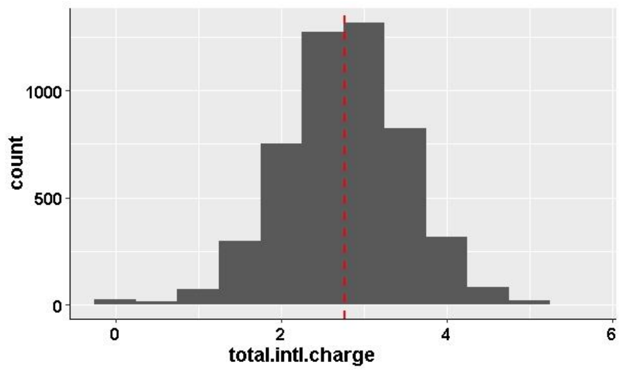

count
total.day.minutes

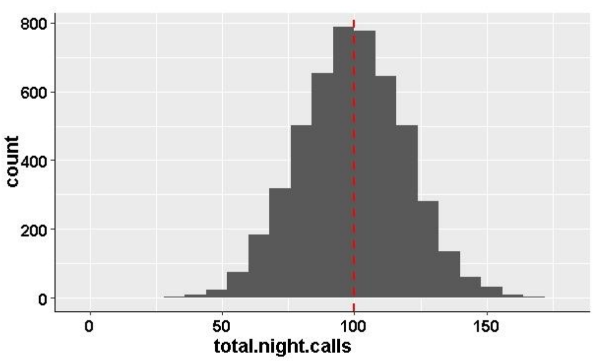

count
total.day.calls

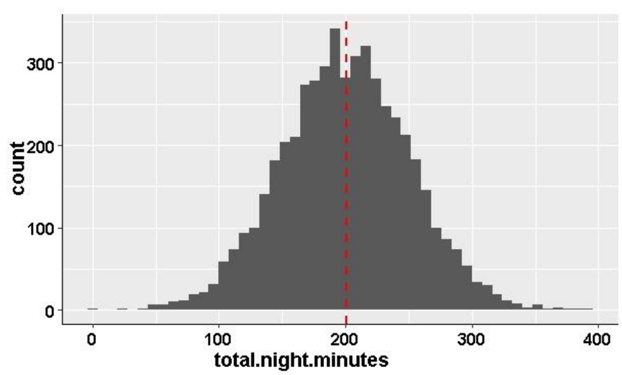

count
total.day.charge

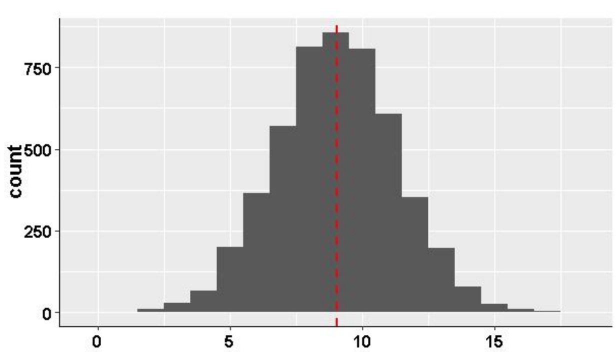

count
total.intl.calls

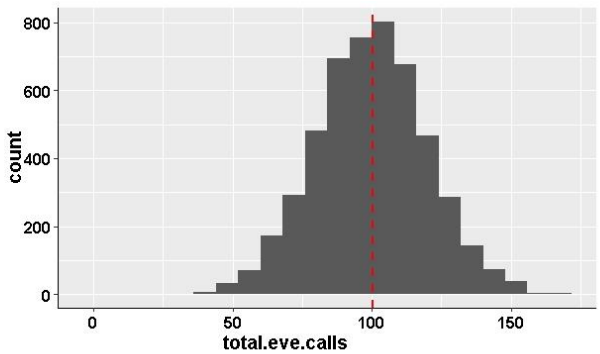

count
total.intl.minutes

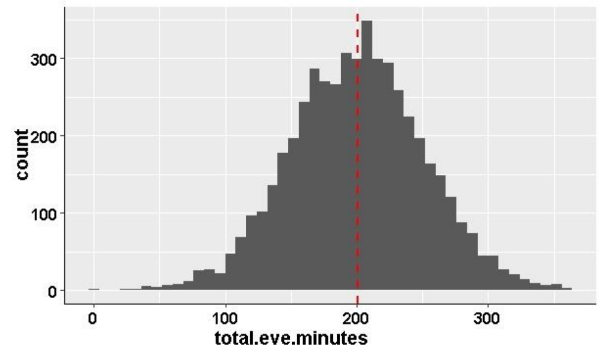

count
total.intl.charge

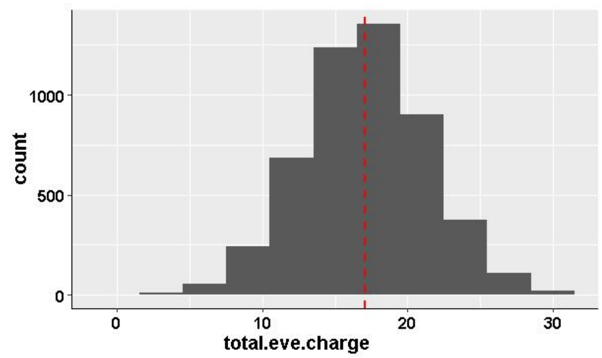

count
total.night.calls


count
total.night.minutes


count
total.night.charge







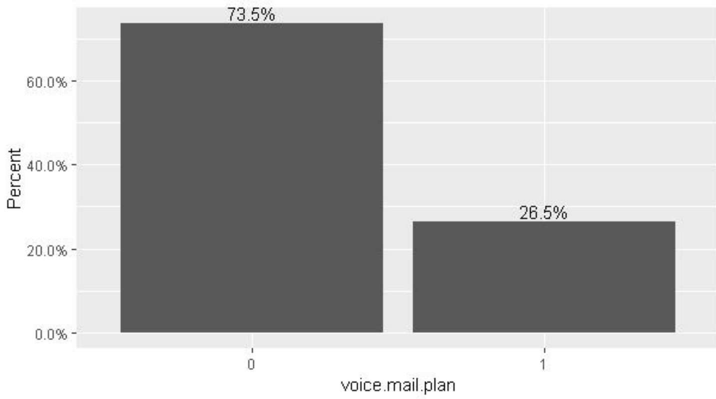
From the above Graphs it was observed that all variables are normally distributed except:

