Predicting Customer Churn

Vrushabhkumar S. Jain

11 February 2019

Contents

Chapter 1

Introduction

1.1Problem Statement

Customer churn occurs when customers or subscribers stop doing business with a company or service, also known as customer attrition. It is also referred as loss of clients or customers. One industry in which churn rates are particularly useful is the telecommunications industry, because most customers have multiple options from which to choose within a geographic location. 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 and the objective of this case is to predict customer behaviour whether he will churn or not.

1.2 Data

Our task is to build classification models which will classify the behaviour of customer whether he will churn or not based on the predictors. The predictors provided are as follows:

State – States of USA where customer lives
Account length - How long account has been active in days
International plan - Whether customer has opted for international plan or not
Voicemail plan - Whether customer has opted for voicemail plan or not
Number of voicemail messages - No .of voicemail messages he used
Total day minutes - Total day minutes he used
Total day calls - Total day calls he made
Total day charge - Total day charges he paid
Total evening minutes - Total evening minutes he used
Total evening calls - Total evening calls he made
Total evening charge - Total evening charges he paid
Total night minutes - Total night minutes he used
Total night calls - Total night calls he made
Total night charge - Total night charges he paid
Total international minutes - Total international minutes he used
Total international calls - Total international calls he made
Total international charge - Total international charges he paid
Number of customer service calls - No. of customer service calls he made

Given below is a sample of the dataset that we are using to predict the behaviour of customer whether he will churn or not:

Table 1.1: Customer Churn Sample Data (Columns:1:9)

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

Table 1.2: Customer Churn Sample Data (Columns:10:19)

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
45.07	197.4	99	16.78	244.7	91	11.01	10	3	2.7
27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7
41.38	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29
50.9	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78
28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73
37.98	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7

Table 1.3: Customer Churn Sample Data (Comluns: 20:21)

number customer service calls	Churn
1	False.
1	False.
0	False.
2	False.
3	False.
0	False.

Chapter 2

Methodology

2.1 Pre Processing

Data pre-processing is crucial in any data mining process as they directly impact success rate of the project. This reduces complexity of the data under analysis as data in real world is unclean. Data is said to be unclean if it is missing attribute, attribute values, contain noise or outliers and duplicate or wrong data. Presence of any of these will degrade quality of the results. This is often called as Exploratory Data Analysis. In this process we will try to look at the relation between independent variable and target variable with the train dataset through visualization and will decide the further analysis according to it.
Fig. 2.1 shows the customer churn ratio.

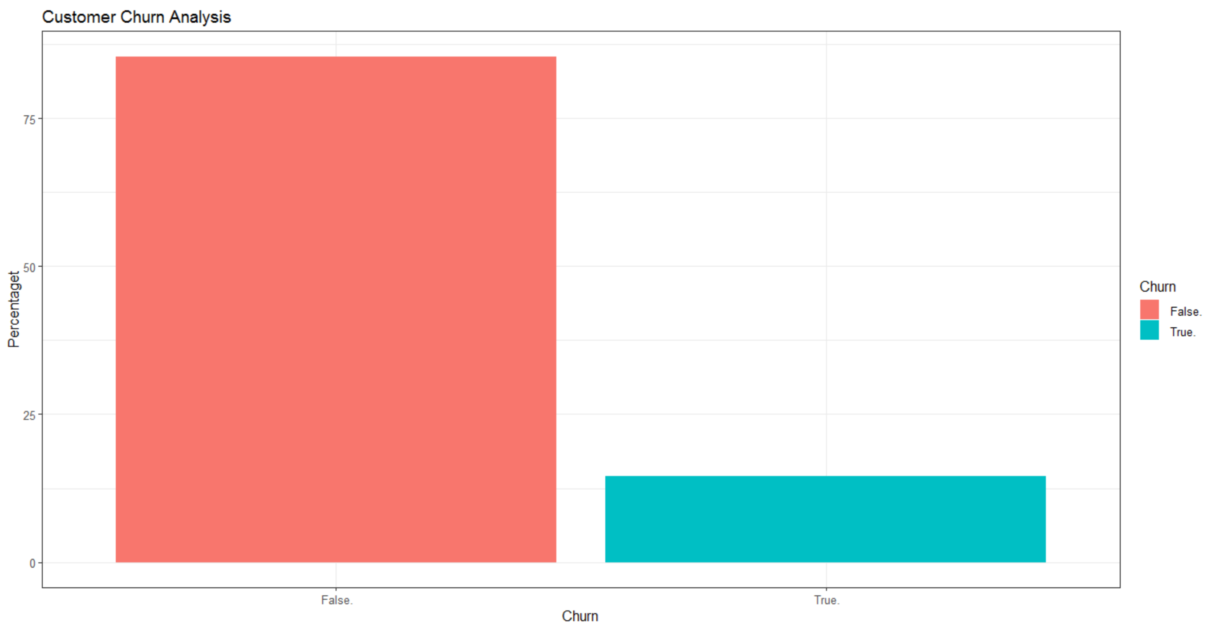


Fig.2.1 Customer Churn Ratio

As per data shown in fig.2.1, about 16.9 % customers are churned and remaining customer continued the service.

Now let’s look at the relation between categorical variable and target variable. Fig 2.2 shows the distribution of customer in different state of USA with their behaviour whether they churned or not.

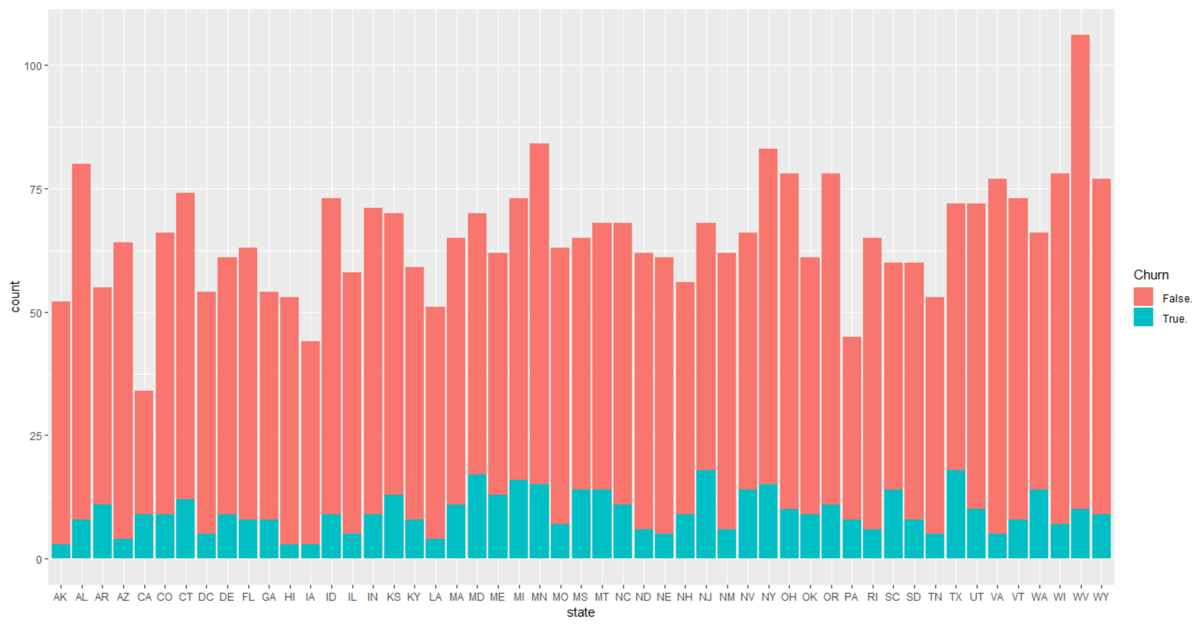


Fig.2.2 Churn ratio with respect to states

From above fig. it can be seen that, most of the customer are from from west Virginia(WV) followed by Minnesota(MN), New York(NY), Alabama(AL) etc. also churn rate is more in Texas(TX) followed by New Jersy(NJ), Maryland(MD) etc.
Similarly, following fig 2.3 also shows the relation between different categorical attribute with customer behaviour.

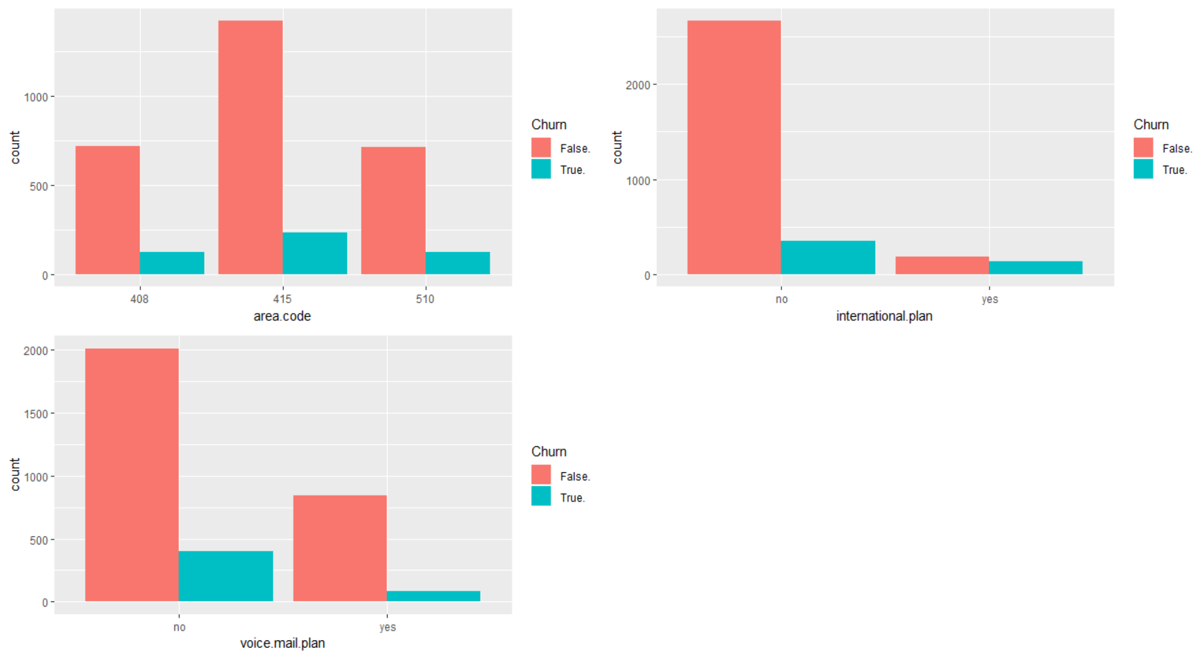


Fig.2.3 Categorical variables vs Churn Ratio

From the above fig. following conclusions can be made:

i. Most of the customers are from area code 415 and churn rate is also more in that area as well as customer churn is almost similar in all area. Looks like area code has no relation with customer behaviour.
ii. Most of the customer didn’t opt for international plan as well as churn ratio is also less as compared to other who opted for international plan. It seems that customer are

not happy with international plan schemes, effective planning should be made to reduce customer churn due to international plan.

iii. Also, ratio of customer churn between the customers who opted for voicemail plan and those who didn't opt for is almost same, looks like customer has no issues with voicemail plan.

As we know most of the analysis gives best result with normal distribution of continuous variables. We can visualize that in a glance by looking at the probability distributions or probability density functions of the continous variable.
In Figure 2.4 we have plotted the probability density functions of all the customer attribute we have available in the data. The blue lines indicate Kernel Density Estimations (KDE) of the variable. The red lines represent the normal distribution. So as you can see in the figure most variables either very closely or somewhat imitate the normal distribution.

