Churn Reduction

Sayantan Adak

19 Oct. 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

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. We would like to predict whether a new customer will churn out or not based on previously known customer behaviour.

1.2 Data

Our task is to build a classification model which will classify the customer behaviour(Churn or not churn) depending upon multiple factors. Given below is the top 5 observations of the training data set that we are going to use for classification.

state	account.length	area.code	phone.number	international.plan	voice.mail.plan	number.vmail.messages
KS	128	415	382-4657	no	yes	25
OH	107	415	371-7191	no	yes	26
NJ	137	415	358-1921	no	no	0
OH	84	408	375-9999	yes	no	0
OK	75	415	330-6626	yes	no	0

Table 1.1: Training data (Column 1-7)

Table 1.2: Training data (Column 8-14)

total.day.minutes	total.day.calls	total.day.charge	total.eve.minutes	total.eve.calls	total.eve.charge	total.night.minutes
265.1	110	45.07	197.4	99	16.78	244.7
161.6	123	27.47	195.5	103	16.62	254.4
243.4	114	41.38	121.2	110	10.3	162.6
299.4	71	50.9	61.9	88	5.26	196.9
166.7	113	28.34	148.3	122	12.61	186.9

Table 1.3: Training data (Column 14-21)

total.night.calls	total.night.charge	total.intl.minutes	total.intl.calls	total.intl.charge	number.customer.service.calls	Churn
91	11.01	10	3	2.7	1	False
103	11.45	13.7	3	3.7	1	False
104	7.32	12.2	5	3.29	0	False
89	8.86	6.6	7	1.78	2	False
121	8.41	10.1	3	2.73	3	False

As we can see here are total 20 predictor variables and our target variable is the 21th variable(Churn).

Table 1.4: Predictor Variables:

Serial No.	Predictors
1	State
2	Account.length
3	Area.code
4	Phone.number
5	International.plan
6	Voice.mail.plan
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

Chapter 2

Methodology

2.1 Exploratory data analysis

Data exploration is a crucial stage for any predictive model. The quality of the input decides the quality of the output.

Train Data: The predictive model is always built on train data set. An intuitive way to identify the train data is, that it always has the 'response variable' included.

Test Data: Once the model is built, it's accuracy is 'tested' on test data. This data always contains less number of observations than train data set. Also, it does not include 'response variable'.

Need for Data Cleaning or Data Preparation:

- Dataset might contain discrepancies in the names or codes.
- Dataset might contain outliers or errors.
- Dataset lacks your attributes of interest for analysis.

- All in all the dataset is not qualitative but is just quantitative.

From the experiences of many data scientists it is said that the data exploration, cleaning and preparation can take upto 70% time of the total project.

2.1.1. Data pre-processing

If we see the structure of the data set then we can observe there are different kinds of variable: few are numerical and few are categorical. So we have to treat different kinds of variables in different ways.

```
'data.frame':     3333 obs. of  21 variables:
 $ state                  : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50 ...
 $ account.length           : int  128 107 137 84 75 118 121 147 117 141 ...
 $ area.code                : int  415 415 415 408 415 510 510 415 408 415 ...
 $ phone.number             : Factor w/ 3333 levels " 327-1058"," 327-1319",..:
 $ international.plan       : Factor w/ 2 levels " no"," yes": 1 1 1 2 2 2 1 2 1 2 ...
 $ voice.mail.plan          : Factor w/ 2 levels " no"," yes": 2 2 1 1 1 1 2 1 1 2 ...
 $ number.vmail.messages      : int  25 26 0 0 0 0 24 0 0 37 ...
 $ total.day.minutes          : num  265 162 243 299 167 ...
 $ total.day.calls          : int  110 123 114 71 113 98 88 79 97 84 ...
 $ total.day.charge           : num  45.1 27.5 41.4 50.9 28.3 ...
 $ total.eve.minutes          : num  197.4 195.5 121.2 61.9 148.3 ...
 $ total.eve.calls          : int  99 103 110 88 122 101 108 94 80 111 ...
 $ total.eve.charge           : num  16.78 16.62 10.3 5.26 12.61 ...
 $ total.night.minutes        : num  245 254 163 197 187 ...
 $ total.night.calls          : int  91 103 104 89 121 118 118 96 90 97 ...
```

$ total.night.charge : num 11.01 11.45 7.32 8.86 8.41 ...

$ total.intl.minutes : num 10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...

$ total.intl.calls : int 3 3 5 7 3 6 7 6 4 5 ...

$ total.intl.charge : num 2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...

$ number.customer.service.calls: int 1 1 0 2 3 0 3 0 1 0 ...

$ Churn : Factor w/ 2 levels " False."," True.": 1 1 1 1 1 1 1 1 1 1 ...

If you closely look at **area.code** variable then you can see it is categorized as numerical variable(int) but there are only three unique values(408,415,510). So we can convert it to a categorical variable with each unique value as a category to make the data more precise.

Now look at the variable **number.vmail.messages**. You can see out of 3333 observations 2411 are 0s and the distribution looks like this:

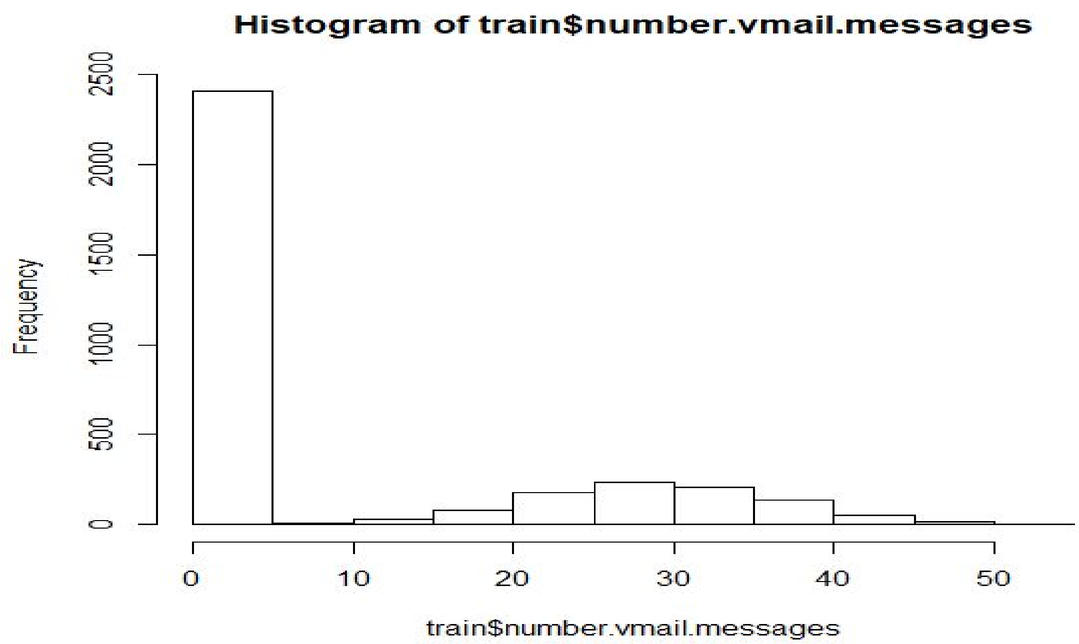


Figure 2.1.1: histogram of number.vmail.messages

It is heavily left skewed. If we put this variable in the model, the output can be biased. So we can treat the **0** as a **missing value** in the data.