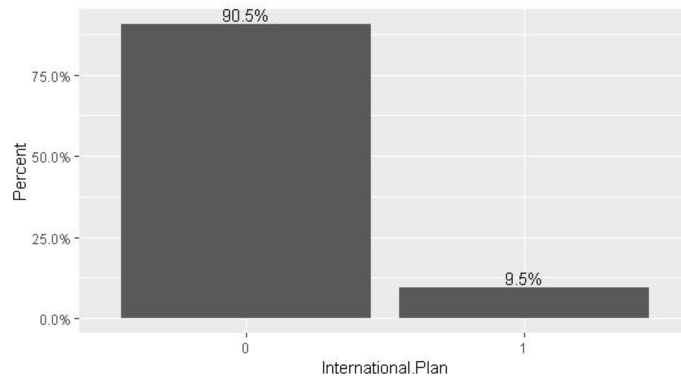
1) number.vmail.messages - Left Skewed
2) total.intl.calls - Left Skewed
3) number.customer.service.calls - Left Skewed

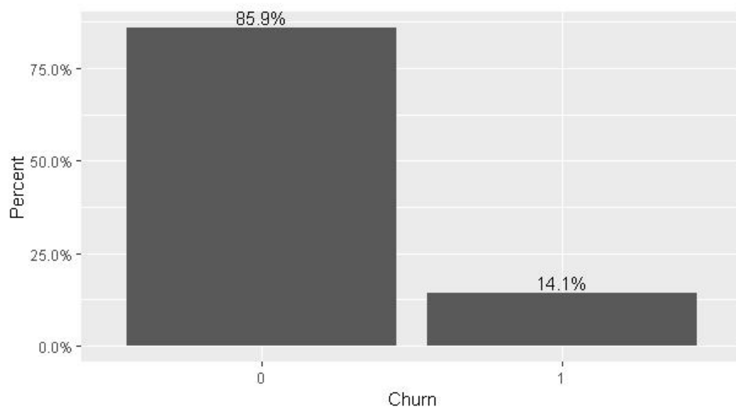
We will use the above information in **Feature Scaling**

Categorical Variables:-

For categorical variables, we'll use percentage of values under each category. It can be be measured using two metrics, Count and Count% against each category. Bar chart is used as visualization.







Missing Values Analysis

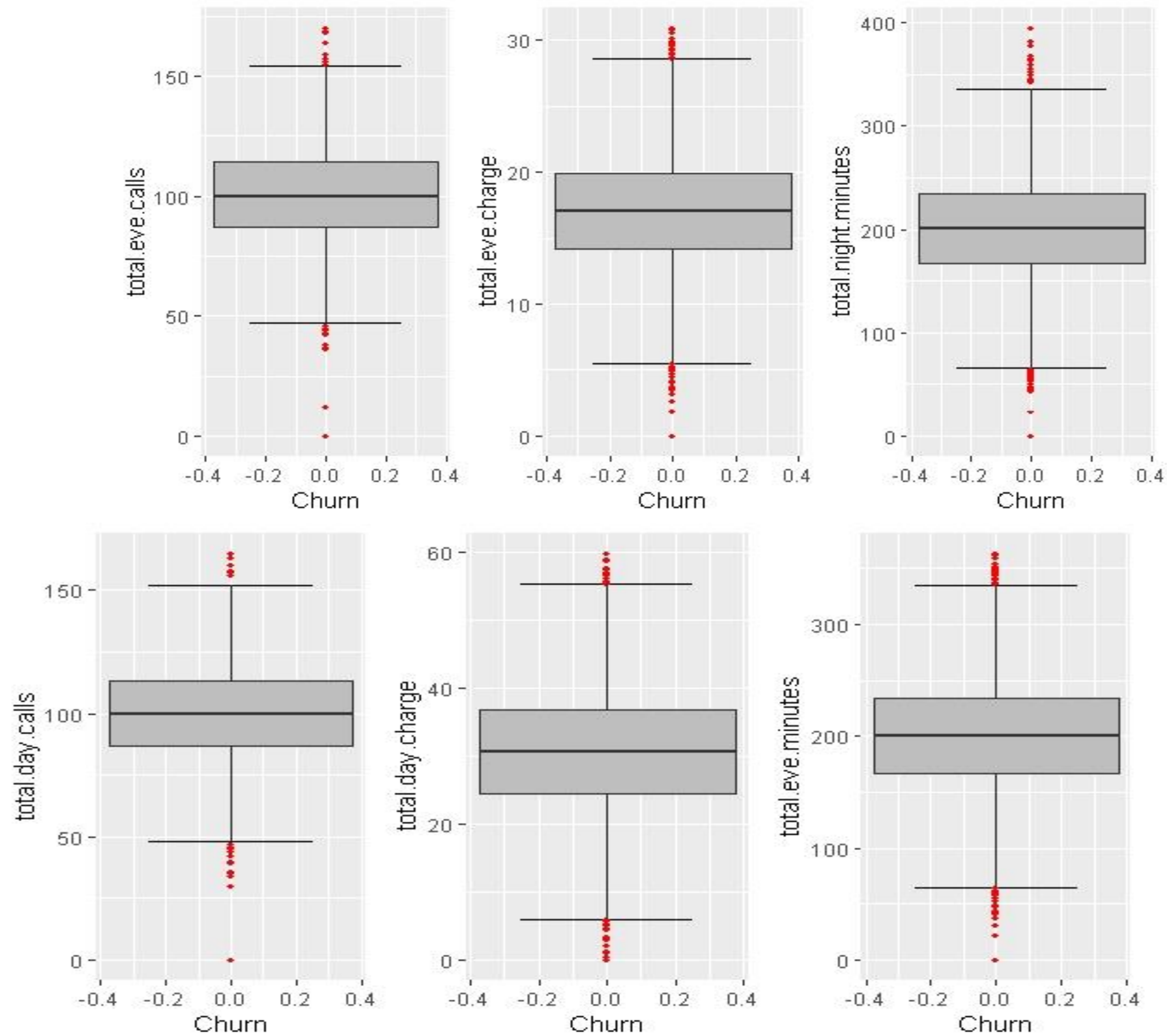
Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