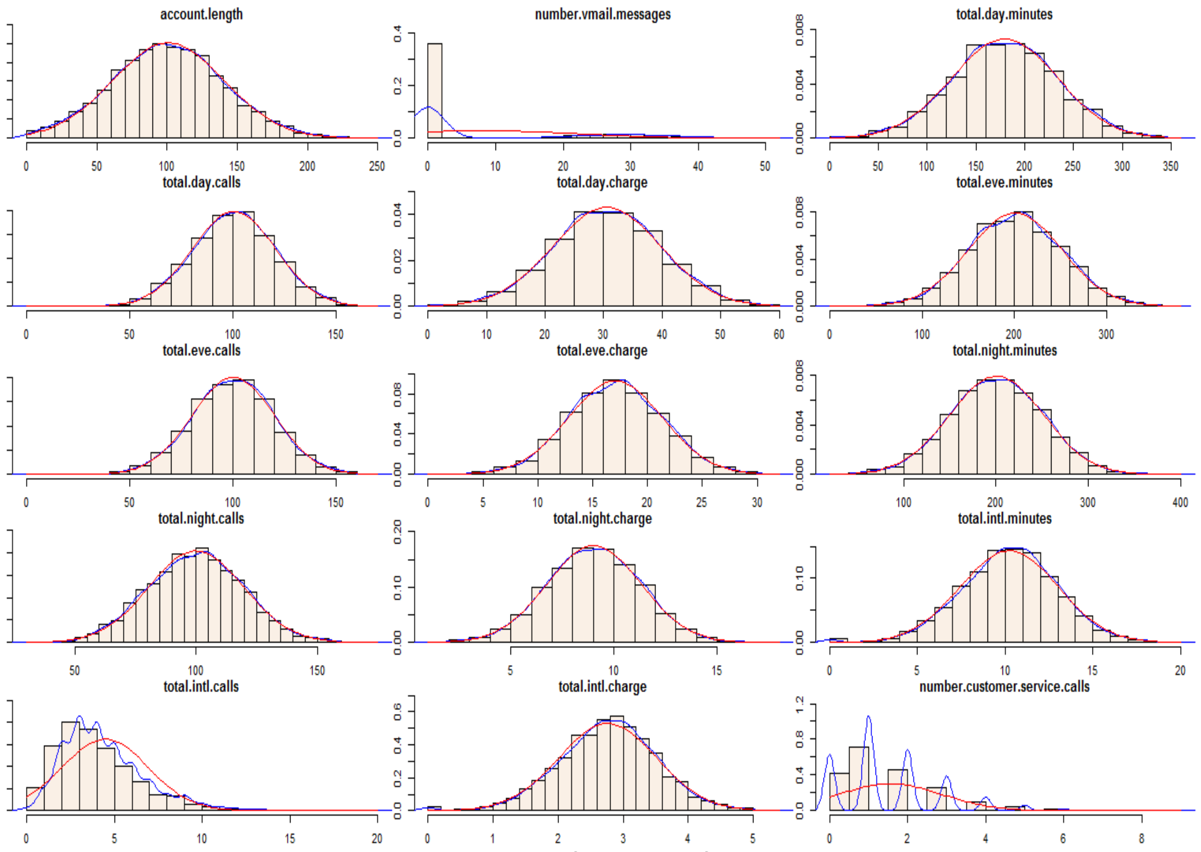


Fig.2.4: Probability density function of Customer Churn Data

As we can see some of the variables do not follow normal distribution. Further we will do some pre-processing like missing value imputation, outlier analysis et. And will try to make the distribution of these variable normal.

2.1.1 Missing value analysis

After checking for the missing value in our dataset, we found that there are no missing value in our dataset. But if found following are the strategies to deal with missing value

Omitting the missing value columns
If missing value in a particular column is more than 30%, it is advisable to eliminate those as most of the data will be based on assumption, it may not relate to real world situations
Following are the methods of imputing the missing values:

i. Mean method

In this method, missing value is imputed with mean of continuous variable.

ii. Median method

In this method, missing value is imputed with median of continuous variable.

iii. Mode method

In this method, missing value is imputed with maximum occurring value.

iv. KNN Imputation

In this method, missing value is imputed with KNN imputed imputation based on Euclidean or Manhattan distance formula. This method can be used for both continuous and categorical variable.

v. Prediction method

In this method missing value is imputed with machine learning algorithms.

2.1.2 Outlier Analysis

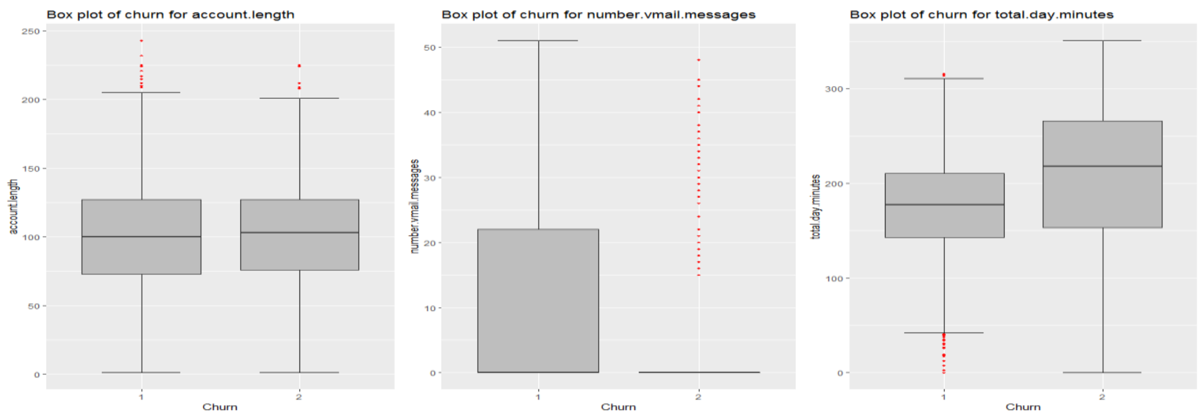
Several definitions of outlier have been presented in the data mining literature. two of them:

i. An outlier is defined as a data point which is very different from

the rest of the data based on some measure and

ii. An outlier is a case that does not follow the same model as the rest of the data and appears as though it comes from a different probability distribution.

We can clearly observe from fig. 2.4 the probability distributions that most of the variables are skewed, for e.g. total international calls, no. of customer service calls, no of vmail messages etc. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. One of the other steps of pre-processing apart from missing value analysis is the presence of outliers. In this case we use a classic approach of removing outliers, Tukey's method. We visualize the outliers using boxplots.
In figure 2.5 we have plotted the boxplots of the 15 predictor variables with respect to customer behaviour whether he churned or not. A lot of useful inferences can be made from these plots. First as you can see, we have outliers and extreme values in each of the data set.



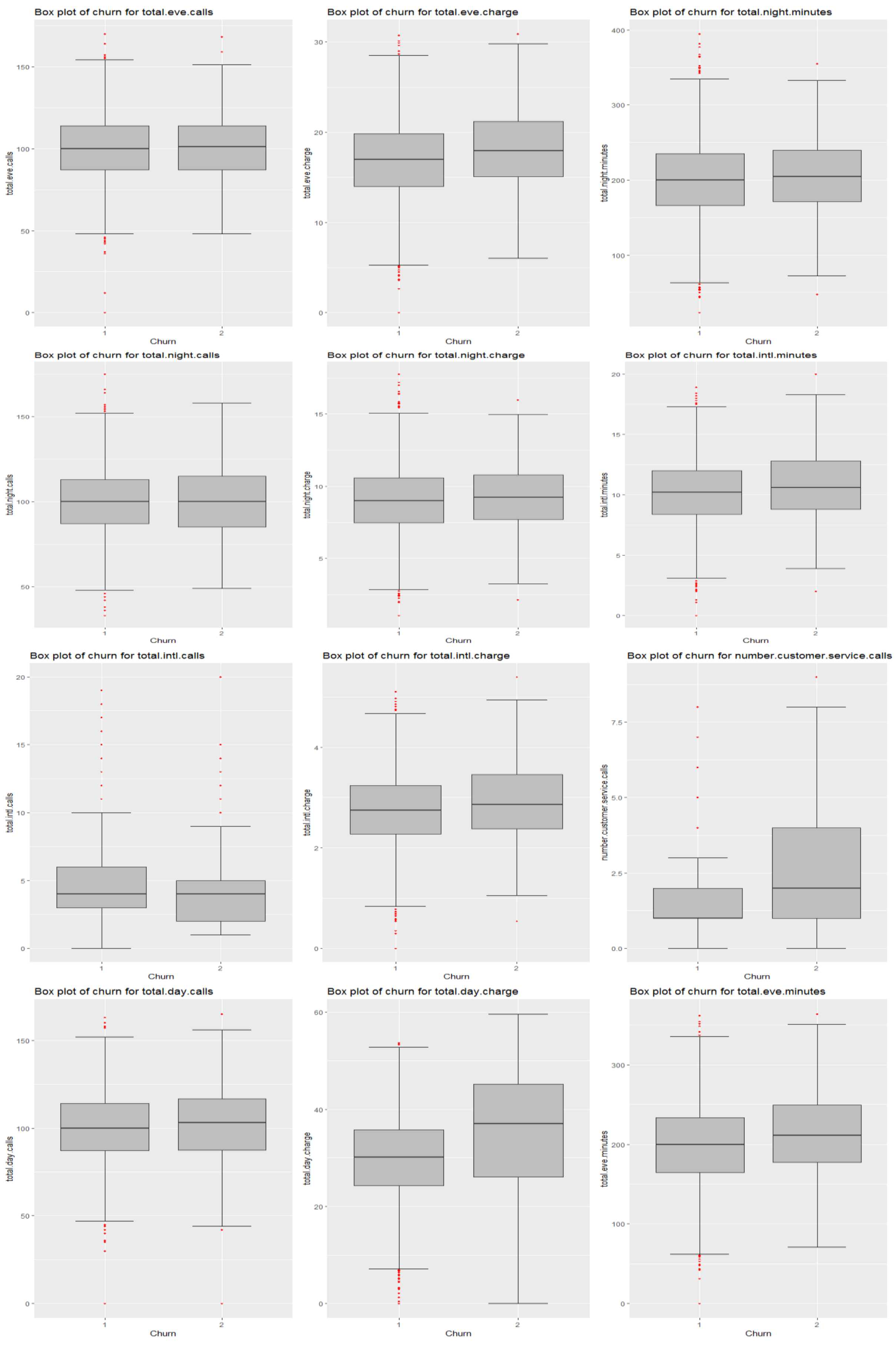
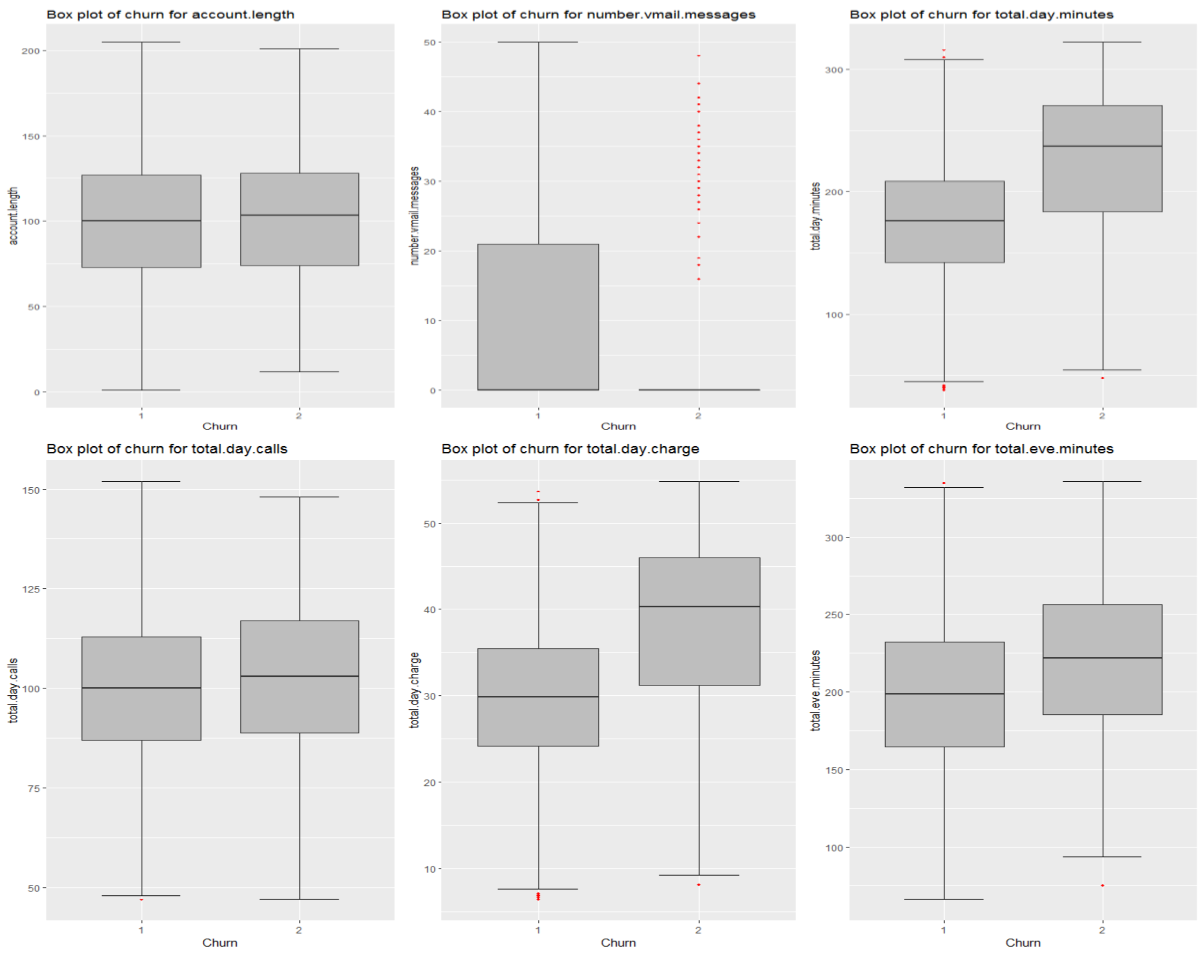