2.1.2 Missing Value Analysis

Now check if the data contains any missing value.

state account.length area.code phone.number international.plan

0	*0*	*0*	*0*	*0*
voice.mail.plan	*number.vmail.messages*	*total.day.minutes*	*total.day.calls*	*total.day.charge*
0	*2411*	*0*	*0*	*0*
total.eve.minutes	*total.eve.calls*	*total.eve.charge*	*total.night.minutes*	*total.night.calls*
0	*0*	*0*	*0*	*0*
total.night.charge	*total.intl.minutes*	*total.intl.calls*	*total.intl.charge*	*number.customer.service.calls*
0	*0*	*0*	*0*	*0*
Churn				
0				

As we can see only the variable **number.vmail.messages** contains 2411 missing values. We can impute the missing values with **mean** or **median** or **mode** or apply **knn imputation.** But in this variable above 50% of the observations contains missing values. So we need to drop that variable from the data for better accuracy.

2.1.3 Outlier Analysis

We can clearly observe from the distribution that most of the variables are either right or left skewed. The skewness in the distribution is most likely explained by the presence of outliers and extreme values in the data.

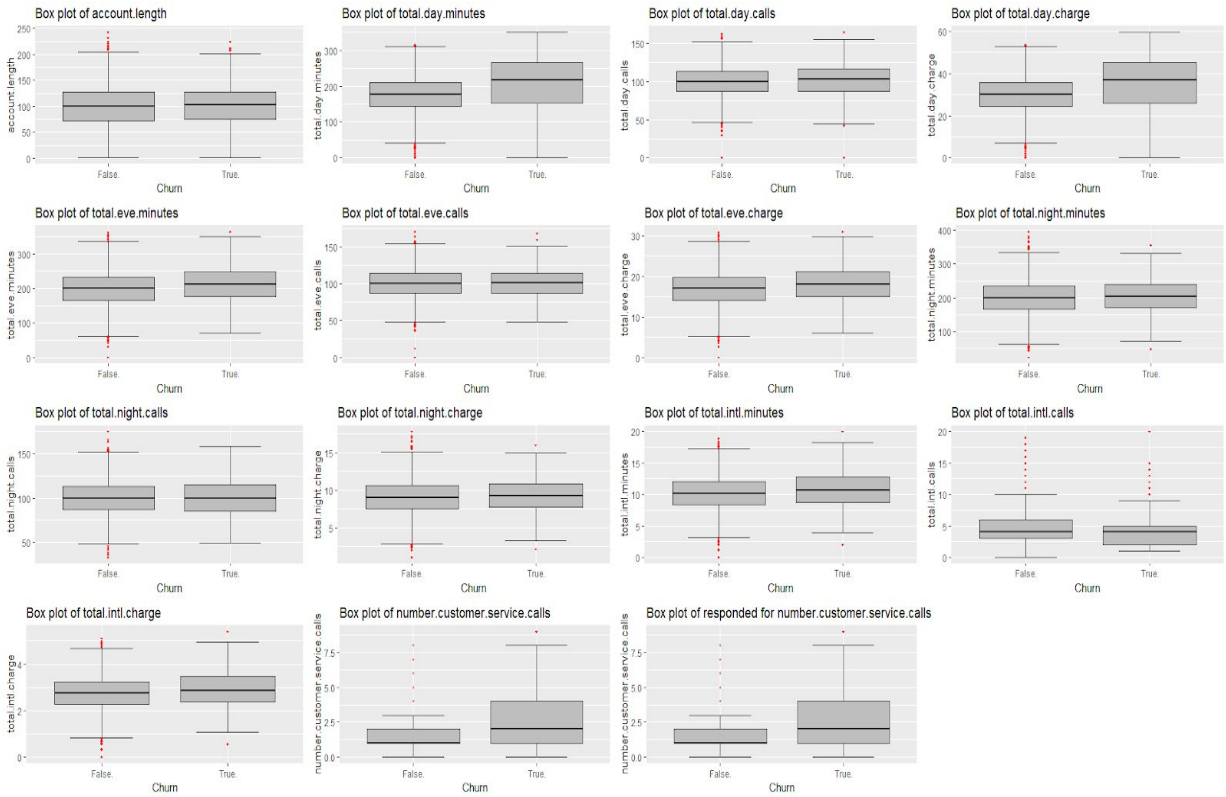


Figure 2.1.3.1: Boxplot of predictors with outliers

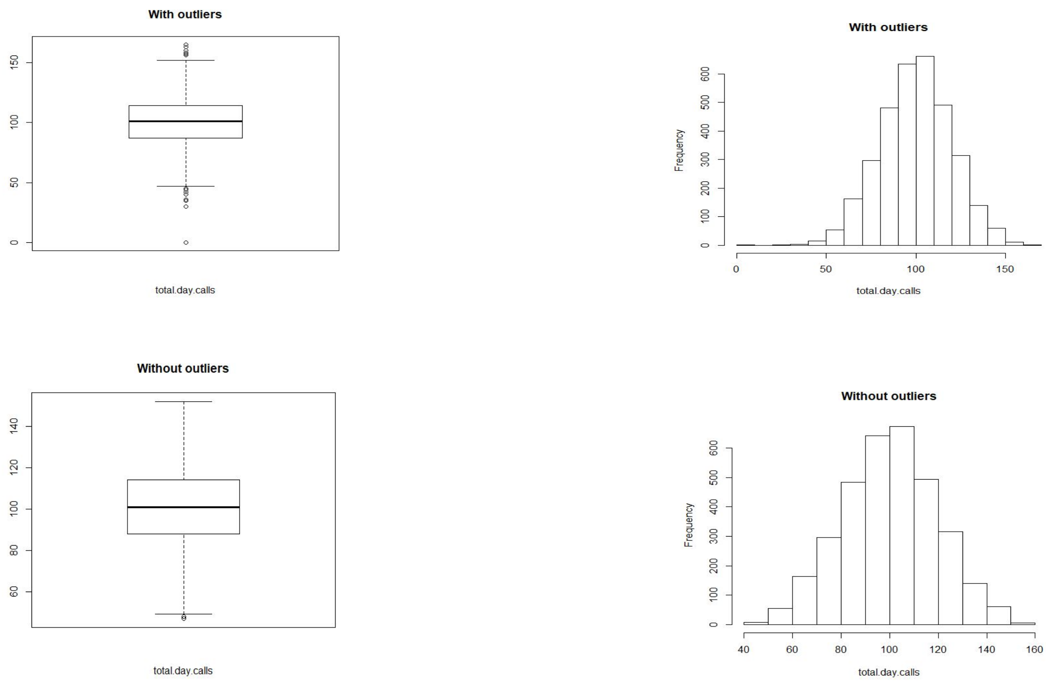


Figure 2.1.3.2: Effect of outliers in **total.day.calls**

The ouliers can be replaced with missing values. Then we can do missing value analysis on top of it.

Here we have used **knnImputation** method to impute the missing values.