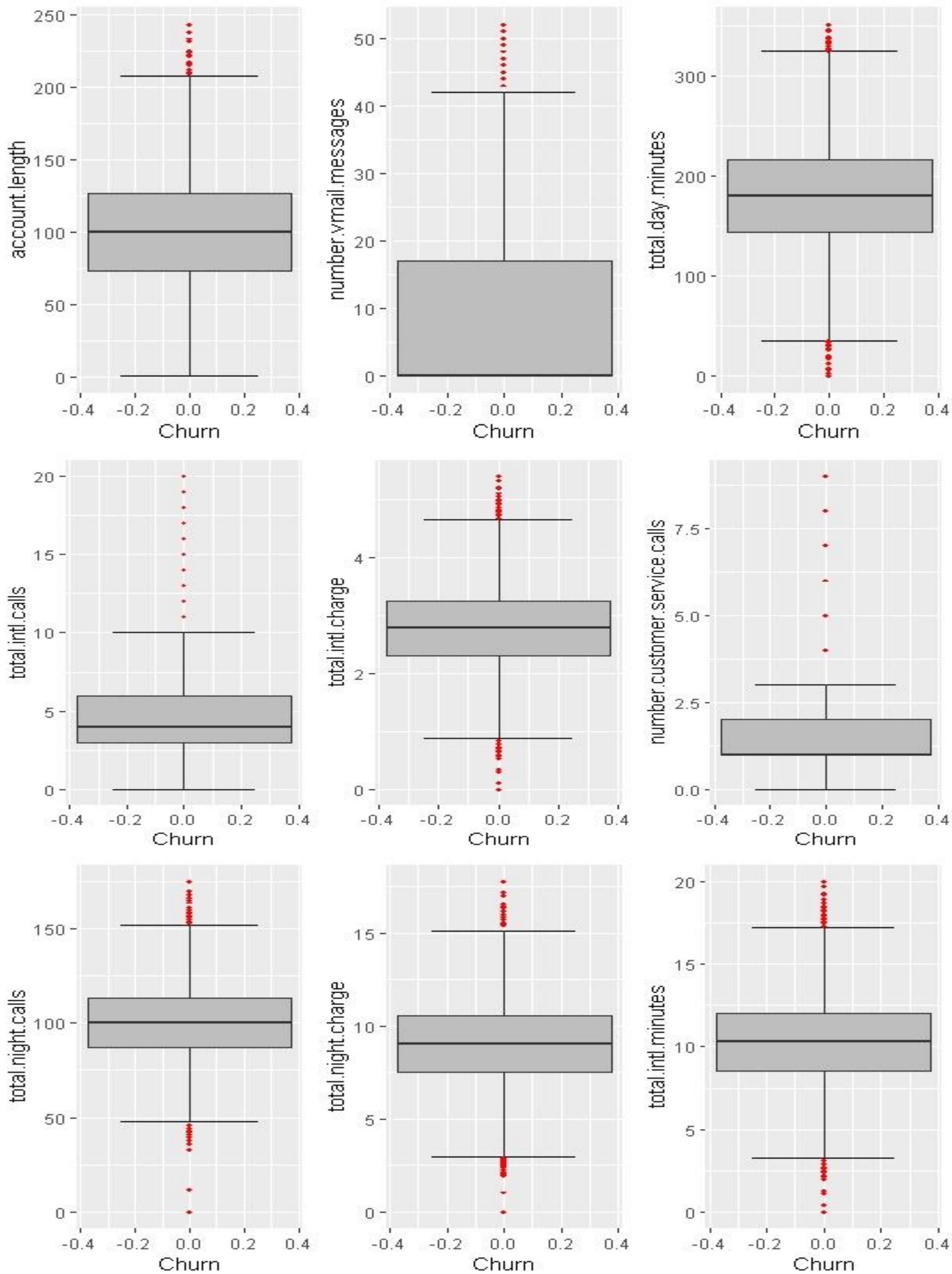
In the given dataset we did not find any missing values.

2.1.1 Outlier Analysis

One of the other steps of **pre-processing** apart from checking for normality is the presence of outliers. We visualize the outliers using *boxplots*.