Fig2.5 Custmer behaviour vs Predictors for customer churn data

In above box plot, along x-axis, 1- Not Churn and 2- Churn.
Other useful interference can also be made from these plots. For e.g if you compare total day charge with customer behaviour, median value of total day charge is more for customer who have churned, this might be the reason behind customer churn.
Based on above plots, following strategies should be made to reduce the customer churn:

i. As we can see customer who churned have made more calls and used more call minutes and relatively their charges are also high, some attractive offers should be made for these high service users' customers in terms of reducing the charges or made some special plan based on their service usage pattern.
ii. Also it can be seen that customer who have churned has made more customer service calls before they churned, but they might didn't get proper response, that might be the reason behind their churn.

Following is the boxplot visualisation after outlier removal.



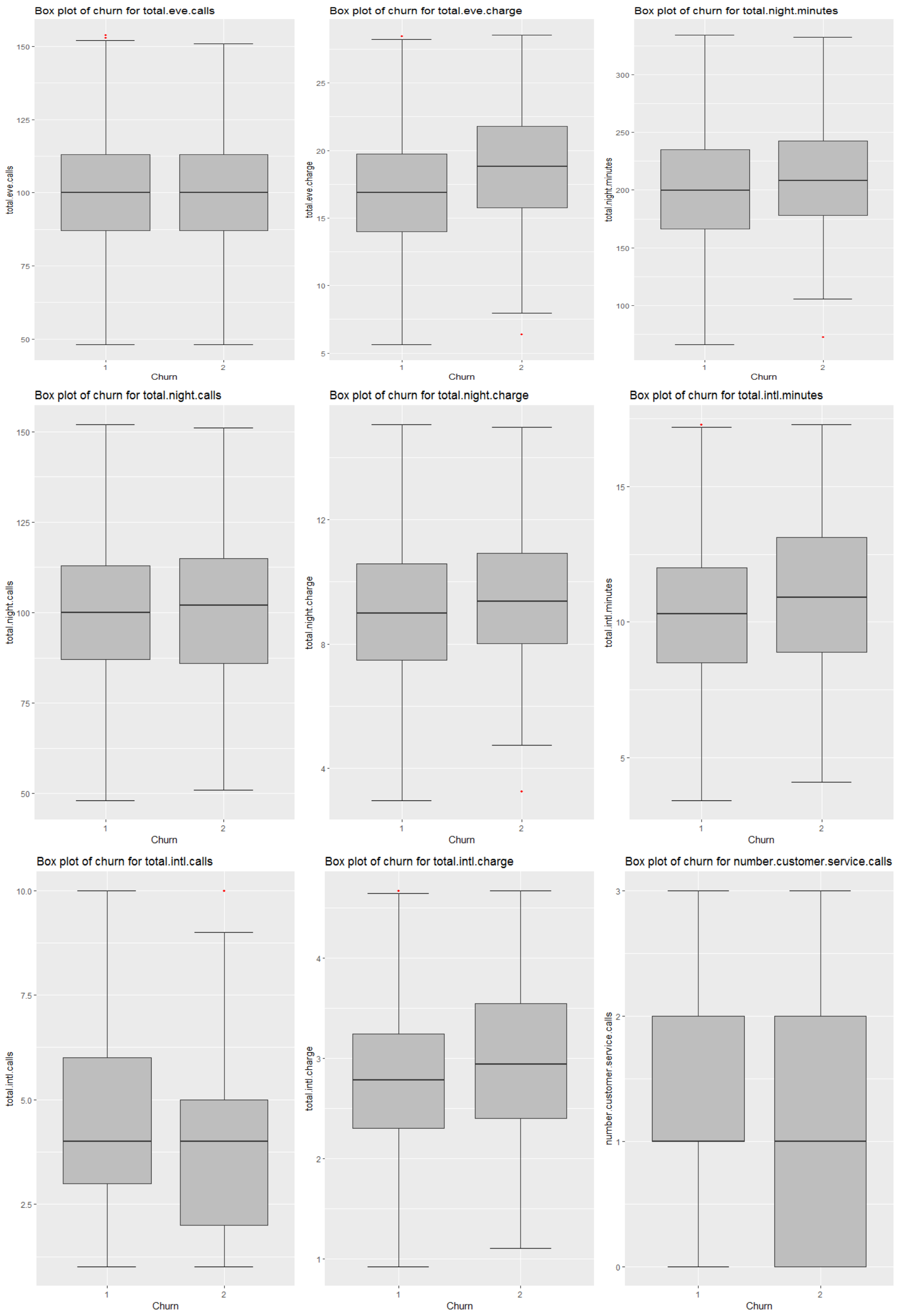


Fig2.6 Customer behaviour vs Predictors for customer churn data after outlier removal

As we can see in the boxplot before and after removal of outlier;variable no. of vmail messages has maximum outlier and we left with only 0 values after outlier removal and also in fig.2.4 we can see that this variable is far displaced from normal distribution curve, also it does not give significant information to predict customer behaviour, hence better to drop this variable.
Apart from these, You can see in figure 2.7 what effect outliers have on the normalisation of our data. Similar graphs can be seen in appendix.

Effect of 18 (0.54%) of outlier on Account Length

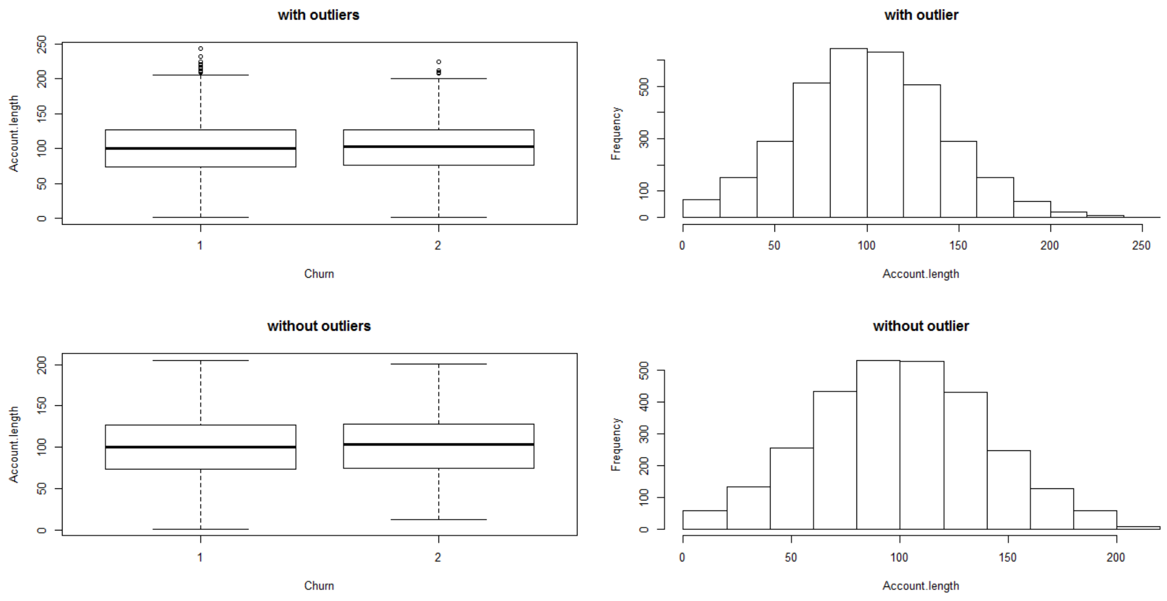


Fig.2.7: Effect of Outliers on Predictor Variables of customers churn data

2.1.3 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. We have used correlation map and chi-square test to perform features selection. From below correlation plot, extreme blue colour indicates highly positively correlated variable, we will eliminate any one variable from a pair of correlated variable.

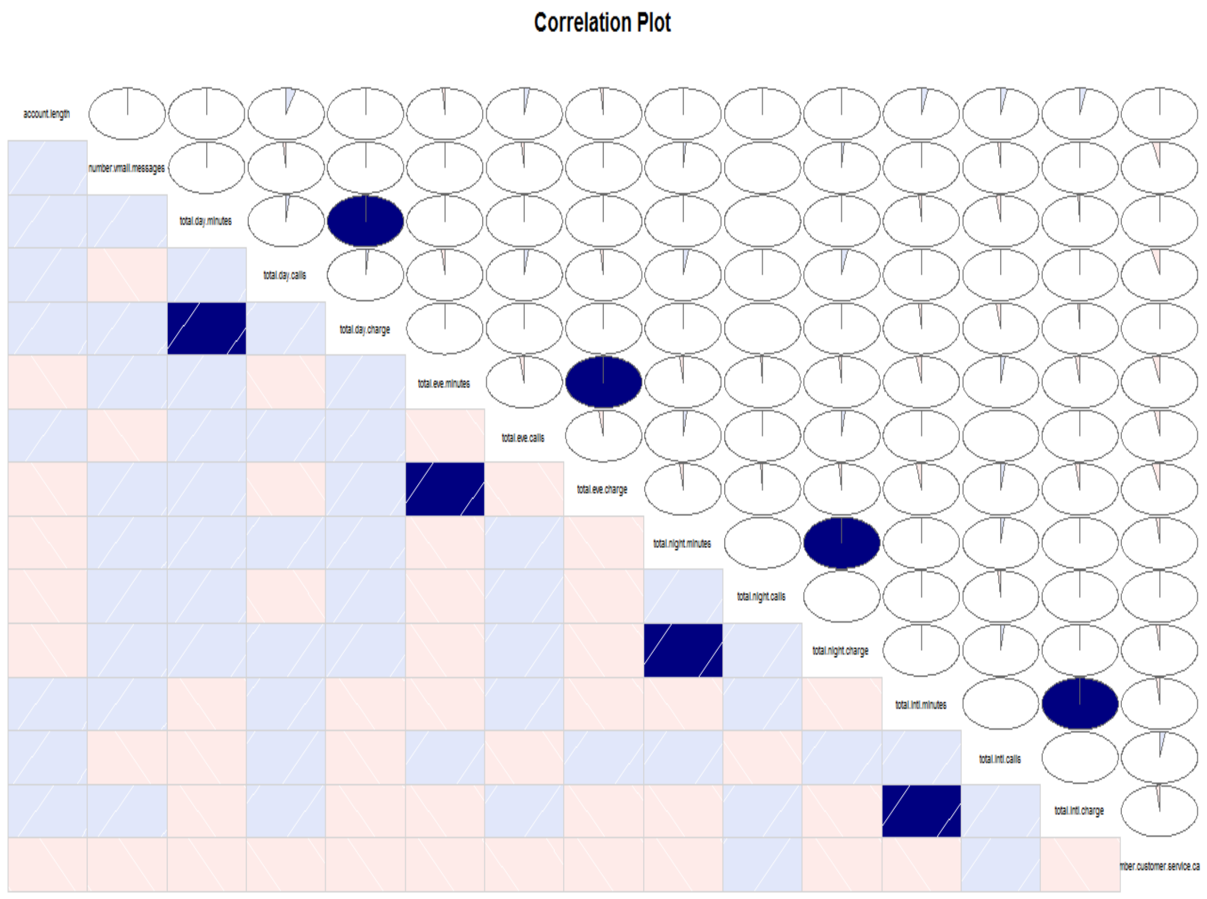


Fig.2.8:Correlation Plot

As correlation analysis is valid only for continuous variable, we have performed chi-square test to determine correlation between categorical variable and target variable. The criteria to eliminate particular variable is the p-value, the variable having p-value more than 0.05 is not significant and is eliminated.
Following table shows the variable and its corresponding p-value

Table2.1: Chi-square test result

Categorical variable	p-value	Significant?
State	0.02074	Yes
Area.code	0.6972	No
International.plan	2.2e-16	Yes
Voice.mail.plan	2.644e-07	Yes

Now, after dimensionality reduction we are left with 14 independent variable and one dependent variable.

