Churn reduction

Gourav Kumar

27 June 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 the customers responses according to the given data.

1.2 Data

Data set given for the analysing has 21 variables and 3333 observations. Our task is to build classification on top of this. Given below is the sample of the data set.

Table 1.1 Sample data (Column 1-6)

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

Table 1.2 Sample data (Column 7-12)

number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls
25	265.1	110	45.07	197.4	99
26	161.6	123	27.47	195.5	103
0	243.4	114	41.38	121.2	110
0	299.4	71	50.9	61.9	88

Table 1.3 Sample data (Column 13-18)

total eve charge	Total night minutes	total night calls	total night charge	total intl minutes	total intl calls
16.78	244.7	91	11.01	10	3
16.62	254.4	103	11.45	13.7	3
10.3	162.6	104	7.32	12.2	5
5.26	196.9	89	8.86	6.6	7
12.61	186.9	121	8.41	10.1	3

Table 1.4 Sample data (Column 19-21)

total intl charge	number customer service calls	Churn
2.7	1	False.
3.7	1	False.
3.29	0	False.
1.78	2	False.
2.73	3	False.

Table 1.5: Predictor Variables

Serial No.	Predictor
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
21	Churn

Chapter 2

Methodology

2.1 Pre Processing

In every data analysis Pre Processing of data is very important. Raw data is not a good fit for the model. For doing modeling or getting insight of a data we need to mold the data. Data may contain some missing values in the observation or may be some higher value which could make the model bias.

There are some Pre Processing techniques:

- Missing Value Analysis
- Outlier Analysis
- Feature Selection
- Feature Scaling
- Sampling

For our give data set we perform missing value analysis. If there will be any missing value we need to compute its value or eliminate it according to our requriment.

```
#************************** Missing Value for Training data ****************************
missing_val = data.frame(apply(Train_data,2,function(x){sum(is.na(x))}))
missing_val
missing_val$Columns = row.names(missing_val)
```

```
state                     0
account.length            0
area.code                 0
phone.number              0
international.plan        0
voice.mail.plan           0
number.vmail.messages     0
total.day.minutes         0
total.day.calls           0
total.day.charge          0
total.eve.minutes         0
total.eve.calls           0
total.eve.charge          0
total.night.minutes       0
total.night.calls         0
```

```
total.night.charge                  0
total.intl.minutes                  0
total.intl.calls                    0
total.intl.charge                   0
number.customer.service.calls       0
Churn                               0
```

There is no missing value in any column.

After this we need to check the datatype of the variables

```
# state                          : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32
# 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                   : chr " 3824657" " 3717191" " 3581921" "
# 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
```

For further analysis we need to convert categorical data into factor data.

2.1.1 Outlier Analysis

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

For this I have used BOX Plot method. Box plot can be used on continuous data. Box plot has Inter quartile range, Upper fence and Lower fence values present above or below the fences are consider as an outliers.

Here we have two values for the Churn (1 -> False ,2 -> True)