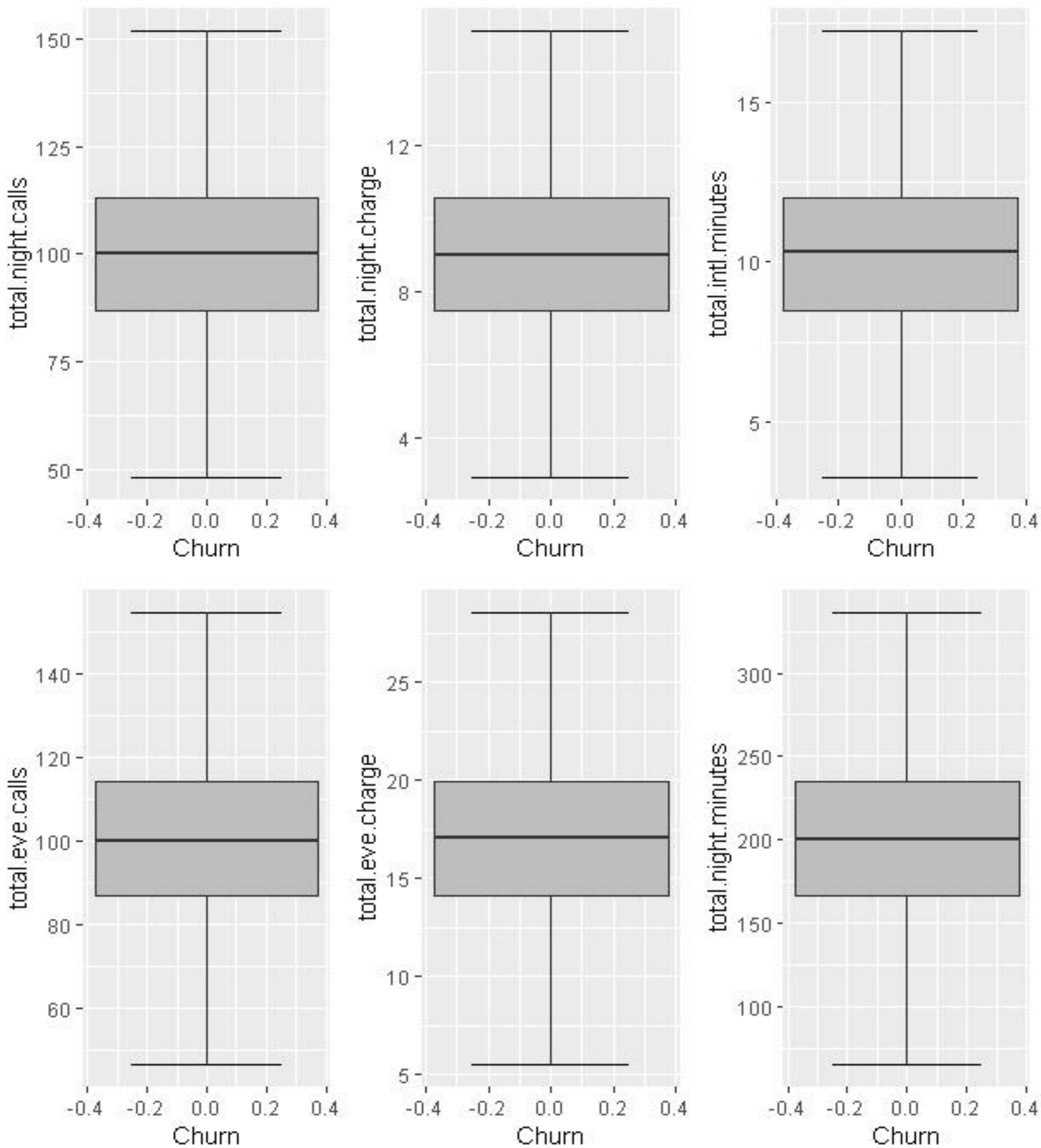
Below we have plotted the boxplots of the 15 continuous predictor variables with respect to Churn.