2.1.2 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

The motivation behind feature scaling is 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.

Following are the method to perform feature scaling:

i. Standardization/ z- score

It requires data to be normally distributed. It performs scaling based on below formula:

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

Where,

z = difference between raw score and population mean in the units of standard deviation.

x = observation

μ = mean of population

σ = standard deviation of population

ii. Normalization

It does not require data to be normally distributed. It scales values between the range of 0-1. It performs scaling based on below formula:

$$New\ Value = \frac{Value - min(Value)}{max(Value) - min(Value)}$$

After the outlier removal we have got fig showing distribution of continuous variable:

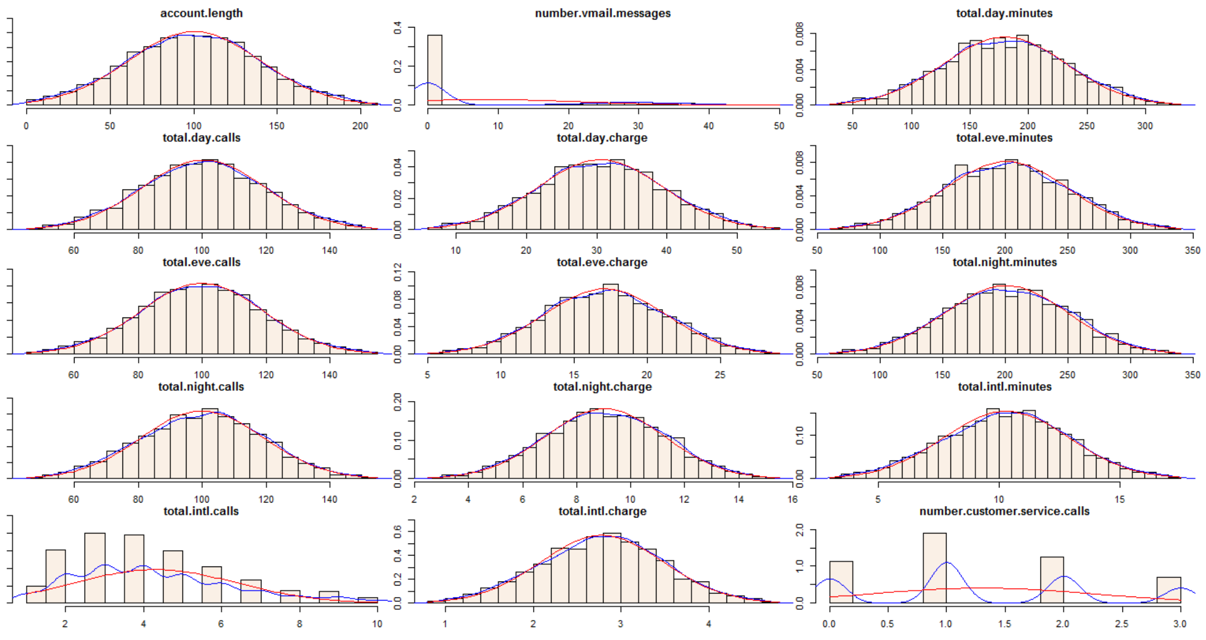


Fig.2.9: probability density function of customers churn data after outlier removal

As we can see, some of the variable like total international calls, number.customer.service.calls etc. are not normally distributed we will be using normalization approach for feature scaling.

After performing all these preprocessing technique our data is now ready to build model on top of it for which we are going to split our dataset into train and test with 80:20 ratio.

2.2 Modelling

2.2.1 Model Selection

After performing pre-processing technique first step in model building is the selection of model for a particular problem. As our target variable "Churn" falls in the category of nominal, we can use classification approach for the predictive analysis.

We always start our model building from the most simplest to more complex. Therefore we are using Decision tree followed by Random Forest, Logistic Regression, KNN algorithm, Naïve Bayes algorithm.

2.2.2 Decision tree

Following is the summary of decision tree:

```
Call:
C5.0.formula(formula = Churn ~ ., data = train, trials = 50, rules = TRUE)

Rule-Based Model
Number of samples: 2239
Number of predictors: 14

Number of boosting iterations: 50
Average number of rules: 20

Non-standard options: attempt to group attributes

Attribute usage:

        100.00% state
        100.00% international.plan
        100.00% voice.mail.plan
        100.00% total.day.minutes
        100.00% total.eve.minutes
        100.00% total.night.minutes
        100.00% total.intl.minutes
         99.96% total.intl.calls
         99.87% total.night.calls
         99.51% account.length
         99.33% total.day.calls
         98.12% total.eve.calls
         97.50%  number.vmail.messages
         69.85%  number.customer.service.calls
```

Confusion matrix:

Confusion Matrix and Statistics

```
predicted
actual   1   2
     1 496   2
     2  17  43

            Accuracy : 0.9659
              95% CI : (0.9473, 0.9794)
 No Information Rate : 0.9194
 P-Value [Acc > NIR] : 5.273e-06

               Kappa : 0.8007
Mcnemar's Test P-Value : 0.001319

         Sensitivity : 0.9669
         Specificity : 0.9556
      Pos Pred Value : 0.9960
      Neg Pred Value : 0.7167
          Prevalence : 0.9194
      Detection Rate : 0.8889
Detection Prevalence : 0.8925
   Balanced Accuracy : 0.9612

    'Positive' Class : 1
```

Using 50 decision trees we have got the accuracy upto 96.59% .

2.2.2 Random Forest

Next model we are building is the random forest. The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

Following is the result of Random Forest:

```
Call:
 randomForest(formula = Churn ~ ., data = train, importance = TRUE,      ntree = 250)
               Type of random forest: classification
                     Number of trees: 250
No. of variables tried at each split: 3
Confusion Matrix and Statistics

      predicted
```

```
actual   1   2
     1 496   2
     2  22  38

               Accuracy : 0.957
                 95% CI : (0.9367, 0.9723)
    No Information Rate : 0.9283
    P-Value [Acc > NIR] : 0.0034487

                  Kappa : 0.7374
 Mcnemar's Test P-Value : 0.0001052

            Sensitivity : 0.9575
            Specificity : 0.9500
         Pos Pred Value : 0.9960
         Neg Pred Value : 0.6333
             Prevalence : 0.9283
         Detection Rate : 0.8889
   Detection Prevalence : 0.8925
      Balanced Accuracy : 0.9538

       'Positive' Class : 1
```

With the random forest using 250 decision tress, we have got the accuracy of 95.7%.

2.2.3 Logistic regression

Logistic Regression is one of the basic and popular algorithm to solve a classification problem. It is named as '*Logistic Regression*', because it's underlying technique is quite the same as Linear *Regression*. The term "*Logistic*" is taken from the *Logit* function that is used in this method of classification.
Following is the summary of logistic regression:

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9793  -0.3907  -0.2150  -0.0983   3.5588

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -12.04193    1.37890  -8.733  < 2e-16 ***
state2                         0.92334    1.25483   0.736 0.461832
state3                         1.48644    1.32102   1.125 0.260496
state4                         1.48392    1.32547   1.120 0.262908
state5                         2.18591    1.29579   1.687 0.091617 .
state6                         1.87379    1.25770   1.490 0.136261
state7                         1.43707    1.23018   1.168 0.242735
state8                         1.81748    1.29997   1.398 0.162085
```

```
state9                    1.16661    1.25709   0.928 0.353396
state10                   1.37776    1.25320   1.099 0.271595
state11                   0.53746    1.45597   0.369 0.712023
state12                   1.02238    1.40577   0.727 0.467058
state13                   1.09383    1.52997   0.715 0.474651
state14                   1.98223    1.22627   1.616 0.105994
state15                   0.01686    1.31669   0.013 0.989781
state16                   0.83646    1.28191   0.653 0.514069
state17                   2.24686    1.19148   1.886 0.059326 .
state18                   1.42585    1.28604   1.109 0.267551
state19                   2.22325    1.34932   1.648 0.099417 .
state20                   2.16999    1.20731   1.797 0.072276 .
state21                   2.29291    1.20865   1.897 0.057818 .
state22                   2.36599    1.22128   1.937 0.052709 .
state23                   1.78153    1.21280   1.469 0.141851
state24                   2.05165    1.18295   1.734 0.082858 .
state25                   1.56804    1.27285   1.232 0.217982
state26                   1.86622    1.23771   1.508 0.131605
state27                   2.15408    1.25485   1.717 0.086053 .
state28                   1.84952    1.21011   1.528 0.126416
state29                   0.56596    1.29415   0.437 0.661879
state30                   1.62014    1.22526   1.322 0.186077
state31                   2.07469    1.25436   1.654 0.098130 .
state32                   2.64189    1.17258   2.253 0.024256 *
state33                   0.47302    1.39011   0.340 0.733650
state34                   1.86708    1.20456   1.550 0.121139
state35                   1.77626    1.23572   1.437 0.150595
state36                   1.83419    1.22642   1.496 0.134768
state37                   1.55882    1.27660   1.221 0.222058
state38                   1.30847    1.26185   1.037 0.299759
state39                   1.35546    1.33131   1.018 0.308613
state40                   1.06210    1.28353   0.827 0.407965
state41                   2.75102    1.24467   2.210 0.027088 *
state42                   1.65942    1.25072   1.327 0.184587
state43                   0.86345    1.30836   0.660 0.509285
state44                   2.67411    1.21460   2.202 0.027690 *
state45                   1.05914    1.28478   0.824 0.409727
state46                  -0.19815    1.39685  -0.142 0.887195
state47                   0.91202    1.31205   0.695 0.486985
state48                   2.61221    1.20481   2.168 0.030147 *
state49                   1.27661    1.26566   1.009 0.313143
state50                   0.72902    1.26199   0.578 0.563482
state51                   1.35764    1.22032   1.113 0.265910
account.length            0.43861    0.43815   1.001 0.316799
international.plan2       2.51331    0.21098  11.912  < 2e-16 ***
voice.mail.plan2         -2.29221    0.89541  -2.560 0.010469 *
number.vmail.messages     1.66434    1.42054   1.172 0.241347
total.day.minutes         6.38533    0.52277  12.214  < 2e-16 ***
total.day.calls           0.45752    0.44972   1.017 0.308986
total.eve.minutes         3.80613    0.50079   7.600 2.96e-14 ***
total.eve.calls               0.41259    0.46988   0.878 0.379901
total.night.minutes            1.80238    0.46701   3.859 0.000114 ***
total.night.calls              0.14323    0.45194   0.317 0.751311
total.intl.minutes             2.03071    0.46554   4.362 1.29e-05 ***
total.intl.calls             -1.16879    0.37187  -3.143 0.001672 **
number.customer.service.calls   0.24103    0.24945   0.966 0.333934
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1542.1  on 2238  degrees of freedom
Residual deviance: 1027.5  on 2175  degrees of freedom
AIC: 1155.5
Confusion Matrix
predicted
actual   0   1
     1 493   5
     2  42  18

Accuracy : 91.57
FNR : 0.7
Sensitivity : 0.3
Specificity : 0.9899
```

With Logistic Regression, we have got the accuracy of 91.57%

2.2.4 KNN Algorithm

K nearest neighbours is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

```
Confusion Matrix:
Confusion Matrix
   prediction
actual   1   2
     1 493   5
     2  57   3

Accuracy : 88.88
FNR : 0.95
Sensitivity : 0.05
Specificity : 0.9899
```

With KNN algorithm, we have got the accuracy of

2.2.5 Naïve Bayes model

Naïve Bayes is probabilistic machine learning model. It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Confusion Matrix:

```
Confusion Matrix and Statistics

      predicted
actual   1   2
     1 495   3
     2  40  20

               Accuracy : 0.9229
                 95% CI : (0.8976, 0.9437)
    No Information Rate : 0.9588
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4491
 Mcnemar's Test P-Value : 4.021e-08

            Sensitivity : 0.9252
            Specificity : 0.8696
         Pos Pred Value : 0.9940
         Neg Pred Value : 0.3333
             Prevalence : 0.9588
         Detection Rate : 0.8871
   Detection Prevalence : 0.8925
      Balanced Accuracy : 0.8974

       'Positive' Class : 1
```

with naïve baye’s we have got the accuracy of 92.29%.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. In customer churn prediction, it is more important to identify customer who will churn rather than not churn. In customer churn, False negative rate should be low as predicting churn customer incorrectly will lead to company loss where as though False positive rate is high as predicting nonchurn customer incorrectly will make the company loose some money but customer will be happy getting rewards and will continue service. Hence we can use sensitivity and specificity as our evaluation criteria as Low FNR gives highest sensitivity i.e.positive predicted value(TP) and low FPR gives highest specificity i.e. Negatively predicted value(TN). However, sensitivity is more important than specificity as higher sensitivity avoids incorrectly predicting churned customer. Hence, we can compare the models using following criteria:

i. Accuracy – should be high
ii. FNR - should be low
iii. Sensitivity – should be high
iv. Specificity – should be high

3.1.1 Accuracy

Classification accuracy is the ratio of correct predictions to total predictions made.

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + TN + FP + FN}}$$

3.1.2 False Negative rate

False negative rate is the one in which actual test value is positive but we predicted it negative.

$$\boldsymbol{FNR = \frac{FN}{FN + TP}}$$

3.1.3 Sensitivity

Sensitivity (also called the true positive rate, the recall, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified as such (e.g., the percentage of churned customer who are correctly identified as having the condition).

$$\boldsymbol{Sensitivity = \frac{TP}{TP + FN}}$$

3.1.4 Specificity

Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of nonchurn customer who are correctly identified as not having the condition).

$$\boldsymbol{Specificity = \frac{TN}{TN + FP}}$$

For above formulae:
TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative

3.1 Model Selection

Based on the above values for different models, decision tree is the model which gives best results

Appendix A - Extra Figures

Effect of 1(0.03%) outlier on number.vmail.messages

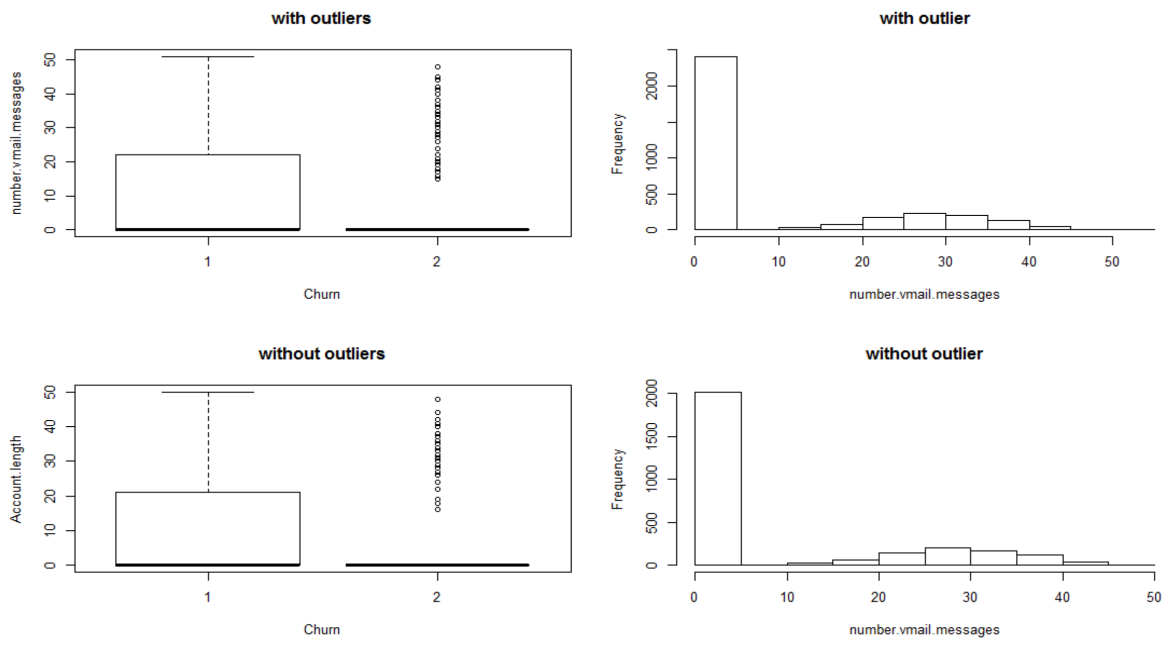


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 24 (0.72%) outlier on total.day.minutes

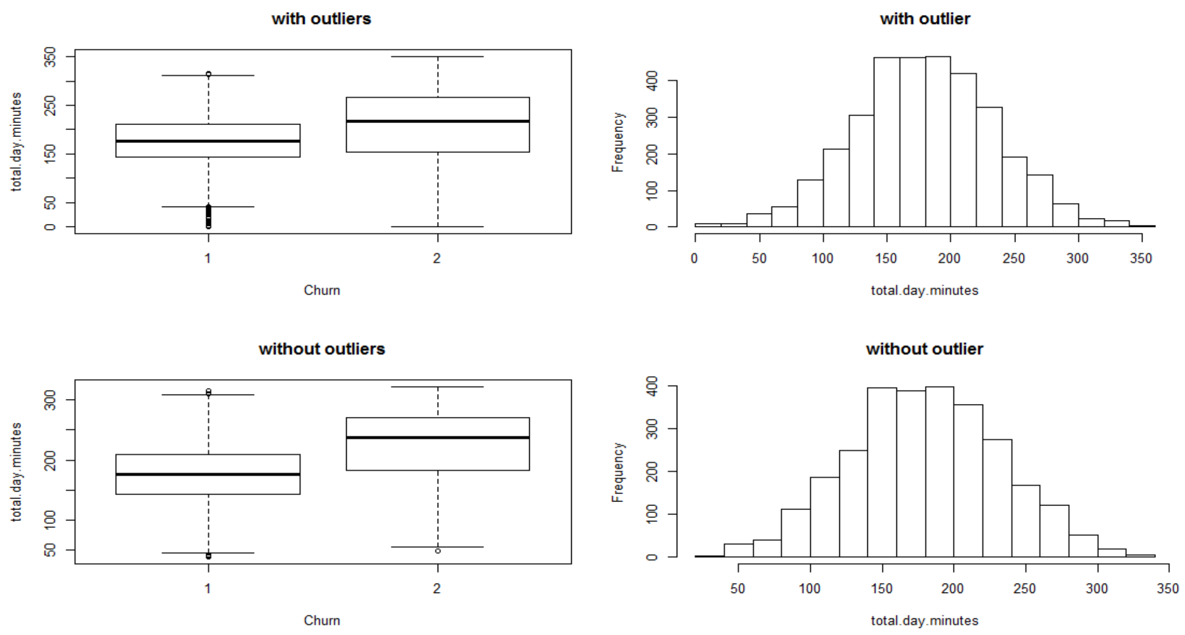


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 21 (0.63%) outlier on total.day.calls

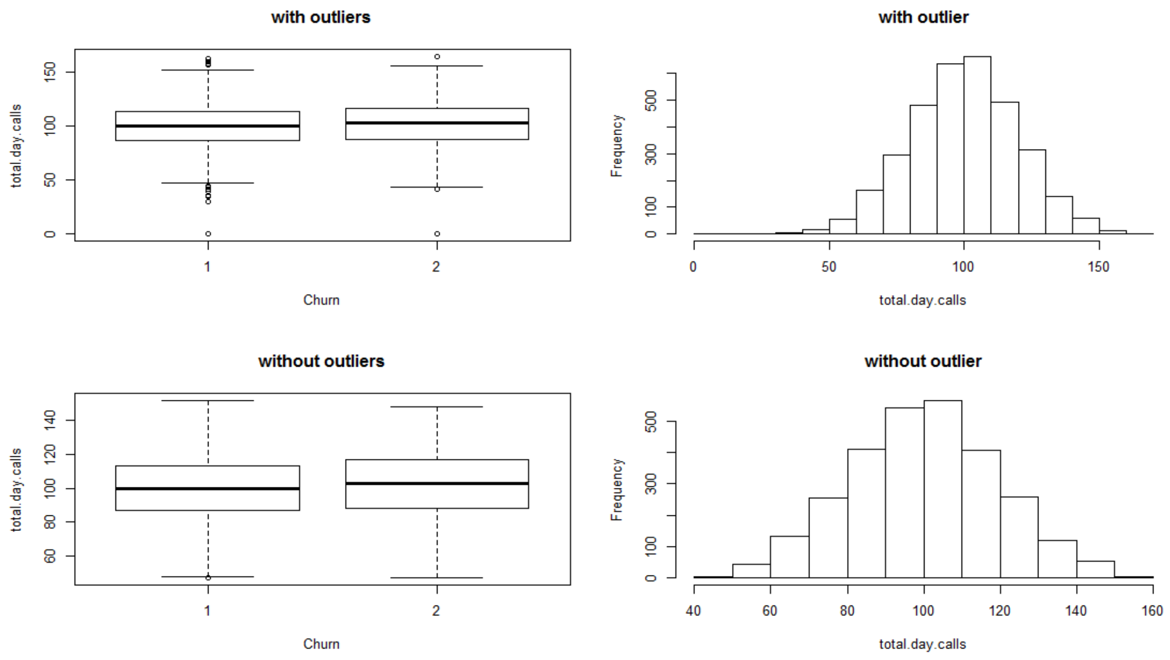


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 2(0.06%) outlier on total.day.charges

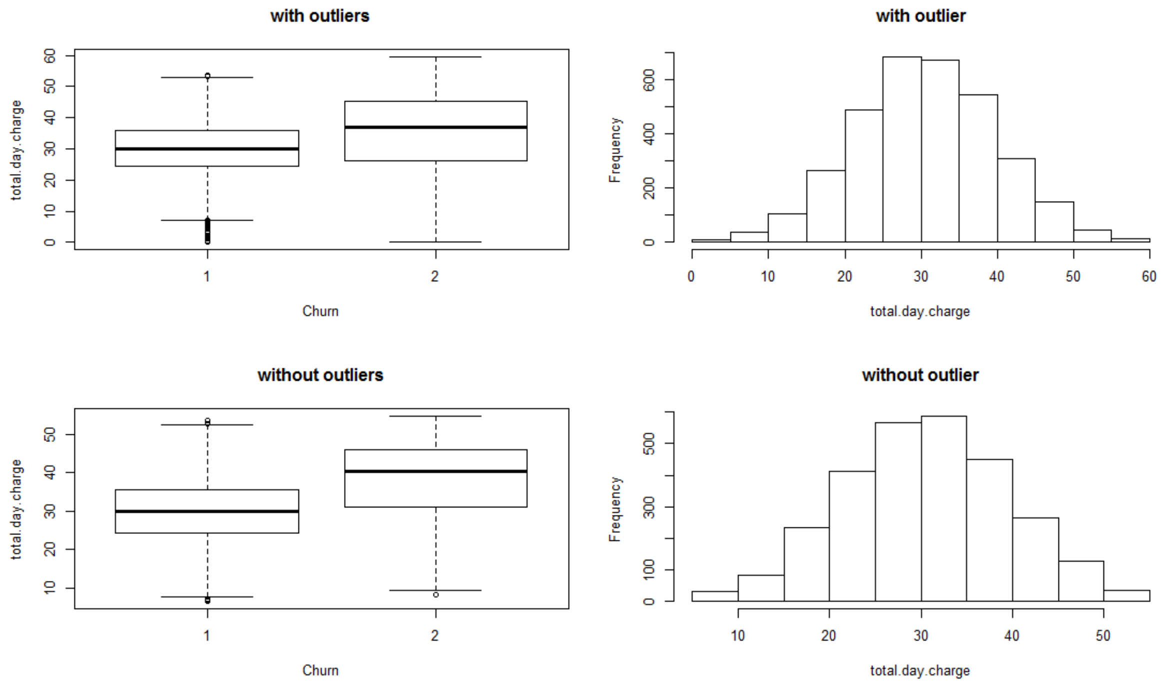


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 25 (0.75%) outlier on total.eve.minutes

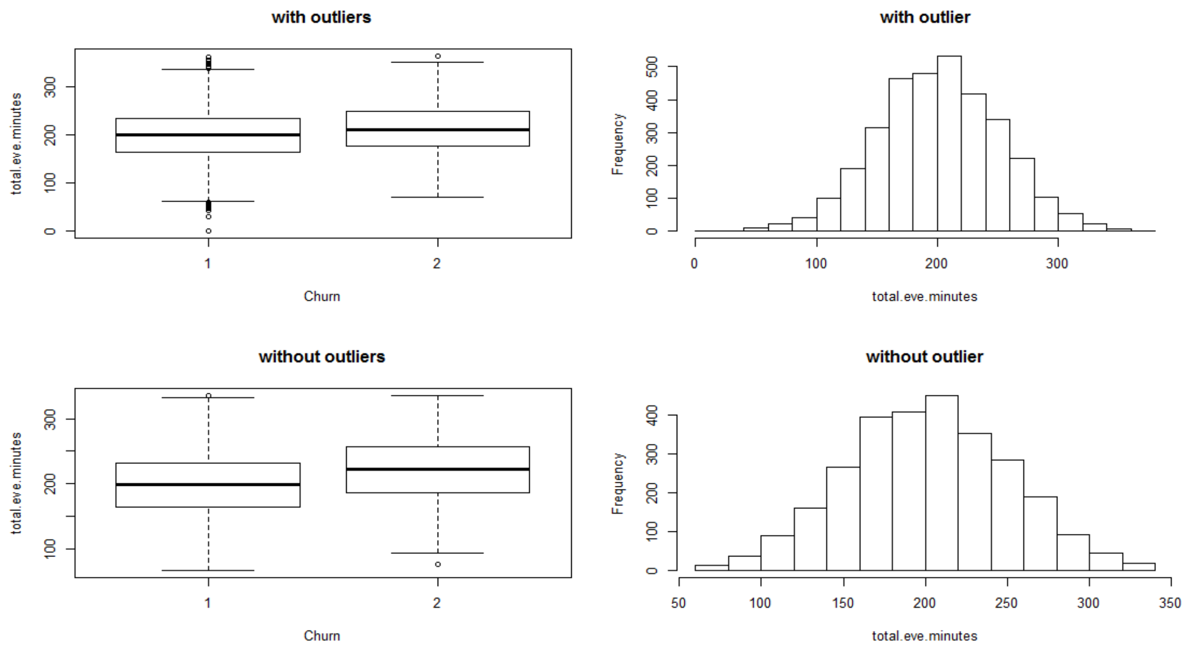


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 19 (0.57%) outlier on total.eve.calls

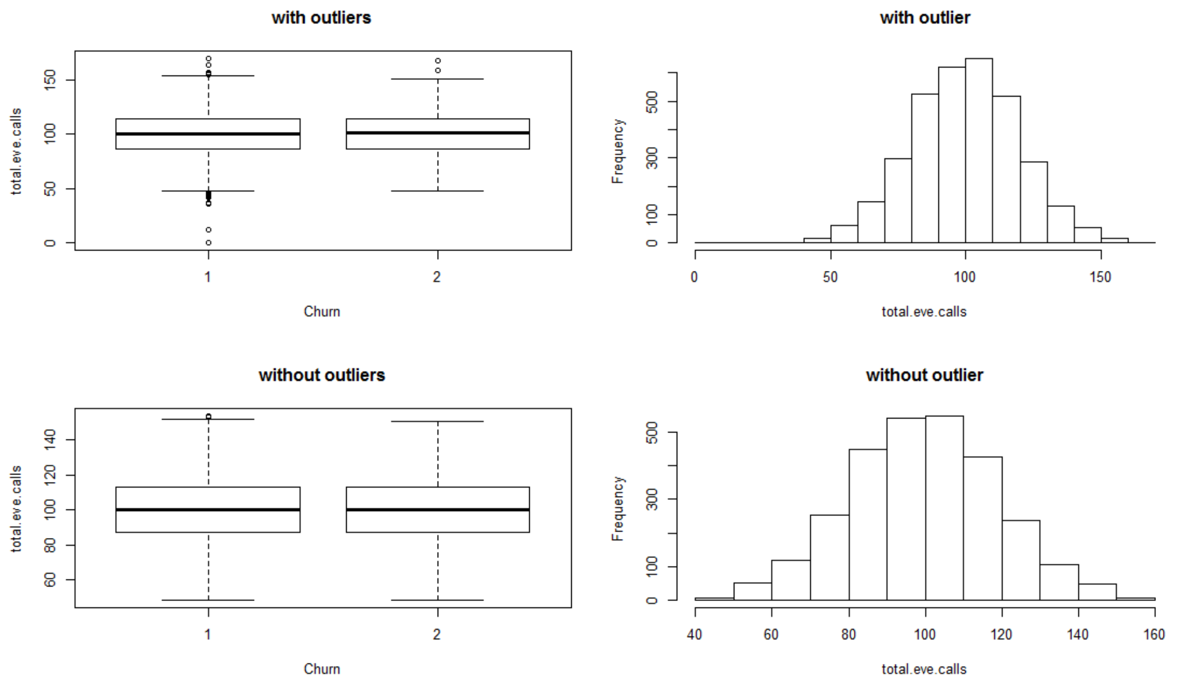


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 2 (0.06%) outlier on total.day.minutes

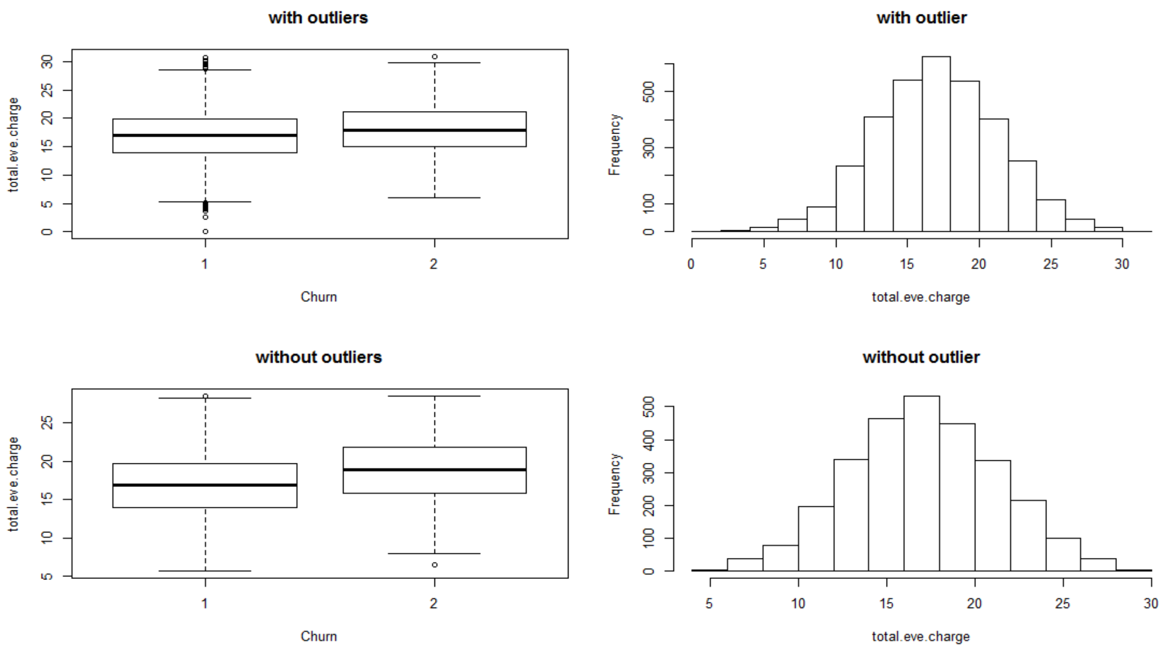


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 30 (0.9%) outlier on total.night.minutes

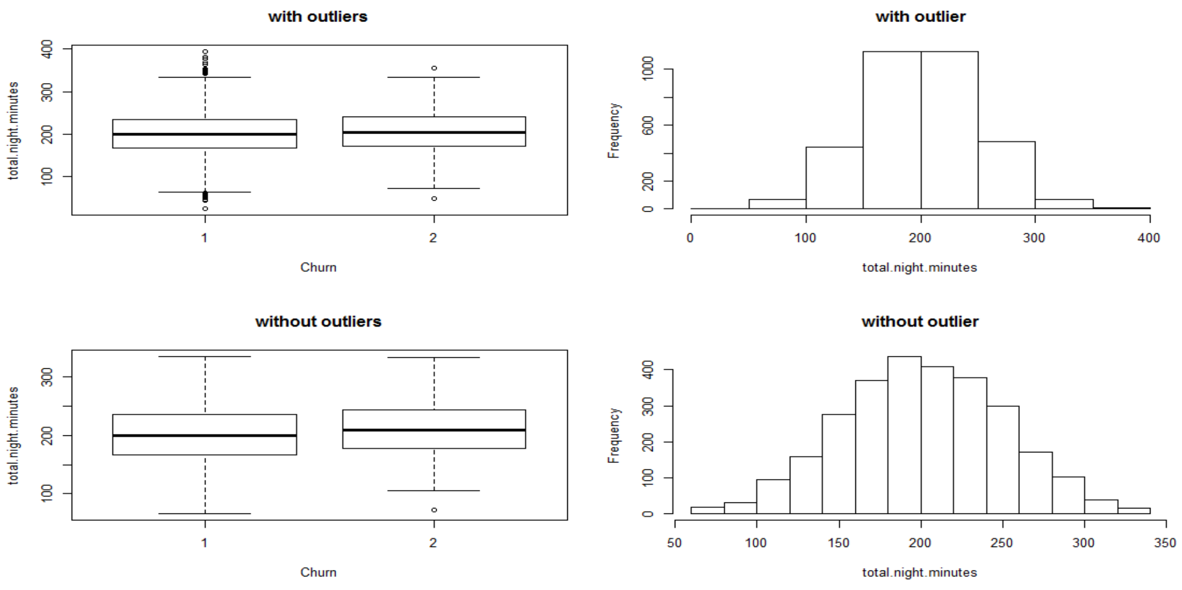


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 20 (0.6%) outlier on total.night.calls

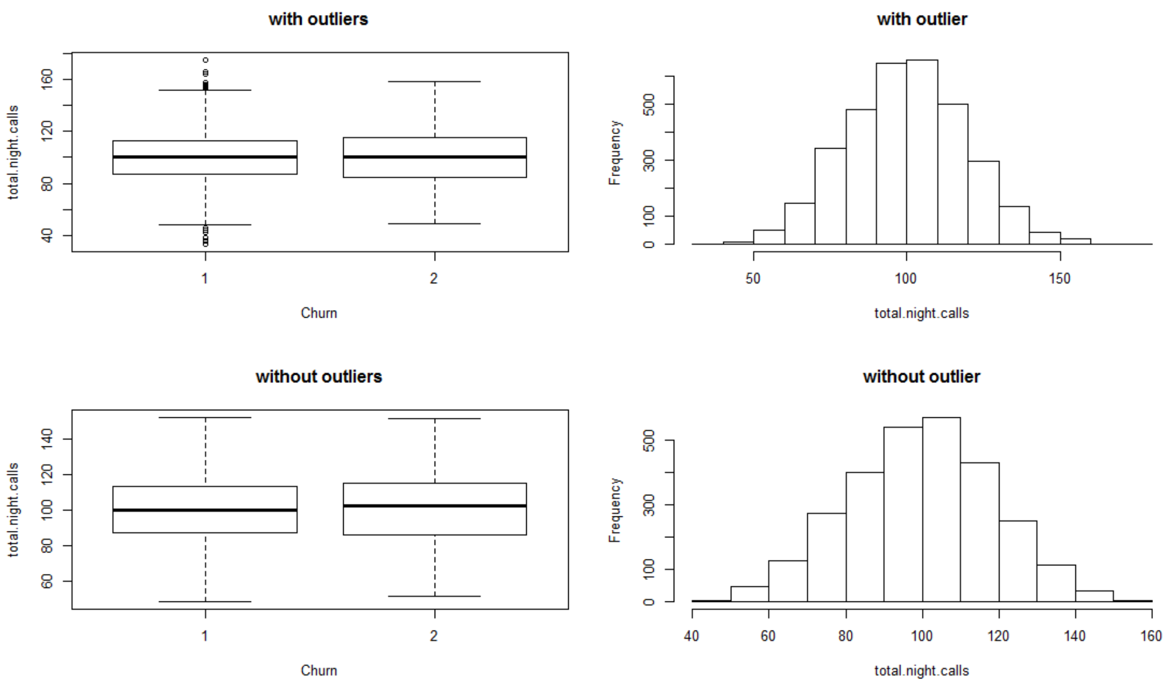


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 44 (1.32%) outlier on total.intl.minutes

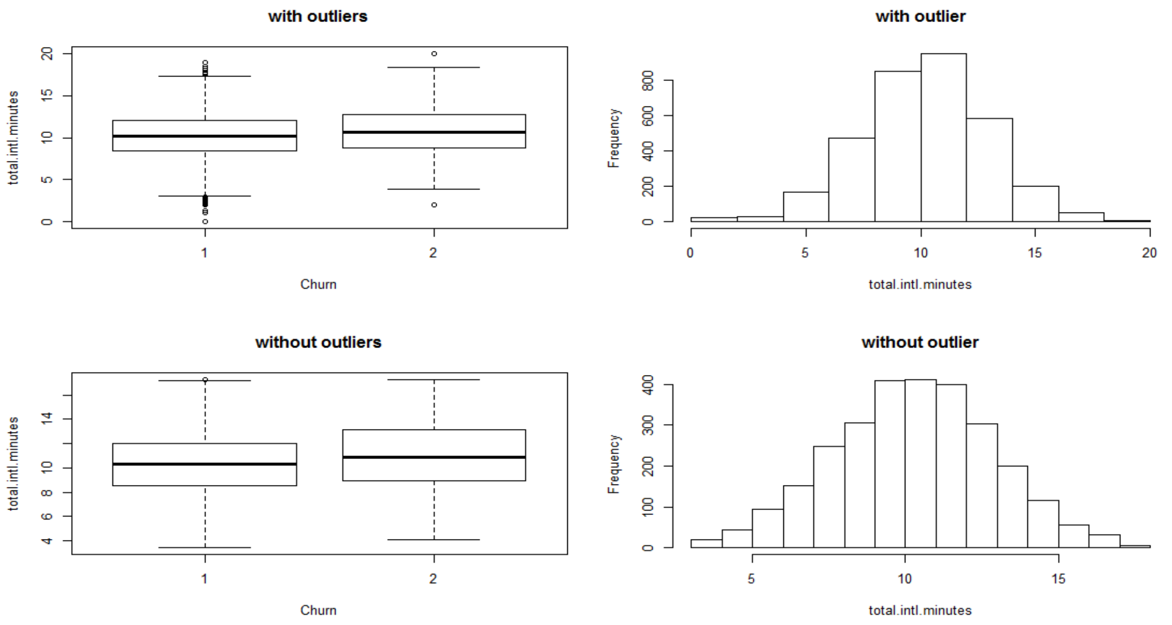


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 5 (0.15%) outlier on total.day.charges

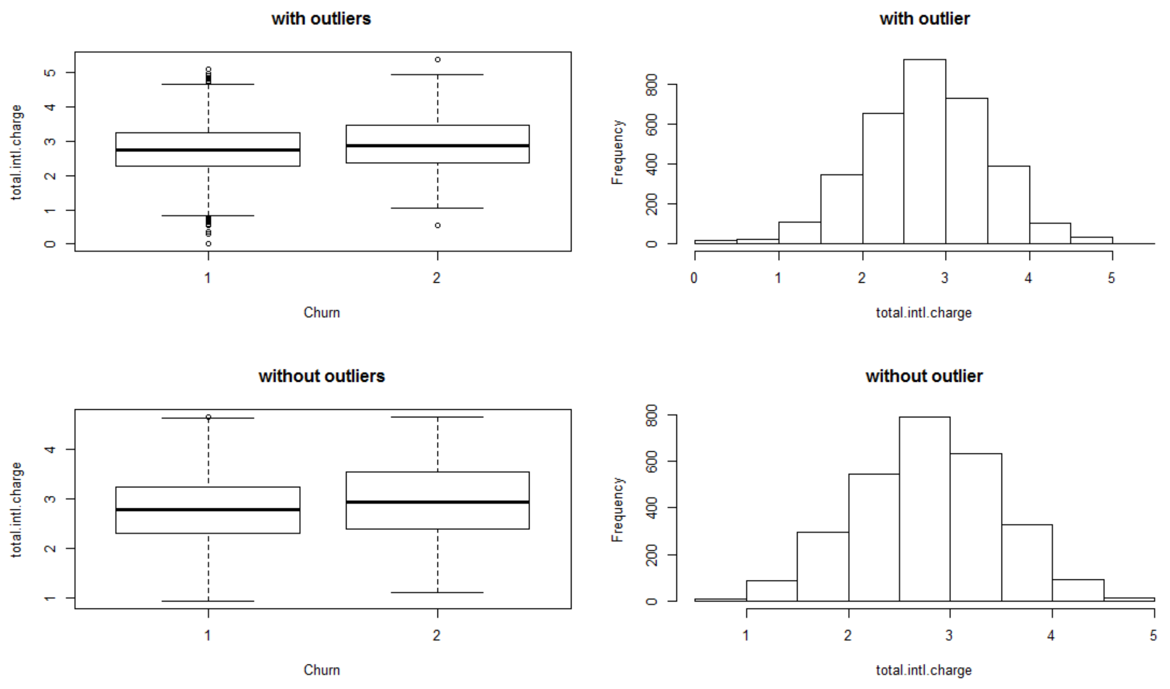


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 76 (2.28%) outlier on total.intl.calls

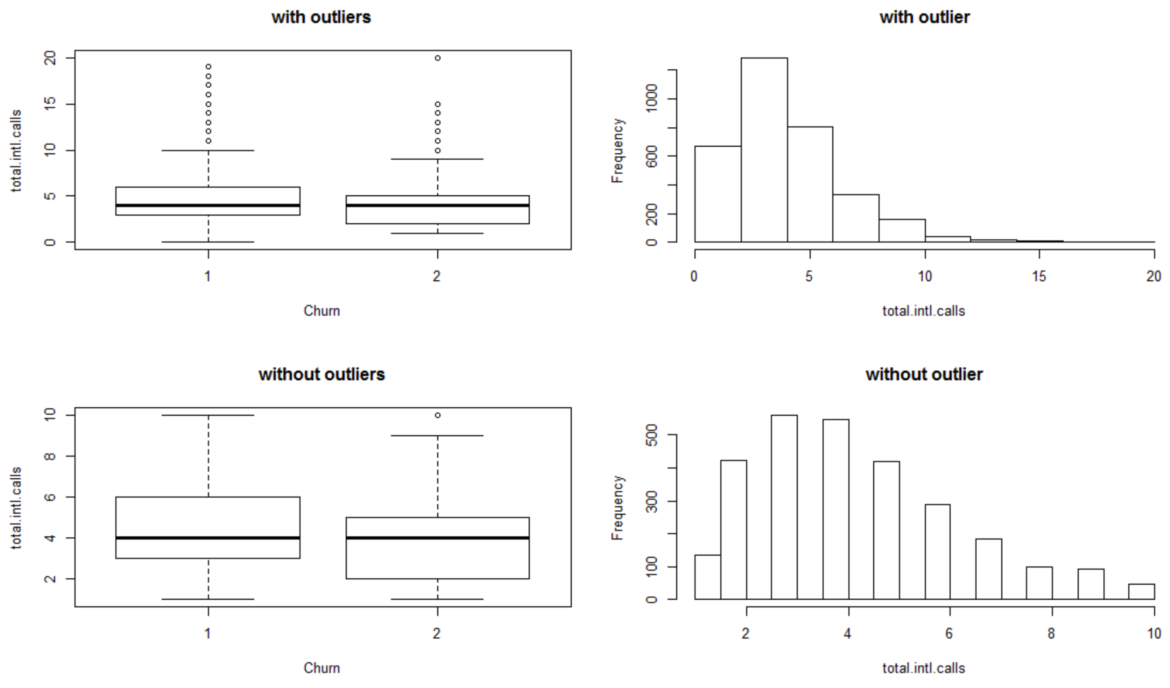


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Effect of 249 (7.07%) outlier on number.customer.service.calls

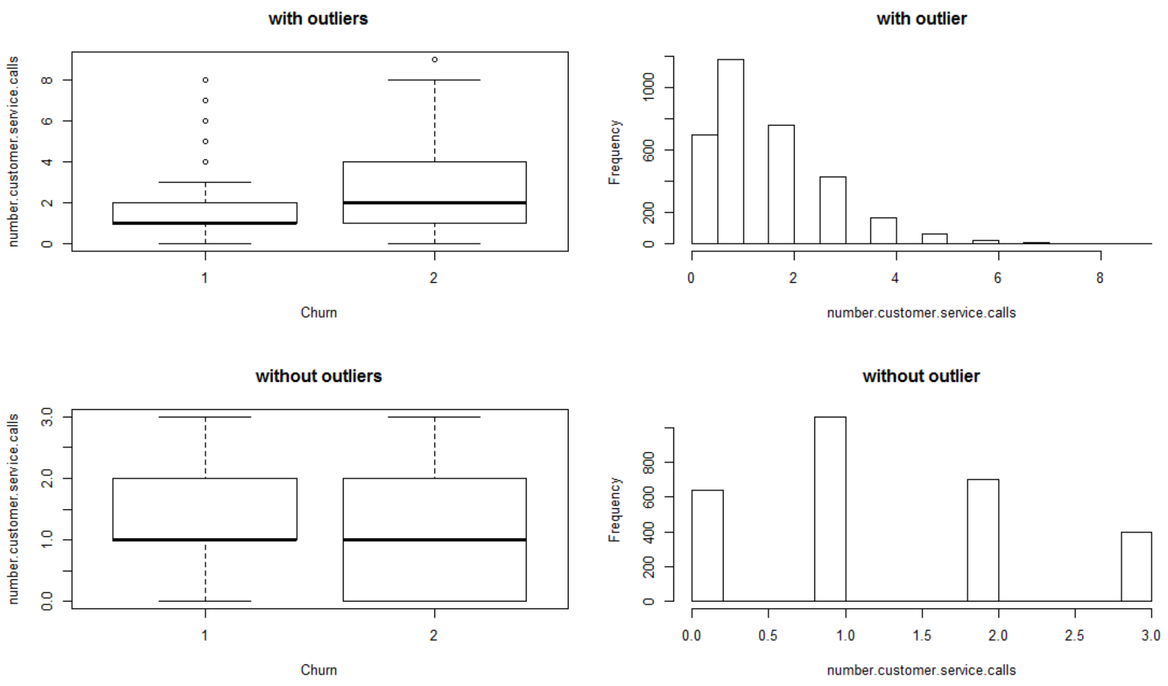


Fig.: Effect of Outliers on Predictor Variables of customers churn data

Appendix B – R Code

Customer churn ratio (Fig.2.1)