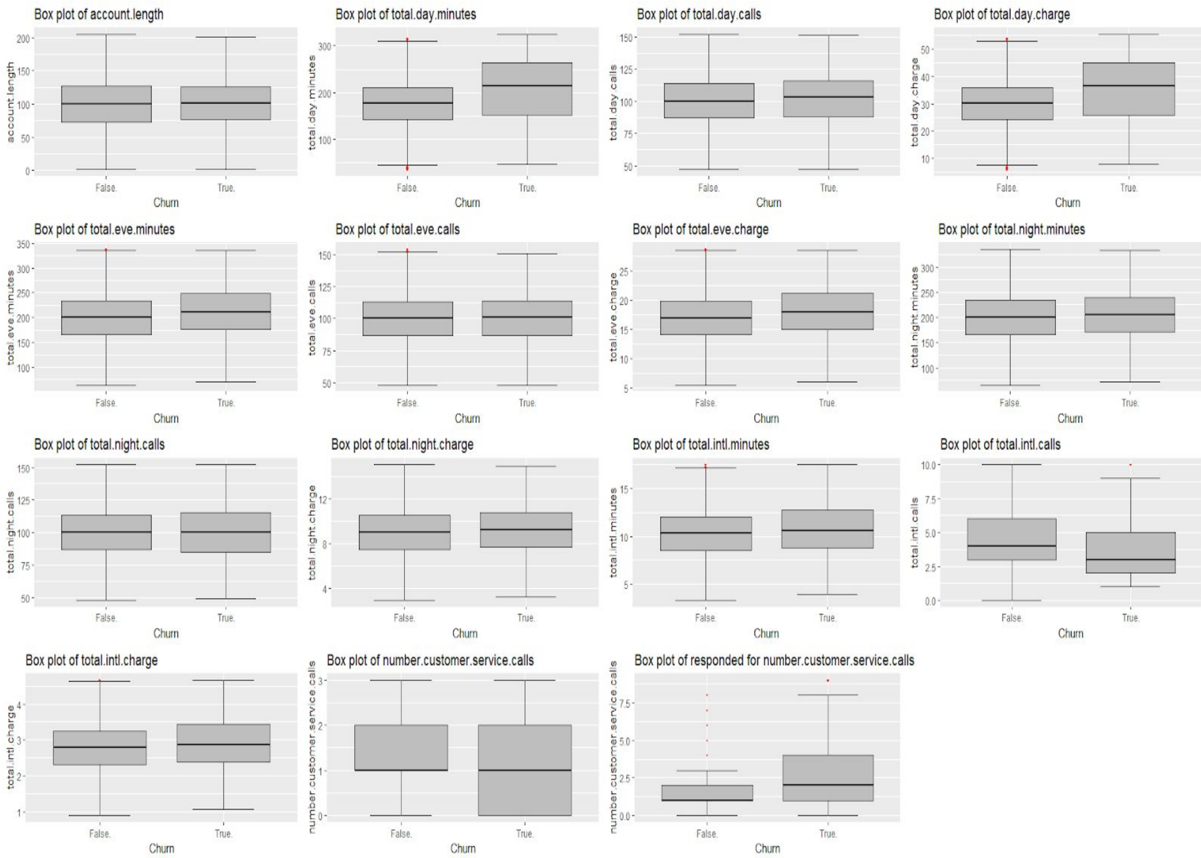


Figure 2.1.3.3: Boxplot of predictors without outliers

2.1.4 Feature Selection

Before performing any type of modelling 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. There are several methods of doing that. In our dataset few variables are continuous and few variables are categorical. We have used correlation plot for ignoring statistically non-significant numerical features and **chi-square test** for selecting important categorical features.

Below is the correlation plot for all the continuous variables.

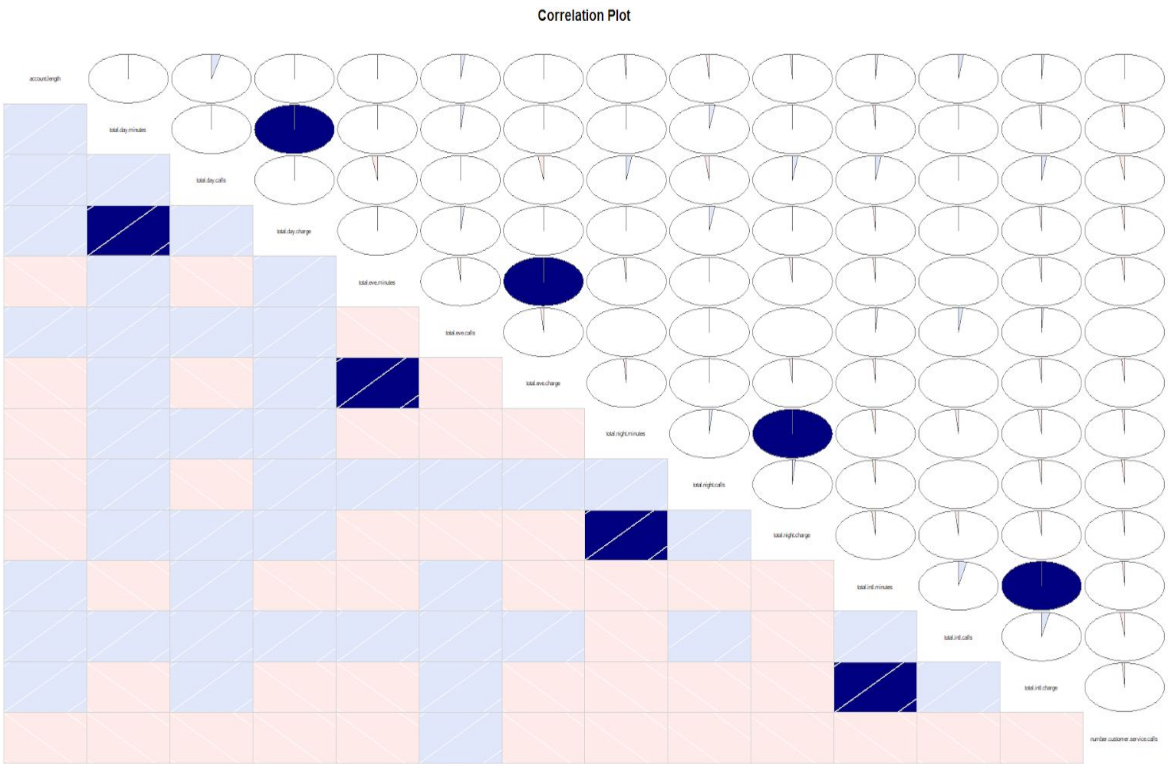


Figure 2.1.4: Correlation Plot

Here the blue circle or blue square indicates there is highly positive correlation between the two variable. For e.g. **total.day.minutes** and **total.day.charge** are highly correlated i.e they are carrying same kind of information. So we can ignore either of the variables. We ignored **total.day.charge, total.eve.charge, total.night.charge, total.intl.charge.**

Next we have used **chi-square test** for the categorical variables. Below is the list of chi-square result for each of the predictor(categorical) with target variable(Churn).

[1] "state"

X-squared = 83.044, df = 50, p-value = 0.002296

[2] "area.code"

X-squared = 0.17754, df = 2, p-value = 0.9151

[3] "phone.number"

X-squared = 3333, df = 3332, p-value = 0.4919

[4] "international.plan"

X-squared = 222.57, df = 1, p-value < 2.2e-16

Here we can see for **state** and **international.plan** the **p-values** are less than critical value(0.05). So we can reject the null hypothesis saying that the **Churn** is dependent on these two variables. But for other two variables the **p-values** are greater than critical value(0.05). So we can accept the null hypothesis for **chi-square** test and assume that **Churn** is not dependent on **area.code** and **phone.number.**

So altogether we have selected a total of 14 variables from the raw data and discarded other variables (**number.vmail.messages**, **total.day.charge, total.eve.charge, total.night.charge, total.intl.charge, area.code, phone.number**).

2.1.5 Feature Scaling

In the raw data there are different feature of different scale in their magnitude. If we feed them directly into the machine learning model without scaling, object function will not work out properly and the result will be biased. That's why we have gone for feature scaling as a data pre-processing technique to reduce variation either within or between variables. We have normalized all the numeric predictors to a fixed range using the formula:

$$\mathbf{Value_{new} = (Value - minValue)/(maxValue - minValue)}$$

This changes the range of each numerical predictor variables to 0 to 1.

2.2 Model Building

2.2.1 Model Selection

Our aim is to classify whether a customer will churn out or not. So the dependent variable is nominal. We can build a binary classification model for this problem statement. First we have made a simple **Decision Tree** model for the classification. Then we would like to build more complex model for better result.

2.2.2 Decision Tree

Decision tree is a rule. Each branch connects nodes with "and" and multiple branches are connected by "or".