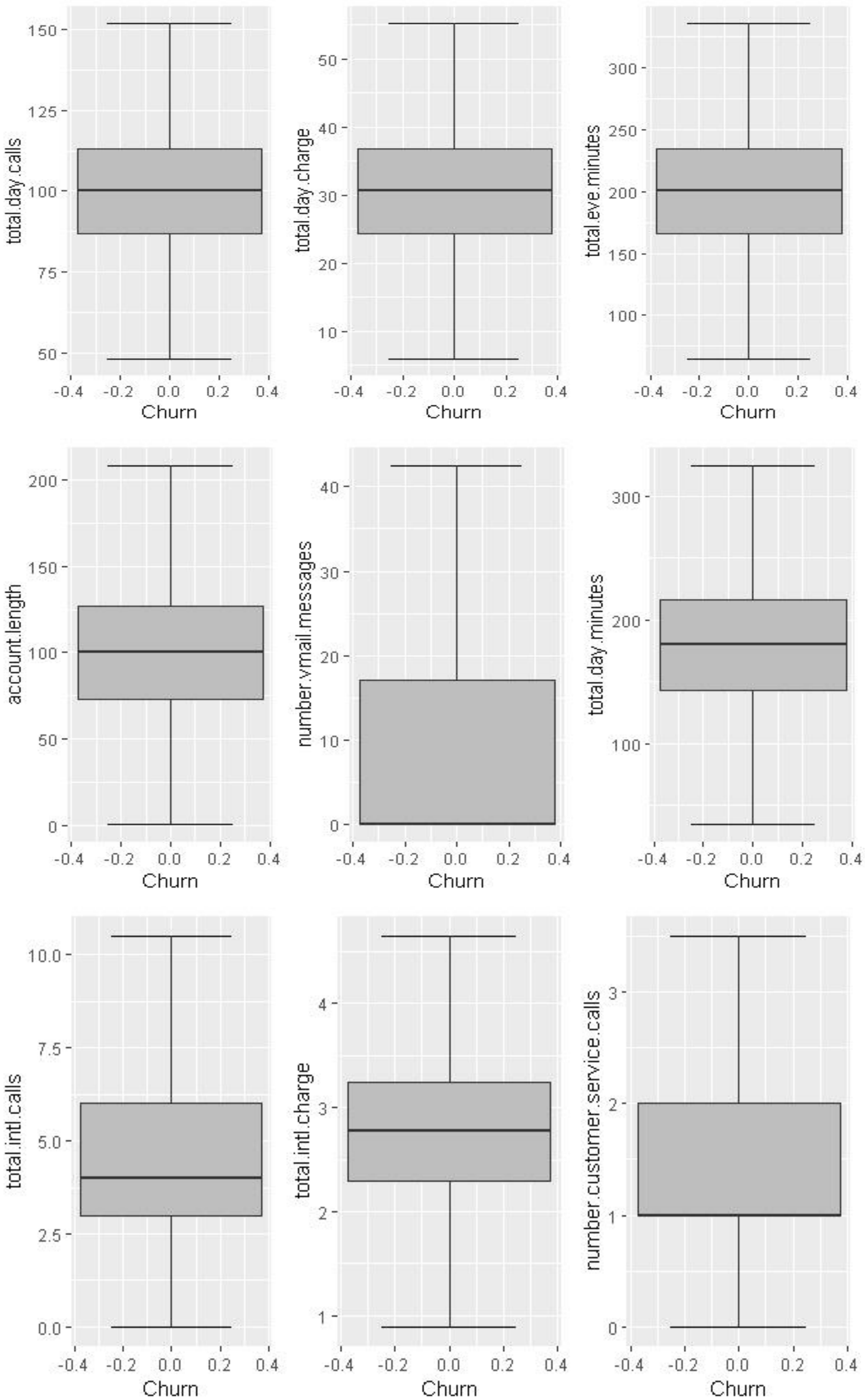




A lot of useful inferences can be made from these plots. First as you can see, we have a lot of outliers and extreme values in each of the data sets.

To deal with the Outliers we replaced them with the corresponding **Maximum and Minimum Values** as per the BoxPlot Statistics.
After performing outlier Analysis and replacing outliers, we plot the boxplots again.





From the above plots it is clear that we do not have any more outliers in our dataset.

2.1.2 Feature Scaling

In our dataset we observed that variables are in different scales and also the variance is very high. Hence we need to Scale these variables so that they are in proportion with each other.

For variables in the dataset which are normally distributed, we will perform Standardisation (-1 to 1) whereas for variables which are not normally distributed we will perform Normalisation(0 to 1)

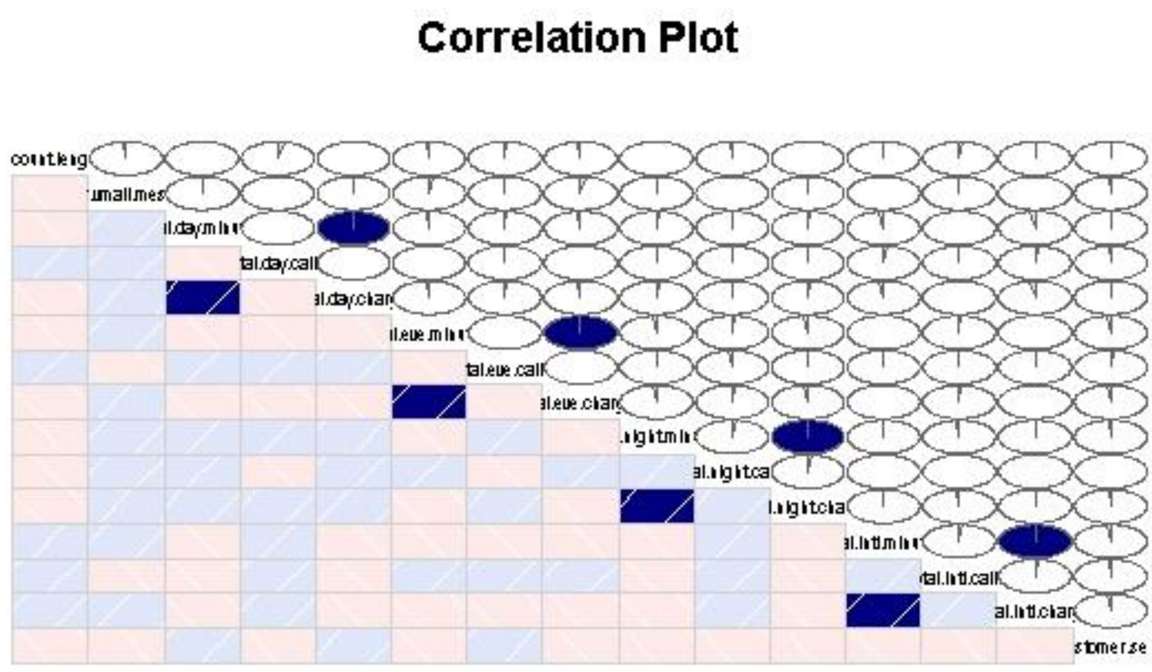
Earlier it was observed that all variables are normally distributed except:
1) number.vmail.messages - Left Skewed
2) total.intl.calls - Left Skewed
3) number.customer.service.calls - Left Skewed

Hence we perform Normalisation for the above 3 variables and Standardisation for all the remaining variables.

2.1.3 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction.

First we look at the correlation in between the **Continuous variables**:



From the above plot it is clear that there is a high positive correlation in between 4 pairs of variables

Total.day.minutes & Total.day.charge
Total.eve.minutes & Total.eve.charge
Total.night.minutes & Total.night.charge
Total.intl.minutes & Total.intl.charge

We also look at VIF (**Variance Inflation Factor**) values to determine collinearity. Below are the results obtained:
>> **vifcor(churn_telecom[1:15], th = 0.9)**

```
4 variables from the 15 input variables have collinearity problem:

total.day.charge total.eve.charge total.night.charge, total.intl.charge

After excluding the collinear variables, the linear correlation coefficients ranges
between:
min correlation ( total.eve.minutes ~ total.day.calls ):  -1.070226e-05
max correlation ( total.day.calls ~ account.length ):  0.02872498

---------- VIFs of the remained variables --------
                    Variables      VIF
1              account.length 1.001561
2         number.vmail.messages 1.000848
3             total.day.minutes 1.000777
4               total.day.calls 1.001246
5             total.eve.minutes 1.001416
6               total.eve.calls 1.000594
7             total.night.calls 1.001075
8           total.night.minutes 1.001367
9            total.intl.minutes 1.001003
10             total.intl.calls 1.000863
11 number.customer.service.calls 1.001009
```

Next we look at the relationship in between Categorical variables and Target Variable by performing a Chi Square Test of Independence:

```
[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
```

For both the Variables p-value is less than 0.05, hence we reject null hypothesis, saying that the target variable (Churn) is dependent on these variables.