```
churn = data.frame(table(churn_train$Churn))
churn$percentage = churn$Freq
churn$percentage = (churn$percentage/nrow(churn_train)*100)
names(churn)[1] = "Churn"
names(churn)[2] = "count"
ggplot(churn, aes(x = churn$Churn, y = churn$percentage,fill = Churn)) +
  geom_bar(stat="identity") + theme_bw() +
  xlab("Churn") + ylab('Percentaget')  +
  ggtitle("Customer Churn Analysis") +  theme(text=element_text(size=12))
```

Bar plot for state variable (Fig.2.2)

```
ggplot(data = churn_train,aes(x=state,fill = Churn)) +
  geom_bar(stat = "count")
```

Bar plot for multiple categorical variable (Fig.2.3)

```
plot1 = ggplot(data = churn_train,aes(x=area.code,fill = Churn)) +
  geom_bar(stat = "count",position = "dodge")
plot2 = ggplot(data = churn_train,aes(x=international.plan,fill = Churn))
+
  geom_bar(stat = "count",position = "dodge")
plot3 = ggplot(data = churn_train,aes(x=voice.mail.plan,fill = Churn)) +
  geom_bar(stat = "count",position = "dodge")
grid.arrange(plot1,plot2,plot3,nrow = 2,ncol = 2)
```

Probability density function of churn data(Fig.2.4)

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

numeric_data = churn_train[,numeric_index]

cnames = colnames(numeric_data)
multi.hist(numeric_data,nrow = 5,ncol = 3,bcol="linen",
dcol=c("blue","red"),dlty=c("solid","solid"),main=NULL)
```