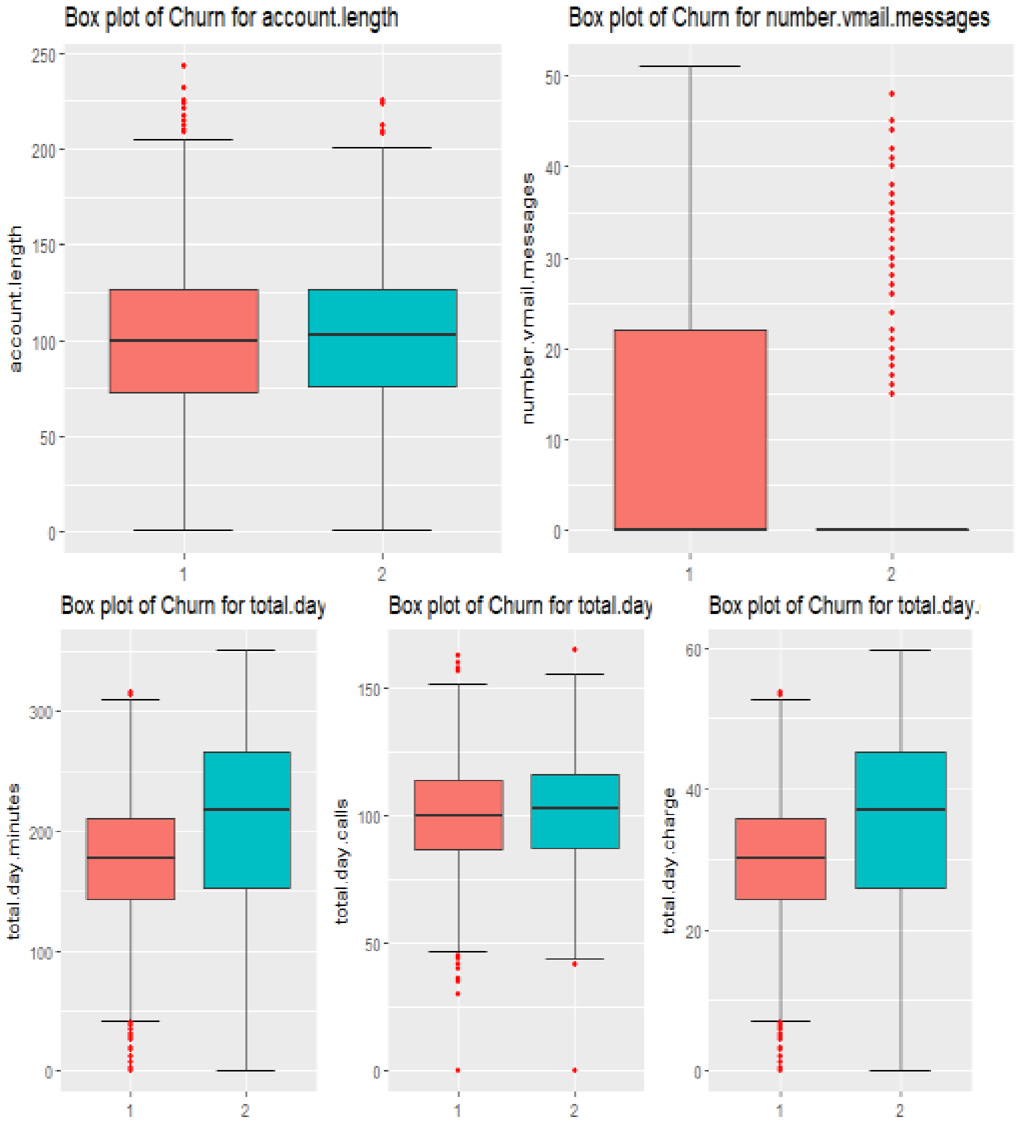


Fig 2.1.1(a) Box plot of Independent variables

Red dots are indicating the Outliers. To fit this data into a model we need to remove it.