We also perform a Chi Square Test amongst the two Categorical Predictor variables:

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

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

data:  table(factor_data$international.plan, factor_data[, i])
X-squared = 0.34273, df = 1, p-value = 0.5583
```

Here the p-value is more than 0.05 , hence we accept null hypothesis , saying that these variables are independent on each other.

Based on the above observations we perform a **Dimension Reduction.**

The variables that we have removed are :

```
total.day.charge
total.eve.charge
total.night.charge
total.intl.charge
```

2.2 Modeling

2.2.1 Model Selection

2.2.2 Decision Tree Classification.

```
               Accuracy : 0.9894
                 95% CI : (0.9812, 0.9947)
    No Information Rate : 0.8663
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9558
 Mcnemar's Test P-Value : 0.002569

            Sensitivity : 0.9878
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9267
             Prevalence : 0.8663
         Detection Rate : 0.8558
   Detection Prevalence : 0.8558
      Balanced Accuracy : 0.9939

       'Positive' Class : 0
```

2.2.3 Random Forest Classification

```
RF_Predictions
      0   1
  0 890   0
  1   9 141

               Accuracy : 0.9913
                 95% CI : (0.9836, 0.996)
    No Information Rate : 0.8644
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.964
 Mcnemar's Test P-Value : 0.007661

            Sensitivity : 0.9900
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9400
             Prevalence : 0.8644
         Detection Rate : 0.8558
   Detection Prevalence : 0.8558
      Balanced Accuracy : 0.9950

       'Positive' Class : 0
```

2.2.4 Logistic Regression

```
logit_Predictions
      0   1
  0 880  10
  1 121  29

               Accuracy : 0.874
                 95% CI : (0.8523, 0.8936)
    No Information Rate : 0.9625
    P-Value [Acc > NIR] : 1

                  Kappa : 0.263
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8791
            Specificity : 0.7436
         Pos Pred Value : 0.9888
         Neg Pred Value : 0.1933
             Prevalence : 0.9625
         Detection Rate : 0.8462
   Detection Prevalence : 0.8558
      Balanced Accuracy : 0.8114

       'Positive' Class : 0
```

2.2.5 Naïve Bayes Classification

```
         predicted
observed   0   1
       0 881   9
       1 111  39

               Accuracy : 0.8846
                 95% CI : (0.8636, 0.9034)
    No Information Rate : 0.9538
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3484
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8881
            Specificity : 0.8125
         Pos Pred Value : 0.9899
         Neg Pred Value : 0.2600
             Prevalence : 0.9538
         Detection Rate : 0.8471
   Detection Prevalence : 0.8558
      Balanced Accuracy : 0.8503