First we have built a **C5.0** decision tree model using the cleaned and prepared data.

```
c50_model <- C5.0(Churn ~.,normalized_train, trials = 100, rules = T)

test_predict <- predict(c50_model,normalized_test[,-14],type = "class")

xtab <- table(observed = normalized_test[,14], predicted = test_predict)

confusionMatrix(xtab)
```

```
Confusion Matrix and Statistics

          predicted
observed   False.  True.
   False.    1437      6
   True.      108    116

               Accuracy : 0.9316
                 95% CI : (0.9184, 0.9433)
```

```
    No Information Rate : 0.9268
    P-Value [Acc > NIR] : 0.2424

                  Kappa : 0.636
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9301
            Specificity : 0.9508
         Pos Pred Value : 0.9958
         Neg Pred Value : 0.5179
             Prevalence : 0.9268
         Detection Rate : 0.8620
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9405

       'Positive' Class :  False.
```

As we can see we have predicted a model with 93% accuracy. But here the Negative Prediction value is quite low(52%). Let us check if we can improve this result with **CART** model.

rpart_model <- rpart(Churn~.,data = normalized_train,method = "class")

pred <- predict(rpart_model,normalized_test[,-14],type = "class")

xtab <- table(observed = normalized_test[,14], predicted = pred)

confusionMatrix(xtab)

```
Confusion Matrix and Statistics

         predicted
observed   False.  True.
   False.    1425     18
   True.      123    101

               Accuracy : 0.9154
                 95% CI : (0.901, 0.9283)
    No Information Rate : 0.9286
    P-Value [Acc > NIR] : 0.982

                  Kappa : 0.5467
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9205
            Specificity : 0.8487
         Pos Pred Value : 0.9875
         Neg Pred Value : 0.4509
             Prevalence : 0.9286
         Detection Rate : 0.8548
```

```
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8846

       'Positive' Class :  False.
```

We can see the **C5.0** model gives better prediction than **CART** model.

2.2.3 Random Forest

Let us check if we can improve the result using **Ensemble** machine learning method. An ensemble method is nothing but combining multiple base models and taking the average of each base models.

A Random Forest uses multiple decision trees to classify or predict by taking the average of all the decision tree results.

First we have used a Random Forest with 200 decision trees and see the output below:

randomforest_model <- randomForest(Churn~.,normalized_train,importance=T, ntree=200)

pred <- predict(randomforest_model,normalized_test[,-14])

xtab <- table(observed = normalized_test[,14], predicted = pred)

confusionMatrix(xtab)

```
Confusion Matrix and Statistics


            predicted
observed    False.  True.
   False.     1430     13
   True.       126     98

               Accuracy : 0.9166
                 95% CI : (0.9023, 0.9294)
    No Information Rate : 0.9334
    P-Value [Acc > NIR] : 0.9967

                  Kappa : 0.5445
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9190
            Specificity : 0.8829
         Pos Pred Value : 0.9910
         Neg Pred Value : 0.4375
             Prevalence : 0.9334
```

```
          Detection Rate : 0.8578
    Detection Prevalence : 0.8656
       Balanced Accuracy : 0.9010

        'Positive' Class :  False.
```

Now take look at result with 300 decision trees

randomforest_model <- randomForest(Churn~.,normalized_train,importance=T, ntree=300)

pred <- predict(randomforest_model,normalized_test[,-14])

xtab <- table(observed = normalized_test[,14], predicted = pred)

confusionMatrix(xtab)

```
Confusion Matrix and Statistics

         predicted
observed   False.  True.
   False.    1430     13
   True.      129     95

               Accuracy : 0.9148
                 95% CI : (0.9004, 0.9278)
    No Information Rate : 0.9352
    P-Value [Acc > NIR] : 0.9995

                  Kappa : 0.5313
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9173
            Specificity : 0.8796
         Pos Pred Value : 0.9910
         Neg Pred Value : 0.4241
             Prevalence : 0.9352
         Detection Rate : 0.8578
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8984

       'Positive' Class :  False.
```

With 300 decision trees:

randomforest_model <- randomForest(Churn~.,normalized_train,importance=T, ntree=500)

pred <- predict(randomforest_model,normalized_test[,-14])

xtab <- table(observed = normalized_test[,14], predicted = pred)

confusionMatrix(xtab)

```
Confusion Matrix and Statistics

         predicted
observed   False.  True.
   False.    1427     16
   True.      130     94

               Accuracy : 0.9124
                 95% CI : (0.8978, 0.9256)
    No Information Rate : 0.934
    P-Value [Acc > NIR] : 0.9997

                  Kappa : 0.5204
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9165
            Specificity : 0.8545
         Pos Pred Value : 0.9889
         Neg Pred Value : 0.4196
             Prevalence : 0.9340
         Detection Rate : 0.8560
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8855

       'Positive' Class :  False.
```

Chapter 3

Conclusion

3.1 Model Evaluation

Now we have few models for predicting target variable, we need to decide which model to choose. The performance of any classification model does not only depend upon its prediction accuracy. Depending upon business understanding we need to consider different kinds of metrics to evaluate our models. The choice of metric completely depends on the type of model and the implementation plan of the model.

There are different types of metrics depend on what kind of problem you are trying to solve:

For Classification problem:

a. Confusion Matrix
    i. Accuracy
    ii. Recall
    iii. Sensitivity
    iv. Specificity
    v. F1
    vi. False Positive Rate
    vii. False Negative Rate

Confusion Matrix: It is also called error matrix or transition matrix. It helps to evaluate the performance of the classification model. It shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. The following table displays a 2x2 confusion matrix for two classes (Yes and No).

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

The entries in the confusion matrix have the following meaning in the context of our study:

- a is the number of correct predictions that an instance is Yes,
- b is the number of incorrect predictions that an instance is No,
- c is the number of incorrect of predictions that an instance Yes, and
- d is the number of correct predictions that an instance is No.

It is again supervised learning method and can be used for binary or multi class classifier problem.

- Accuracy: the proportion of the total number of predictions that was correct. It say how accurately model can able to classify. It can be calculated as (TP+TN)/Total observations
- Misclassification Error: Classifying a record as belonging to one class when it belongs to another class. It can be calculated as (FP+FN)/Total observations
- Sensitivity or Recall: the proportion of actual positive cases which are correctly identified. TP/TP+FN
- Specificity: the proportion of actual negative cases which are correctly identified. TN/TN+FP
- False positive rate or fall out: indicates a given condition has been fulfilled, when it actually has not been fulfilled. FP/FP+TN
- False Negative rate or miss rate: test result indicates that a condition failed, while it actually was successful. FN/FN+TP

In this case if our model predicts that a customer will not churn out but he is actually going to churn out then industry will lose revenue as the company is going to spend money and resources on that customer but the customer is going to churn out. So False Negative rate is going to take an important aspect for model selection.

3.2 Model Selection

From the above models we can see the **Decision Tree C5.0** model gives better accuracy and better **False Negative** rate(1- **negative prediction**). So we can select this model for our final evaluation.

Generally ensemble methods give better performance than it’s base model. But in our case Decision Tree outperformed the Random Forest performance. This happens when the data set is quite small and Decision Tree is stable enough.