Box plot for churn data (Fig.2.5)

```
numeric_index = sapply(churn_train,is.numeric) #selecting only numeric
numeric_data = churn_train[,numeric_index]
with_outlier = cbind(numeric_data,churn_train$Churn)
names(with_outlier)[16] = "Churn"
cnames = colnames(numeric_data)


for (i in 1:length(cnames))
```

```
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"),
data = subset(churn_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],x="Churn")+
           ggtitle(paste("Box plot of churn for",cnames[i])))
}
```

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)
```

Effect of outlier on predictor variable (Fig.2.7)

```
par(mfcol = c(2,2))
plot(with_outlier$Churn,with_outlier$account.length,
     xlab = "Churn",
     ylab = "Account.length",
     main = "with outliers")
plot(without_outlier$Churn,without_outlier$account.length,
     xlab = "Churn",
     ylab = "Account.length",
     main = "without outliers")
hist(with_outlier$account.length,xlab = "Account.length",main = "with outl
ier")
hist(without_outlier$account.length,xlab = "Account.length",main = "withou
t outlier")
```

Complete R File:

```
#Remove all object to clear the environment
rm(list=ls(all=T))
#set the working directory
setwd("C:/Users/vrush_000/Desktop/Data science/Project/Main Project/Custom
er_churn")
#Check the working directory
getwd()

## [1] "C:/Users/vrush_000/Desktop/Data science/Project/Main Project/Custo
mer_churn"
```

```r
#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced"
, "C50", "dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", "sampling", "DataCombine", "inTrees"
,"psych","gridExtra","class","e1071")
#install.packages(x)
lapply(x, require, character.only = TRUE)

rm(x)

## Read the data
churn_train = read.csv("Train_data.csv", header = T, na.strings = c(" ", "
", "NA"))
churn_test = read.csv("Test_data.csv", header = T, na.strings = c(" ", "",
"NA"))
#Create a new variable in train and test data to find and reseparate them
after merging
churn_train$Istraindataset = TRUE
churn_test$Istraindataset = FALSE
#create a new combined dataset
churn_full = rbind(churn_train,churn_test)

##Explore the data
#dimension of data
dim(churn_full)

#structure of data
str(churn_full)


#checking for missing values
sum(is.na(churn_full))

#As our data does not contain any missing value, no need to do missing val
ue analysis
## Univariate Analysis and Variable Consolidation
#conversion of datatype of variable area.code to factor
churn_full$area.code = as.factor(as.character(churn_full$area.code))
#Let's have a look at summary of each variable
summary(churn_full)

#as feature "phone.number" is unique for each row and it does not have imp
act on our target variable "Churn",
#better to drop this variable
churn_full$phone.number = NULL
##Exploratory data analysis
#separate combine data
churn_train = churn_full[churn_full$Istraindataset == TRUE,]
churn_test = churn_full[churn_full$Istraindataset == !TRUE,]
#visualising churn percentages
churn = data.frame(table(churn_train$Churn))
churn$percentage = churn$Freq
churn$percentage = (churn$percentage/nrow(churn_train)*100)
names(churn)[1] = "Churn"
names(churn)[2] = "count"
```

```r
ggplot(churn, aes(x = churn$Churn, y = churn$percentage,fill = Churn)) +
  geom_bar(stat="identity") + theme_bw() +
  xlab("Churn") + ylab('Percentaget')  +
  ggtitle("Customer Churn Analysis") +  theme(text=element_text(size=12))

##visualising effect of categorical variable on target variables

#for state variable
ggplot(data = churn_train,aes(x=state,fill = Churn)) +
  geom_bar(stat = "count")

#for multiple plot on same page
plot1 = ggplot(data = churn_train,aes(x=area.code,fill = Churn)) +
  geom_bar(stat = "count",position = "dodge")
plot2 = ggplot(data = churn_train,aes(x=international.plan,fill = Churn))
+
  geom_bar(stat = "count",position = "dodge")
plot3 = ggplot(data = churn_train,aes(x=voice.mail.plan,fill = Churn)) +
  geom_bar(stat = "count",position = "dodge")
grid.arrange(plot1,plot2,plot3,nrow = 2,ncol = 2)

#Probability density function of churn data
numeric_index = sapply(churn_train,is.numeric) #selecting only numeric

numeric_data = churn_train[,numeric_index]

cnames = colnames(numeric_data)
multi.hist(numeric_data,nrow = 5,ncol = 3,bcol="linen", dcol=c("blue","red
"),dlty=c("solid","solid"),main=NULL)

##Data Manupulation; convert string categories into factor numeric
for(i in 1:ncol(churn_full)){

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

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

  }
}
#separate combine data to remove outliers
churn_train = churn_full[churn_full$Istraindataset==TRUE,]
churn_test = churn_full[churn_full$Istraindataset==FALSE,]

##Outlier Analysis
#BoxPlots - Distribution and Outlier Check
numeric_index = sapply(churn_train,is.numeric) #selecting only numeric
numeric_data = churn_train[,numeric_index]
with_outlier = cbind(numeric_data,churn_train$Churn)
names(with_outlier)[16] = "Churn"
cnames = colnames(numeric_data)

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


# ## 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)

# # #Remove outliers using boxplot method
# # #loop to remove from all variables
for (i in cnames){
  print(i)
  val = churn_train[,i][churn_train[,i] %in% boxplot.stats(churn_train[,i]
)$out]
  print(length(val))
  print((length(val)*100/nrow(numeric_data)))
  churn_train = churn_train[which(!churn_train[,i] %in% val),]
}

#probability density function of churn data without outlier
numeric_index = sapply(churn_train,is.numeric) #selecting only numeric

numeric_data = churn_train[,numeric_index]

cnames = colnames(numeric_data)
without_outlier = cbind(numeric_data,churn_train$Churn)
names(without_outlier)[16] = "Churn"
multi.hist(numeric_data,nrow = 5,ncol = 3,bcol="linen", dcol=c("blue","red
"),dlty=c("solid","solid"),main=NULL)

par(mfcol = c(2,2))
plot(with_outlier$Churn,with_outlier$account.length,
     xlab = "Churn",
     ylab = "Account.length",
     main = "with outliers")
plot(without_outlier$Churn,without_outlier$account.length,
     xlab = "Churn",
     ylab = "Account.length",
     main = "without outliers")
hist(with_outlier$account.length,xlab = "Account.length",main = "with outl
ier")
```

```r
hist(without_outlier$account.length,xlab = "Account.length",main = "withou
t outlier")

#with the above code for subplots, we can draw plots for each continous va
riable by putting it's names

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

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

for (i in 1:4)
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}

## Dimension Reduction
churn_train = subset(churn_train, select = -c(total.day.charge,total.eve.c
harge,total.night.charge,total.intl.charge,area.code))
churn_test = subset(churn_test, select = -c(total.day.charge,total.eve.cha
rge,total.night.charge,total.intl.charge,area.code))
churn_full = rbind(churn_train,churn_test)

##Feature scaling
#Normality check
multi.hist(numeric_data,nrow = 5,ncol = 3,bcol="linen", dcol=c("blue","red
"),dlty=c("solid","solid"),main=NULL)

#store cotinous variables name
cnames = c("account.length","number.vmail.messages","total.day.calls","tot
al.day.minutes","total.eve.calls","total.eve.minutes","total.night.calls",
"total.night.minutes","total.intl.calls","total.intl.minutes","number.cust
omer.service.calls"
)
#Normalization formula
#For train data
for(i in cnames){
  print(i)
  churn_train[,i] = (churn_train[,i] - min(churn_train[,i]))/
    (max(churn_train[,i] - min(churn_train[,i])))
}

#for test data
for(i in cnames){
  print(i)
  churn_test[,i] = (churn_test[,i] - min(churn_test[,i]))/
    (max(churn_test[,i] - min(churn_test[,i])))
}
```

```r
#If the distribution is normal , we can use below formula for feature scal
ing
# #Standardisation
#For train data
#for(i in cnames){
  #print(i)
  #train[,i] = (train[,i] - mean(train[,i]))/
   # sd(train[,i])
#}
#For test data
#for(i in cnames){
 # print(i)
 # test[,i] = (test[,i] - mean(test[,i]))/
 #   sd(test[,i])
#}

#Combine the data
churn_full = rbind(churn_train,churn_test)
####################################Model Development######################
#################
#Clean the environment
rmExcept("churn_full")

#Reset index of data
rownames(churn_full) = NULL
#Separating the data
churn_train =churn_full[churn_full$Istraindataset == TRUE,]
churn_test =churn_full[churn_full$Istraindataset == !TRUE,]
#deleting boolean variable
churn_train$Istraindataset = NULL
churn_test$Istraindataset = NULL
#Divide  churn_train data into train and test
set.seed(1234)
train.index = createDataPartition(churn_train$Churn, p = .80, list = FALSE
)
train = churn_train[train.index,]
test  = churn_train[-train.index,]

##Model building
##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., train, trials = 50, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

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

##Evaluate the performance of classification model
```

```r
ConfMatrix_C50 = table(actual = test$Churn, predicted = C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#Accuracy: 96.59
#FNR: 28.33
#sensitivity = 96.69
#specificity = 95.56
###Random Forest
RF_model = randomForest(Churn ~., train, importance = TRUE ,ntree=250)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)

#
# #Extract rules
exec = extractRules(treeList, train[,-15])  # R-executable conditions

#
# #Visualize some rules
exec[1:2,]

#
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
readableRules[1:2,]

#
# #Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-15], train$Churn)  # get rule met
rics

#
# #evaulate few rules
ruleMetric[1:2,]

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

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

##Result
#Accuracy = 95.7
#FNR = 36.67
#sensitivity = 95.75
#Specificity = 95.00

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

#summary of the model
summary(logit_model)
```

```r
#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
conf_matrix = table(actual=test$Churn, predicted=logit_Predictions)
conf_matrix

TP = conf_matrix[2,2]
TN = conf_matrix[1,1]
FP = conf_matrix[1,2]
FN = conf_matrix[2,1]

Accuracy = (TP+TN)/(TP+TN+FP+FN)
Accuracy

FNR = FN/(FN+TP)
FNR

sensitivity = TP/(TP+FN)
sensitivity

specificity = TN/(TN+FP)
specificity

##Result
#Accuracy:91.57
#FNR:70
#sensitivity = 30
#specificity = 98.99

##KNN Implementation
#Predict test data
KNN_Predictions = knn(train[, 1:14], test[, 1:14], train$Churn, k = 5)

#Confusion matrix
Conf_matrix = table(actual = test$Churn,prediction = KNN_Predictions)
Conf_matrix

TP = Conf_matrix[2,2]
TN = Conf_matrix[1,1]
FP = Conf_matrix[1,2]
FN = Conf_matrix[2,1]
Accuracy = (TP+TN)/(TP+TN+FP+FN)
Accuracy

FNR = FN/(FN+TP)
FNR

sensitivity = TP/(TP+FN)
sensitivity
```

```r
specificity = TN/(TN+FP)
specificity

##Result
#Accuracy = 88.88
#FNR = 95
#sensitivity = 5
#specificity = 98.99

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

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

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

#Accuracy: 92.29
#FNR: 66.67
#sensitivity = 92.52
#specificity = 86.96
#As decision tree gives best result, we will apply it on train data to pre
dict for test data
###prediction on test data
set.seed(121)
train = churn_train
test = churn_test
##Model building
##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., train, trials = 50, rules = TRUE)
#Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-15], type = "class")
#Save predicted outcomes in new file
output = churn_test$state
output = as.data.frame(output)
names(output)[1] = "State"
output$Churn = C50_Predictions
write.csv(output,"R_Prediction_Test_data.csv",row.names= FALSE)
```

References

Jared P. Lander,2014. R for Everyone: Advanced Analytics and Graphics

Winston chang,2012.R Graphics Cookbook

Antonio J. Tallon-Ballesteros and Jose C. Riquelme, "Deleting or Keeping Outlier for Classifier training?", Department of Languages and Computer Systems University of Seville,Seville, Spain