       'Positive' Class : 0
```

Conclusion

3.1 Model Evaluation

3.1.1 Confusion Matrix

Decision Tree Classification

		Predicted 0	Predicted 1
Actual	0	890	0
Actual	1	11	139

Random Forest Classification

		Predicted 0	Predicted 1
Actual	0	890	0
Actual	1	9	141

Logistic Regression

		Predicted 0	Predicted 1
Actual	0	880	10
Actual	1	121	29

Naïve Bayes Classification

		Predicted 0	Predicted 1
Actual	0	881	9
Actual	1	111	39

3.2 Model Selection

From above results it can be understood that Decision Tree and Random Forest have performed well, whereas Logistic Regression, and Naïve Bayes Classification have performed just average.

Based on the Highest Accuracy and least False Positive Rate we choose **Random Forest** as our method of classification

Appendix A – R Code

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

setwd("C:/Users/RAUNAK/Desktop/edwisor/workspace")


#Load Libraries

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

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


install.packages('C50')

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

install_github("kassambara/easyGgplot2")

library(easyGgplot2)

## Read the data

churn_telecom1 = read.csv("Train_data.csv", header = T, na.strings = c(" ", "", "NA"))

churn_telecom2 = read.csv("Test_data.csv", header = T, na.strings = c(" ", "", "NA"))

#Combine the given datasets into one dataset

churn_telecom<- rbind(churn_telecom1, churn_telecom2)

#Save the combined dataset as CSV

write.csv(churn_telecom, "churn_telecom.csv", row.names = F)

#Read the saved dataset

churn_telecom = read.csv("churn_telecom.csv", header = T, na.strings = c(" ", "", "NA"))
```

```
##########################################Explore the data##########################################

#Check Datatypes
str(churn_telecom)

#Remove state, area code,phone_number  columns because we are predicting churn based on Usage and Plans
churn_telecom= subset(churn_telecom, select = -c(state,area.code, phone.number))

#Re-arrange numeric variables and factor variables
churn_telecom <- churn_telecom[, c(1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,2,3,18)]

#Convert all integer/numeric variables to numeric
churn_telecom[1:15] <- sapply(churn_telecom[1:15] , as.numeric)

#Convert factor variables to numeric levels

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

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

   churn_telecom[,i] = factor(churn_telecom[,i],labels=0:(length(levels(factor(churn_telecom[,i])))-1))

 }
}
```

```
#Visualise Histograms for Numeric Variables.

require(ggplot2)
require(scales)
ggplot2.histogram(data=churn_telecom$account.length, xtitle='account.length',
            fill="#FFAAD4", color="#FFAAD4",
            addMeanLine=TRUE, meanLineColor="red",
            meanLineType="dashed", meanLineSize=1, binwidth=8,
            axisLine=c(0.5, "solid", "black")
            )

 ggplot2.histogram(data=churn_telecom$international.plan, xtitle='international.plan',
            fill="#FFAAD4", color="#FFAAD4",
            addMeanLine=TRUE, meanLineColor="red",
            meanLineType="dashed", meanLineSize=1, binwidth=8,
            axisLine=c(0.5, "solid", "black"))

 ggplot2.histogram(data=as.numeric(churn_telecom$voice.mail.plan), xtitle='international.plan',
            fill="#FFAAD4", color="#FFAAD4",
            addMeanLine=TRUE, meanLineColor="red",
            meanLineType="dashed", meanLineSize=1, binwidth=0.1,
            axisLine=c(0.5, "solid", "black"))

 ggplot2.histogram(data=as.numeric(churn_telecom$total.intl.charge), xtitle='total.intl.charge',
            fill="#FFAAD4", color="#FFAAD4",
            addMeanLine=TRUE, meanLineColor="red",
            meanLineType="dashed", meanLineSize=1, binwidth=0.5,
            axisLine=c(0.5, "solid", "black"))
```

```
#Percentage plots for Categorical Variables

 ggplot(churn_telecom, aes(x = international.plan)) +
    geom_bar(aes(y = (..count..)/sum(..count..))) +
   geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))), stat = "count", vjust = -0.25) +
   scale_y_continuous(labels=percent) +
   labs(title = "", y = "Percent", x = "International.Plan")

 ggplot(churn_telecom, aes(x = voice.mail.plan)) +
   geom_bar(aes(y = (..count..)/sum(..count..))) +
   geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))), stat = "count", vjust = -0.25) +
   scale_y_continuous(labels=percent) +
   labs(title = "", y = "Percent", x = "voice.mail.plan")

 ggplot(churn_telecom, aes(x = Churn)) +
   geom_bar(aes(y = (..count..)/sum(..count..))) +
   geom_text(aes(y = ((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))), stat = "count", vjust = -0.25) +
   scale_y_continuous(labels=percent) +
   labs(title = "", y = "Percent", x = "Churn")

##############################Missing Values Analysis###############################################
#Check for null fields
sum(is.na(churn_telecom))
```

```
##################################Outlier Analysis#############################################

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

numeric_index

numeric_data =churn_telecom[,numeric_index]

cnames =  colnames(numeric_data)

 ## BoxPlots - Distribution and Outlier Check

#Generate Box Plots for Numeric variables. The same code has been used for Univariate Analysis
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i])), data = subset(churn_telecom))+
         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")
         )
}
## Plotting plots together
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)
```

```
#Replace outliers with maximum and minimum values

for(i in cnames){

  quantiles <- quantile( churn_telecom[,i], c(.25, .75 ) )

  churn_telecom[,i][ churn_telecom[,i] > (quantiles[2]+1.5*(quantiles[2]-quantiles[1])) ] <-
(quantiles[2]+1.5*(quantiles[2]-quantiles[1]))

  churn_telecom[,i][ churn_telecom[,i] < (quantiles[1]-1.5*(quantiles[2]-quantiles[1])) ] <- (quantiles[1]-1.5*(quantiles[2]-
quantiles[1]))


}



##############################Feature Scaling#######################################33


#Normalisation

cnames_norm = c("number.vmail.messages",

            "total.intl.calls",

            "number.customer.service.calls")


for(i in cnames_norm){

  print(i)

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

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

}

# #Standardisation

cnames_stand = c("account.length",     "total.day.minutes","total.day.calls","total.eve.minutes",
"total.eve.calls","total.night.minutes","total.night.calls","total.intl.minutes")
```

```
  for(i in cnames_stand){
  print(i)
  churn_telecom[,i] = (churn_telecom[,i] - mean(churn_telecom[,i]))/
                         sd(churn_telecom[,i])
 }


#################################Feature Selection############################################


## Correlation Plot
corrgram(churn_telecom[,numeric_index], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")



## Chi-squared Test of Independence
factor_index = sapply(churn_telecom,is.factor)
factor_data = churn_telecom[,factor_index]


for (i in 1:2)
{
 print(names(factor_data)[i])
 print(chisq.test(table(factor_data$voice.mail.plan,factor_data[,i])))
}
```

```
#VIF Test

vifcor(churn_telecom[1:15], th = 0.9)

## Dimension Reduction

#Remove Highly collinear variables.
churn_telecom= subset(churn_telecom, select = -
c(total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

####################################Model Development########################################
#Clean the environment
rmExcept(c("churn_telecom"))

#Check Distribution of Factor Variables
table(churn_telecom$international.plan)

#0    1
#4527  473

table(churn_telecom$voice.mail.plan)
#0    1
#3677 1323
```

```
# ##Stratified Sampling
stratas = strata(churn_telecom, c("international.plan"), size = c(3600, 360), method = "srswor")
train=getdata(churn_telecom,stratas)
train.index = createDataPartition(train$Churn, p = 1, list = FALSE)
test  = churn_telecom[-train.index,]
train= subset(train, select = -c(ID_unit, Prob,Stratum))



##Decision tree for classification

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

#Summary of DT model
summary(C50_model)

#predict for test cases
C50_Predictions = predict(C50_model, test[,-14], type = "class")

summary(C50_Predictions)

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

ConfMatrix_C50
```

```
###Random Forest for Classification

RF_model = randomForest(Churn ~ ., train, importance = TRUE, ntree = 6000)


#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-14])

##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)


#Logistic Regression
logit_model = glm(Churn ~ ., data = train, family = "binomial")

#summary of the model
summary(logit_model)

#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)


##Evaluate the performance of classification model
```

```
ConfMatrix_LP = table(test$Churn, logit_Predictions)

confusionMatrix(ConfMatrix_LP)




#naive Bayes
library(e1071)

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

#predict on test cases #raw
NB_Predictions = predict(NB_model, test[,1:13], type = 'class')

#Look at confusion matrix
Conf_matrix = table(observed = test[,14], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)
```

.