Appendix A – Extra Figures

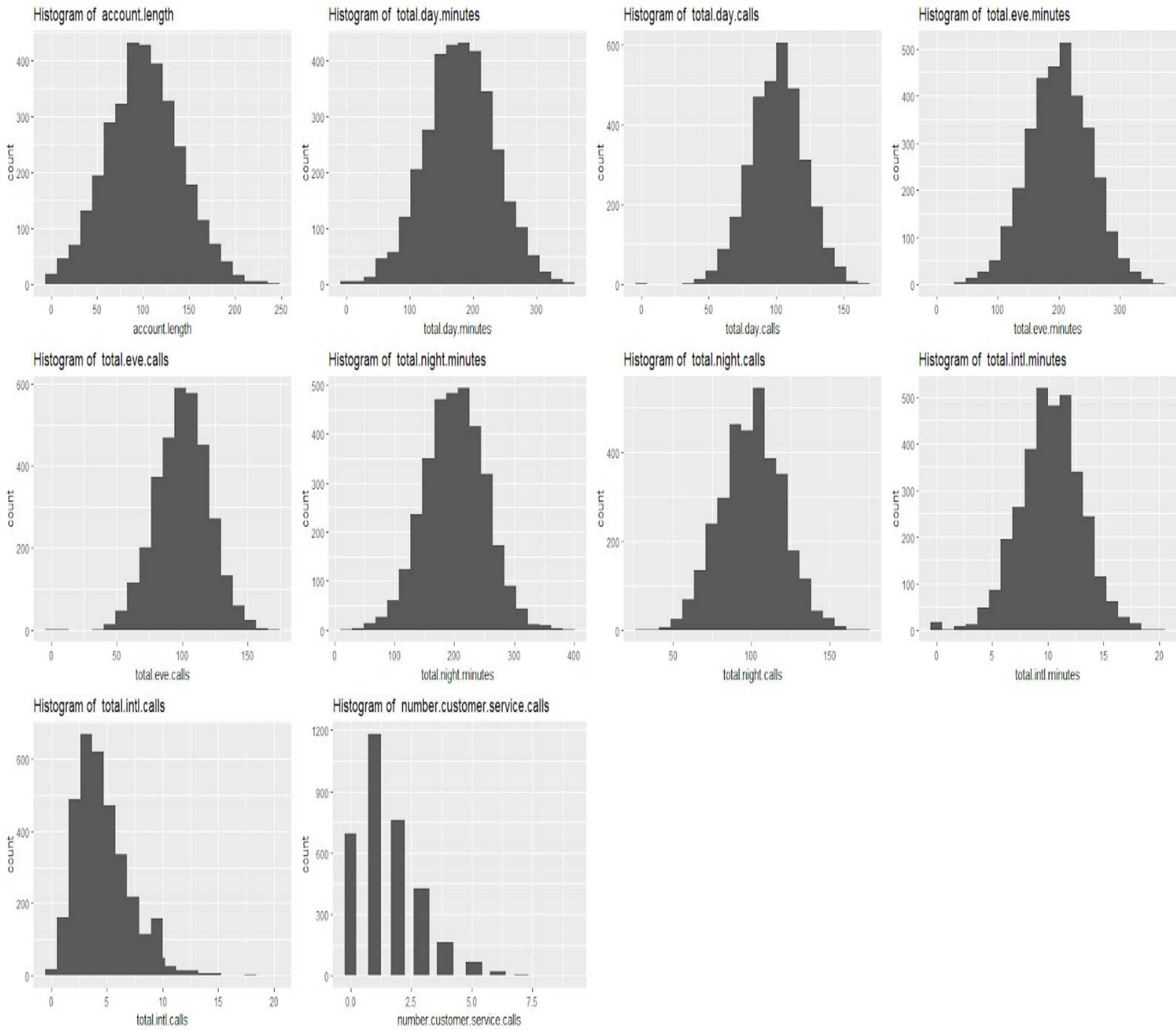
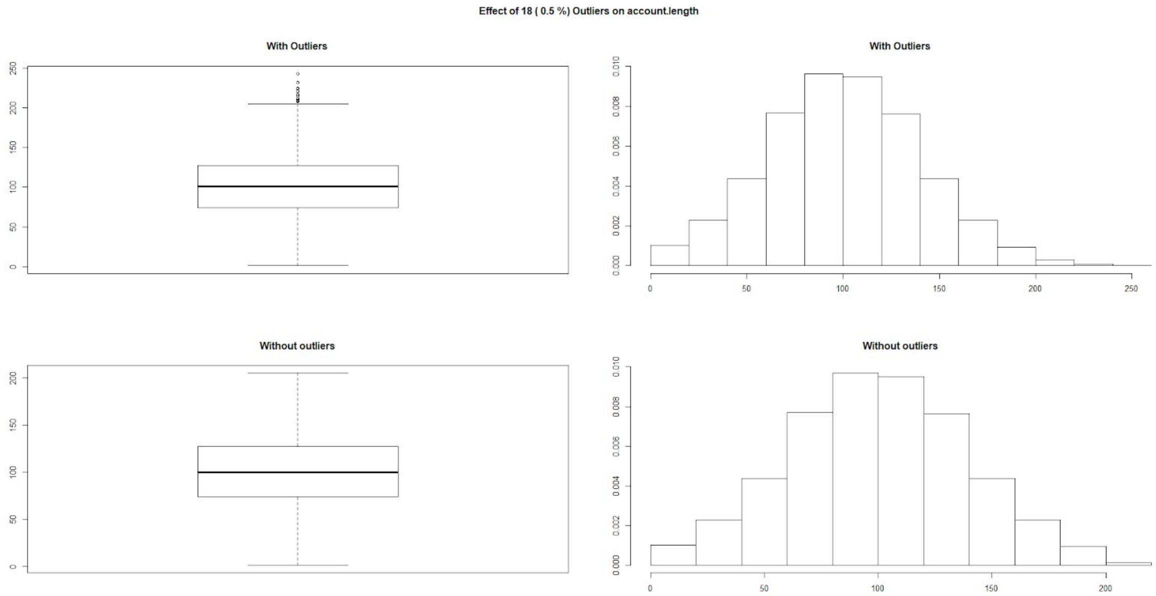
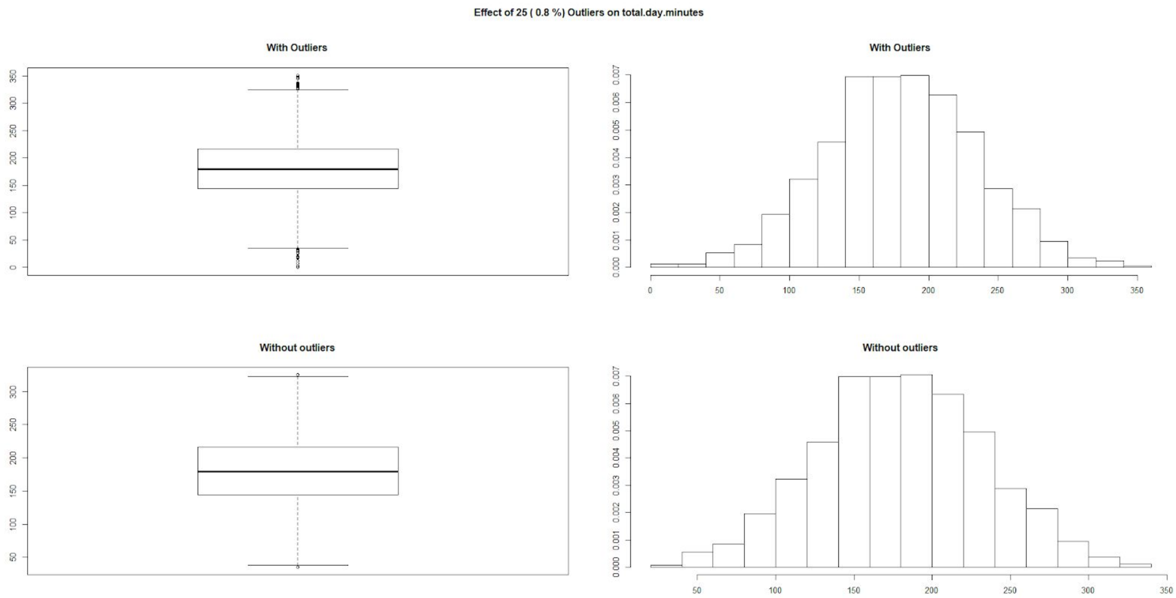
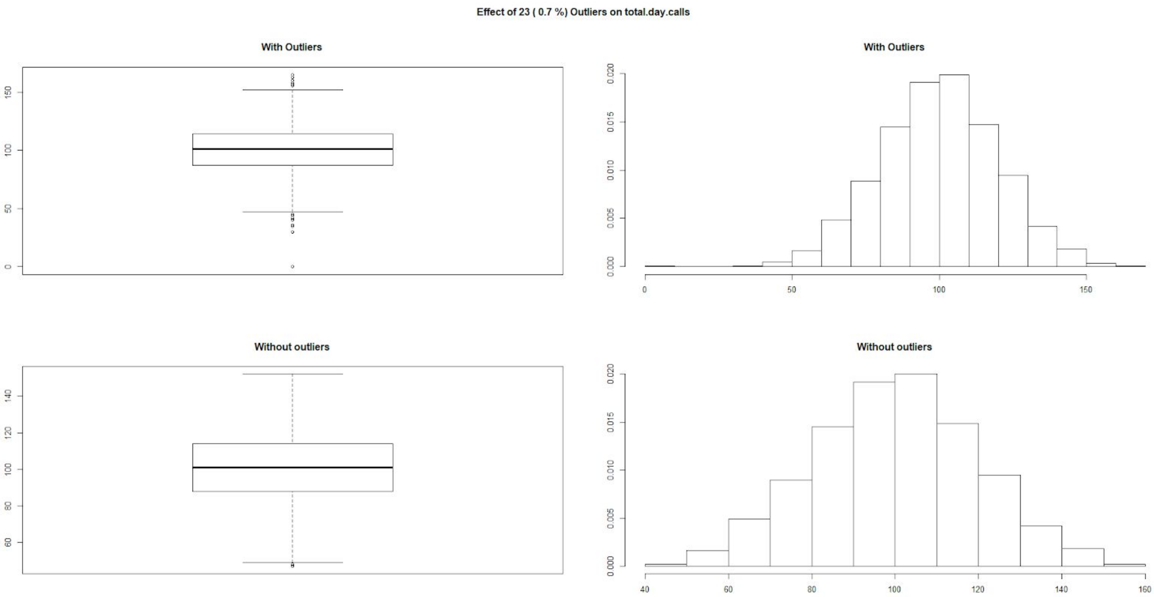
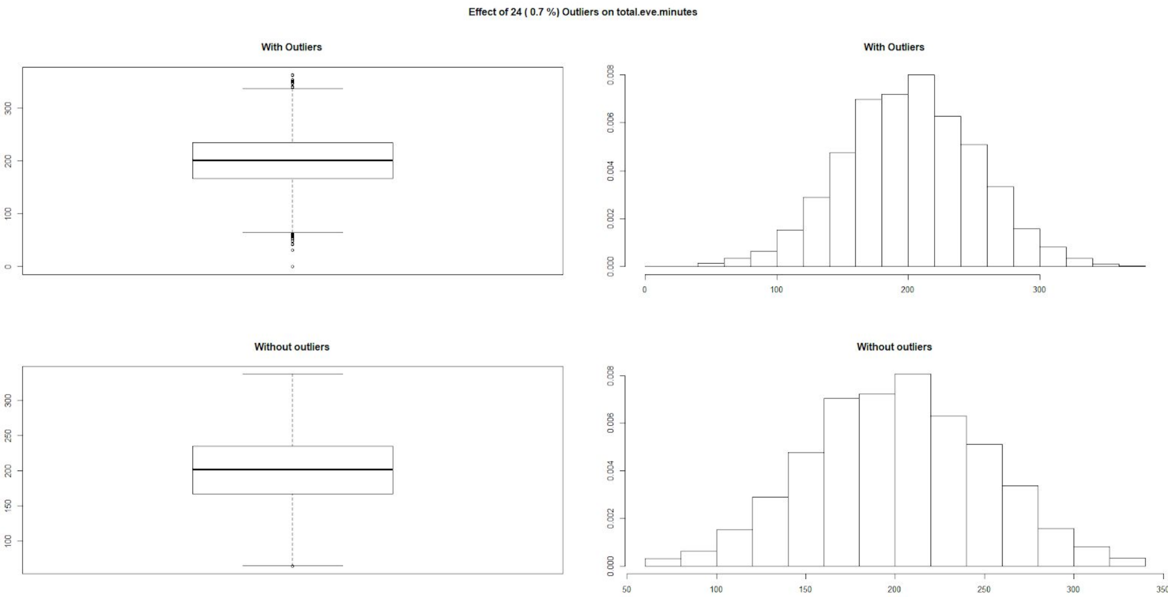


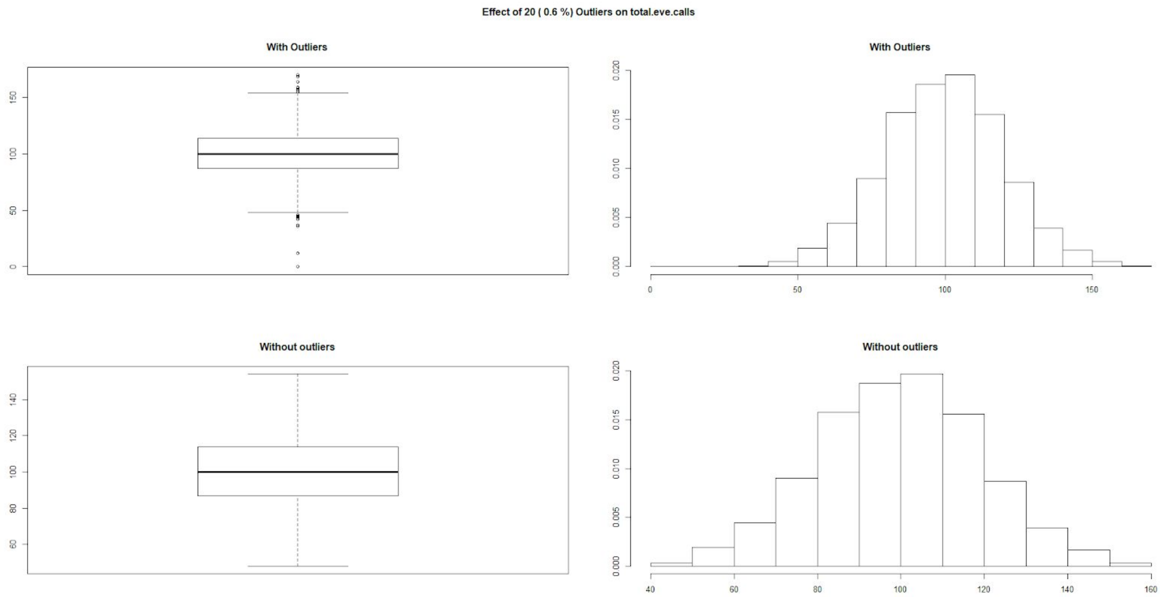
Figure 4.1: Histogram of predictor variables(See R code in Appendix)