Appendix B – Python Code

~~~
**Load Libraries**



import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
from fancyimpute import KNN

```



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



from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
from sklearn.naive_bayes import GaussianNB
```

**Change Working Directory**




os.chdir(r"C:\Users\RAUNAK\Desktop\edwisor\workspace")
```

**Load Data**




churn_telecom = pd.read_csv("churn_telecom.csv")

```

**Exploratory Data Analysis**



churn_telecom.info()
```
~~~

```
#Remove state, area code,phone_number  columns because we are predicting churn based on Usage and Plans
churn_telecom = churn_telecom.drop(['state','area.code', 'phone.number'], axis=1)
```

```
#Assigning levels to the categories of Object type variables
lis = []
for i in range(0, churn_telecom.shape[1]):
    print(i)
    if(churn_telecom.iloc[:,i].dtypes == 'object') :
        churn_telecom.iloc[:,i] = pd.Categorical(churn_telecom.iloc[:,i])
        #print(churn_telecom[[i]])
        churn_telecom.iloc[:,i] = churn_telecom.iloc[:,i].cat.codes
        churn_telecom.iloc[:,i] = churn_telecom.iloc[:,i].astype('object')
        lis.append(churn_telecom.columns[i])
```

Missing Values Analysis

```
#Check if there are any missing values
churn_telecom.isnull().sum()
```

Outlier Analysis

```
# #Plot boxplot to visualize Outliers

plt.boxplot(churn_telecom['total.day.minutes'])
```

```
#Replace outliers with maximum and minimum values
for i in range(0, churn_telecom.shape[1]):
    print(i)
    if(churn_telecom.iloc[:,i].dtypes != 'object') :
        q75, q25 = np.percentile(churn_telecom.iloc[:,i], [75 ,25])
        churn_telecom.iloc[churn_telecom.loc[(churn_telecom.iloc[:,i]> q75+1.5*(q75-q25) )].index.values,i ] = q75+1.5*(q75-q25)
        churn_telecom.iloc[churn_telecom.loc[(churn_telecom.iloc[:,i]< q25-1.5*(q75-q25) )].index.values,i ] = q25-1.5*(q75-q25)


```

Feature Scaling

```
#Normalisation
cnames_norm = ["number.vmail.messages",
               "total.intl.calls",
               "number.customer.service.calls"]
for i in cnames_norm:
    print(i)
    churn_telecom[i] = (churn_telecom[i] -
min(churn_telecom[i]))/(max(churn_telecom[i]) - min(churn_telecom[i]))
```



```
#Standardisation
cnames_stand = ["account.length",
                 "total.day.minutes",
                 "total.day.calls",
                 "total.eve.minutes",
                 "total.eve.calls",
                 "total.night.minutes",
                 "total.night.calls",
                 "total.intl.minutes"]

for i in cnames_stand:
     print(i)
     churn_telecom[i] = (churn_telecom[i] -
churn_telecom[i].mean())/churn_telecom[i].std()
```

Feature Selection

```
##Correlation analysis
#Correlation plot

cnames_numeric =["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",
                 "number.vmail.messages",
                 "number.customer.service.calls"]

df_corr = churn_telecom.loc[:,cnames_numeric]

#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
cnames_object = ["international.plan","voice.mail.plan"]

#loop for chi square values
for i in cnames_object:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(churn_telecom['Churn'],
churn_telecom[i]))
    print(p)
```

```
#Dimension Reduction

churn_telecom =
churn_telecom.drop(['total.day.charge','total.eve.charge','total.night.charge','total
.intl.charge'], axis=1)

```

Model Development

```
 #Stratified sampling

#Select categorical variable
 y = churn_telecom['international.plan']

#select subset using stratified Sampling
Rest, Sample = train_test_split(churn_telecom, test_size = 0.8, stratify = y)
```

```
#replace target categories with Yes or No
churn_telecom['Churn'] = churn_telecom['Churn'].replace( 0,'No')
churn_telecom['Churn'] = churn_telecom['Churn'].replace (1,'Yes')
```

```
churn_telecom.head(5)
```

```
#Divide data into train and test
X = churn_telecom.values[:, 0:14]
Y = churn_telecom.values[:,13]
Y = Y.astype('int')

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

```
#Decision Tree
C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)
```

```
#predict new test cases
C50_Predictions = C50_model.predict(X_test)
```

```
#build 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)
```

```
#Random Forest

RF_model = RandomForestClassifier(n_estimators = 20).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)
```

```
#Let us prepare data for logistic regression
#replace target categories with Yes or No
churn_telecom['Churn'] = churn_telecom['Churn'].replace(0,'No')
churn_telecom['Churn'] = churn_telecom['Churn'].replace(1,'Yes')
```

```
churn_telecom_logit = pd.DataFrame(churn_telecom['Churn'])
```

```
churn_telecom_logit = churn_telecom_logit.join(churn_telecom[cnames_numeric])
```

```
##Create dummies for categorical variables
cat_names = ["international.plan","voice.mail.plan"]

for i in cat_names:
    temp = pd.get_dummies(churn_telecom[i], prefix = i)
    churn_telecom_logit = churn_telecom_logit.join(temp)
```

```
Sample_Index = np.random.rand(len(churn_telecom_logit)) < 0.8
```

```
train = churn_telecom_logit[Sample_Index]
test = churn_telecom_logit[~Sample_Index]
```

```
#select column indexes for independent variables
train_cols = train.columns[1:13]
```

```
#Built Logistic Regression

logit = sm.Logit(train['Churn'], train[train_cols]).fit()

logit.summary()
```

```
#Predict test data
test['Actual_prob'] = logit.predict(test[train_cols])

test['ActualVal'] = 1
test.loc[test.Actual_prob < 0.5, 'ActualVal'] = 0
```

```
#Build confusion matrix
CM = pd.crosstab(test['Churn'], test['ActualVal'])

#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)

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

```
#Naive Bayes
```

```
#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)

```