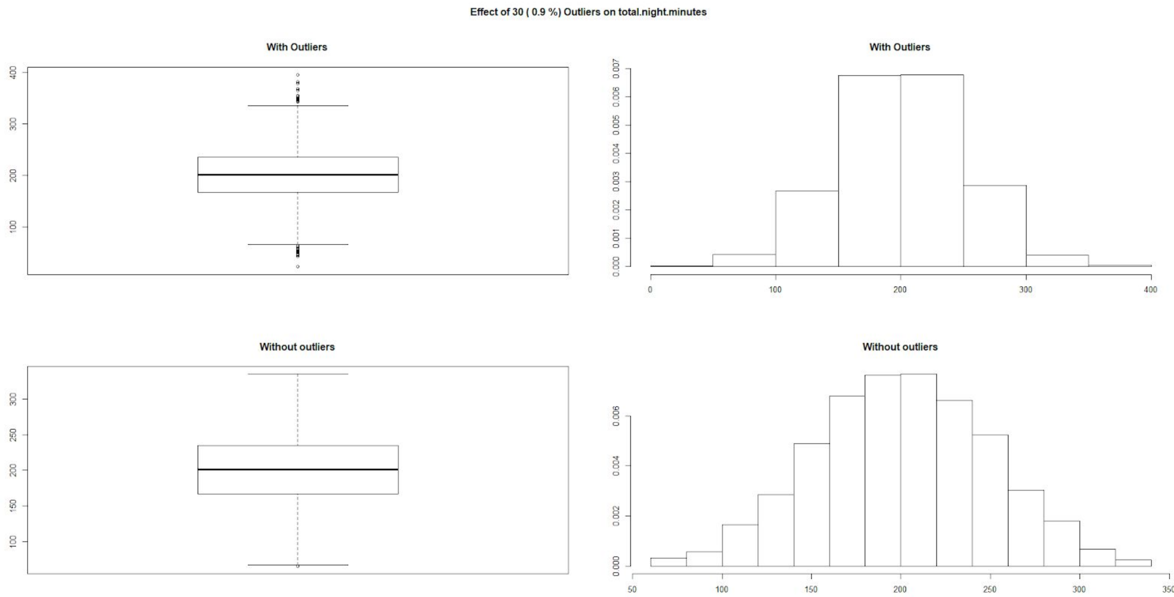

Effect of 18 (0.5 %) Outliers on account.length
With Outliers
With Outliers
Without outliers
Without outliers

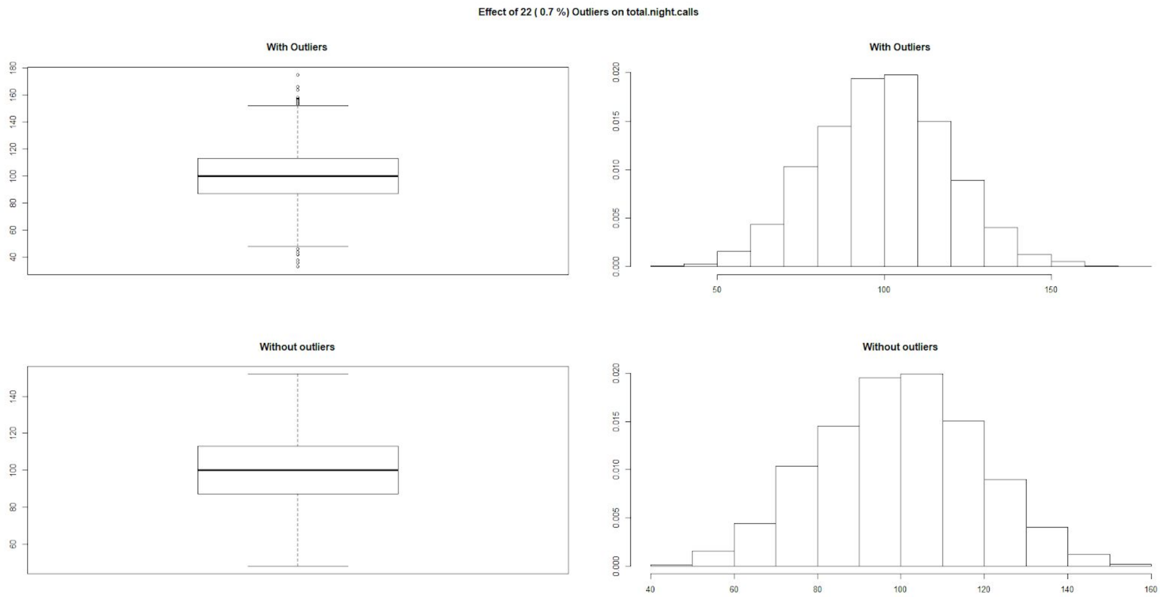

Effect of 25 (0.8 %) Outliers on total.day.minutes
With Outliers
With Outliers
Without outliers
Without outliers

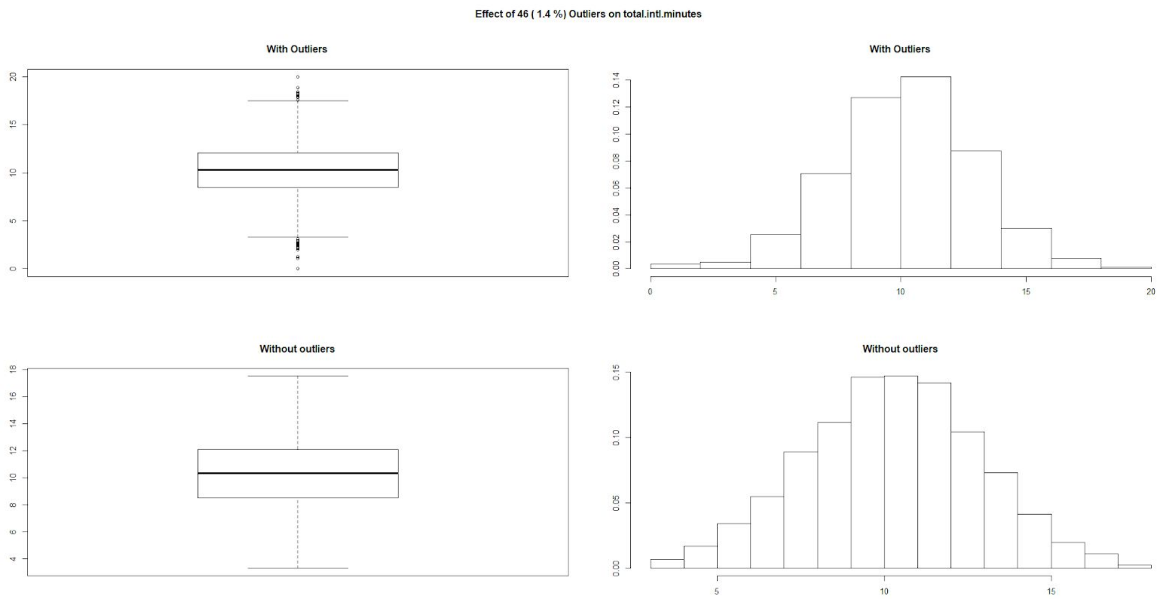

Effect of 23 (0.7 %) Outliers on total.day.calls
With Outliers
With Outliers
Without outliers
Without outliers


Effect of 24 (0.7 %) Outliers on total.eve.minutes
With Outliers
With Outliers
Without outliers
Without outliers


Effect of 20 (0.6 %) Outliers on total.eve.calls
With Outliers
With Outliers
Without outliers
Without outliers


Effect of 30 (0.9 %) Outliers on total.night.minutes
With Outliers
With Outliers
Without outliers
Without outliers


Effect of 22 (0.7 %) Outliers on total.night.calls
With Outliers
With Outliers
Without outliers
Without outliers


Effect of 46 (1.4 %) Outliers on total.intl.minutes
With Outliers
With Outliers
Without outliers
Without outliers

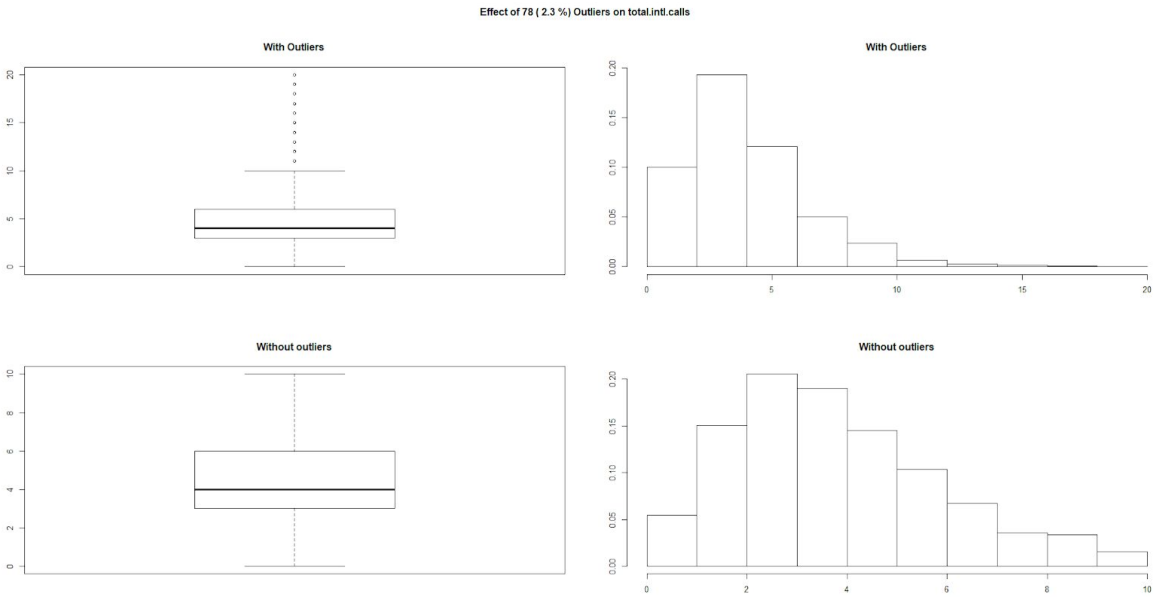


Figure 4.2.1-4.2.10: Effects of outliers

Appendix B – R Code

Histogram of predictors(Figure 4.1):

```
for (i in 1:length(cnames)){
  assign(paste0("hist",i),ggplot(reduced_train,aes_string(cnames[i])) +
        geom_histogram(binwidth = 0.5)+
        stat_bin(bins = 20)+
        ggtitle(paste("Histogram of ",cnames[i])))
}
gridExtra::grid.arrange(hist1,hist2,hist3,hist4,hist5,hist6,hist7,hist8,hist9,hist10,ncol=4)
```

Predictor boxplots(Figure 2.1.3.3):

```
numeric_index <- sapply(reduced_test,is.numeric)
numeric_data <- reduced_test[,numeric_index]
cnames <- colnames(numeric_data)
for (i in 1:length(cnames)){
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]),x="Churn"), data =     reduced_train)+
        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])+
        ggtitle(paste("Box plot of",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn3,gn4,gn5,gn6,gn7,gn8,gn9,gn10,gn11,gn12,gn13,gn14,gn15,gn16,nc
ol=4)
```

Effect of outliers(Figure 4.2.1-4.2.9):

```
outToNa <- function(vl, df) {
 for (i in vl) {
   outlier <- boxplot.stats(df[, i])$out
   df[, i] <- ifelse(df[, i] %in% outlier,
                 NA, df[, i])
 }
 return(df)
}
outliereff <- function(i, df) {
   total = length(df[, i])
   par(mfrow = c(2, 2), oma = c(0, 0, 3,
                        0))
   boxplot(df[, i], main = "With Outliers")
   hist(df[, i], main = "With Outliers",
      xlab = NA, ylab = NA, prob = TRUE)
   df <- outToNa(i, df)
   boxplot(df[, i], main = "Without outliers")
   hist(df[, i], main = "Without outliers",
      xlab = NA, ylab = NA, prob = TRUE)
   out <- sum(is.na(df[, i]))
   per <- round((out)/total * 100, 1)
   title(paste("Effect of", out, "(", per,
          "%)", "Outliers on", colnames(df)[i],
          sep = " "), outer = TRUE)
```

```
}
varlist <- list(2,5,6,7,8,9,10,11,12,13)
for (i in varlist) {
  outliereff(i,reduced_train)
}
```

Correlation Plot(Figure 2.1.4):

```
numeric_index <- sapply(train,is.numeric)
numeric_data <- train[,numeric_index]
corrgram(numeric_data, order = F,upper.panel = panel.pie, text.panel = panel.txt, main =
"Correlation Plot")
```

Complete R Code:

```
## Loading necessary library
library(dplyr)
library(gridExtra)
library(corrgram)
library(ggplot2)
library(DMwR)
library(caret)
library(C50)
library(e1071)
library(rpart)
library(randomForest)
library(inTrees)
```

```
## Loading Data
train <- read.csv("Train_data.csv")
test <- read.csv("Test_data.csv")

## Converting data frame
train$area.code = as.factor(train$area.code)
test$area.code = as.factor(test$area.code)

## Boxplot and Outlier analysis
outToNa <- function(vl, df) {
 for (i in vl) {
   outlier <- boxplot.stats(df[, i])$out
   df[, i] <- ifelse(df[, i] %in% outlier,
                  NA, df[, i])
 }
 return(df)
}
for (i in 1:length(cnames)){
 assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]),x="Churn"), data = train)+
        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])+
        ggtitle(paste("Box plot of",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn3,gn4,gn5,gn6,gn7,gn8,gn9,gn10,gn11,gn12,gn13,gn14,gn15,gn16,nc
ol=4)
train <- outToNa(n_position, train)
train <- knnImputation(train, k=3)
test <- outToNa(n_position, test)
```

```
test <- knnImputation(test, k=3)

gridExtra::grid.arrange(gn1,gn3,gn4,gn5,gn6,gn7,gn8,gn9,gn10,gn11,gn12,gn13,gn14,gn15,gn16,ncol=4)


## Correlation Plot

numeric_index <- sapply(train,is.numeric)

numeric_data <- train[,numeric_index]

corrgram(numeric_data, order = F,upper.panel = panel.pie, text.panel = panel.txt, main = "Correlation Plot")


## Chi Square test

factor_index <- sapply(train, is.factor)

factor_data <- train[,factor_index]

for (i in 1:4){

  print(names(factor_data)[i])

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

}



## Dimensionality Reduction

reduced_train <- subset(train,select = -c(total.day.charge, total.eve.charge, total.night.charge, total.intl.charge, phone.number, area.code, number.vmail.messages))

reduced_test <- subset(test,select = -c(total.day.charge, total.eve.charge, total.night.charge, total.intl.charge, phone.number, area.code,  number.vmail.messages))


## Feature Scaling

normalized_train <- reduced_train

normalized_test <- reduced_test

numeric_index <- sapply(reduced_train,is.numeric)

numeric_data <- reduced_train[,numeric_index]

cnames <- colnames(numeric_data)

for (i in cnames){
```

```
  normalized_train[,i] = (normalized_train[,i] - min(normalized_train[,i]))/
                      (max(normalized_train[,i])-min(normalized_train[,i]))
  normalized_test[,i] = (normalized_test[,i] - min(normalized_test[,i]))/
    (max(normalized_test[,i])-min(normalized_test[,i]))
}

## Use C5.0 Decision Tree Model
set.seed(1234)
c50_model <- C5.0(Churn ~.,normalized_train, trials = 100, rules = T)
summary(c50_model)
test_predict <- predict(c50_model,normalized_test[,-14],type = "class")
xtab <- table(observed = normalized_test[,14], predicted = test_predict)
confusionMatrix(xtab)

## Use CART DT Model
rpart_model <- rpart(Churn~.,data = normalized_train,method = "class")
pred <- predict(rpart_model,normalized_test[,-14],type = "class")
xtab <- table(observed = normalized_test[,14], predicted = pred)
confusionMatrix(xtab)

## Use Random Forest with 500 trees
randomforest_model <- randomForest(Churn~.,normalized_train,importance=T, ntree=500)
randomforest_model
pred <- predict(randomforest_model,normalized_test[,-14])
xtab <- table(observed = normalized_test[,14], predicted = pred)
confusionMatrix(xtab)
```

References:

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 6. Springer.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer Science & Business Media

"The Comprehensive R Archive Network". Retrieved 2018-08-06.

Chatfield, C. (1995). *Problem Solving: A Statistician's Guide* (2nd ed.). Chapman and Hall. ISBN 0412606305