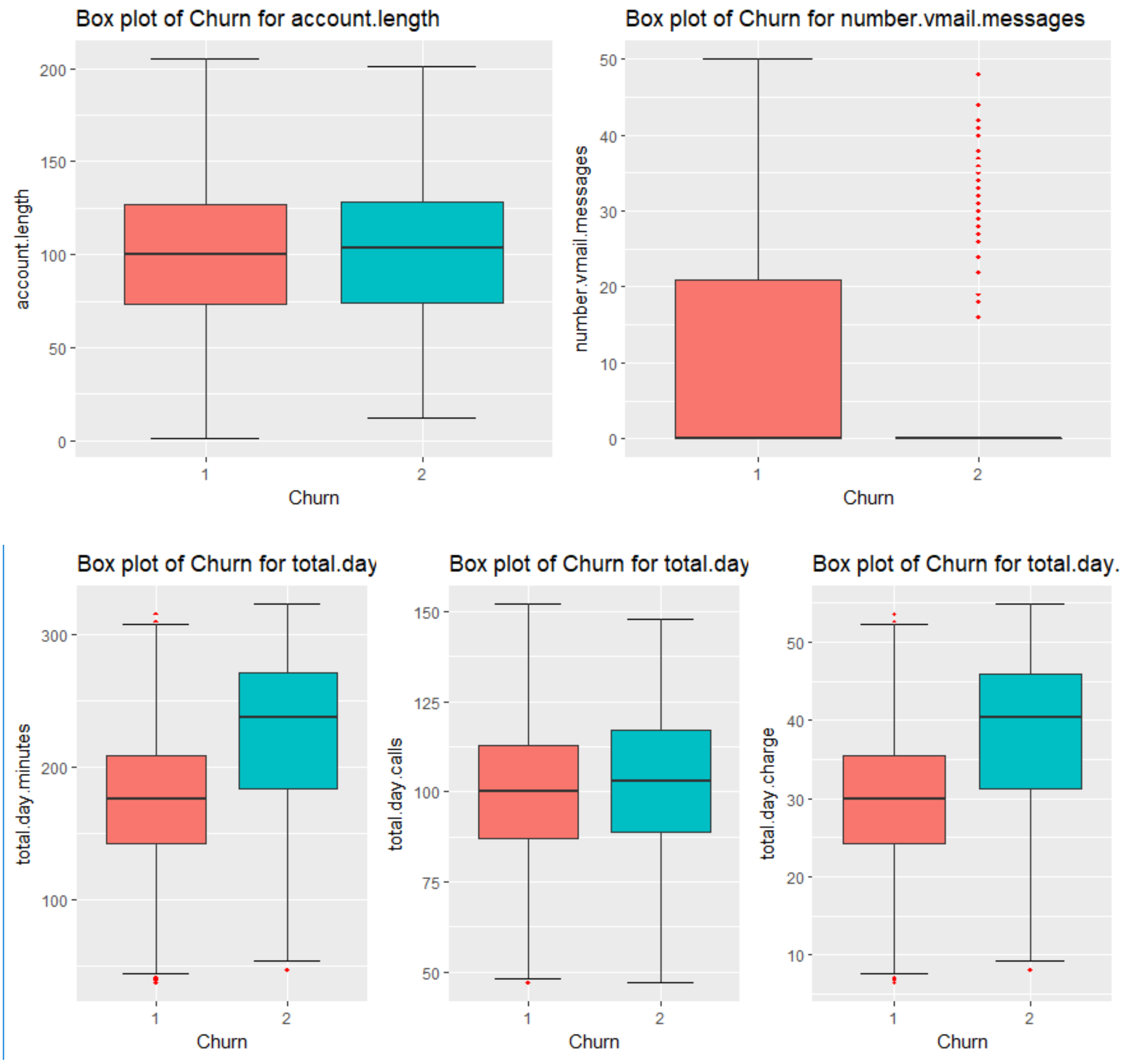


Fig 2.1.1(b) Box plot of Independent variables after removing outliers

2.1.2 Feature Selection

Before performing any type of modeling we need to check the importance of each independent 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. Below we have used Correlation plot for numerical data and chi-.square test for categorical data.

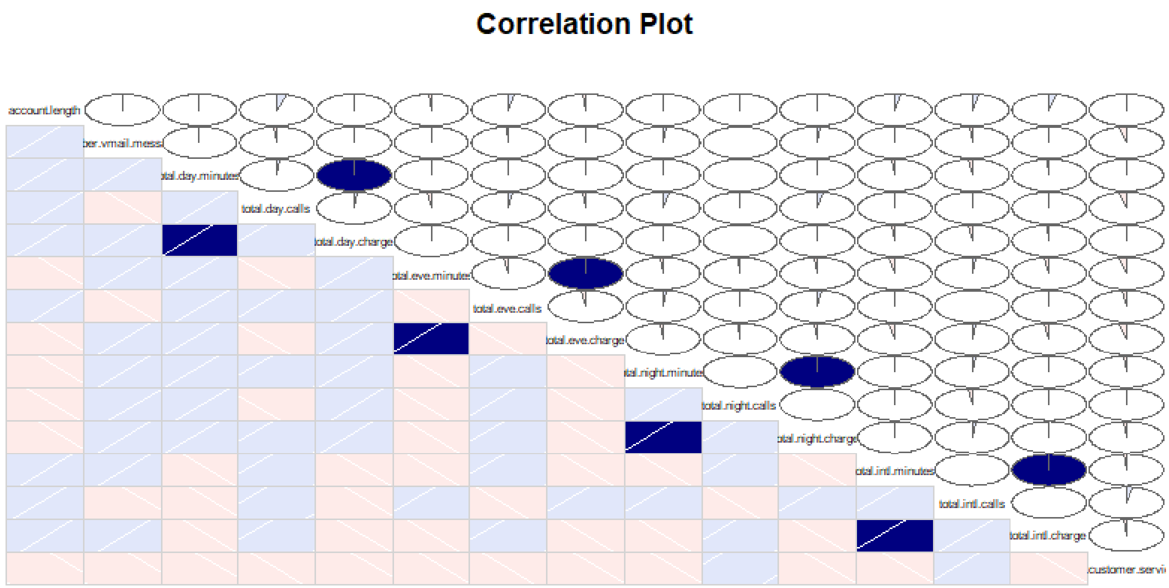


Fig 2.1.2(a) Correlation plot

Blue colour indicates the extremely Positive correlation and Dark Red Colour shows Extremely Negative correlation.

Chi-square test result:

```
[1] "state"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 72.42, df = 50, p-value = 0.02074

[1] "area.code"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 0.72143, df = 2, p-value = 0.6972

[1] "international.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 237.1, 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 = 26.494, df = 1, p-value = 2.644e-07
```

Dependence between Independent and dependent variable should be high

Ideally there should be no dependency between independents variables.

From the fig 2.1.2(a) we come to know that total.day.minutes, total.eve.minutes, total.night.minu, total.intl.minutes and phone.number are showing high correlation with independent varables so we have to remove these to prevent out model with multicollinearity.

Feature Scaling: We scale down the variance of the data in a range of 0 to1, because any higher data point may bias our prediction result. For feature scaling I have used Normalization method.

Normalization = (actual Value - Min. value) / (Max.value - Min.value)

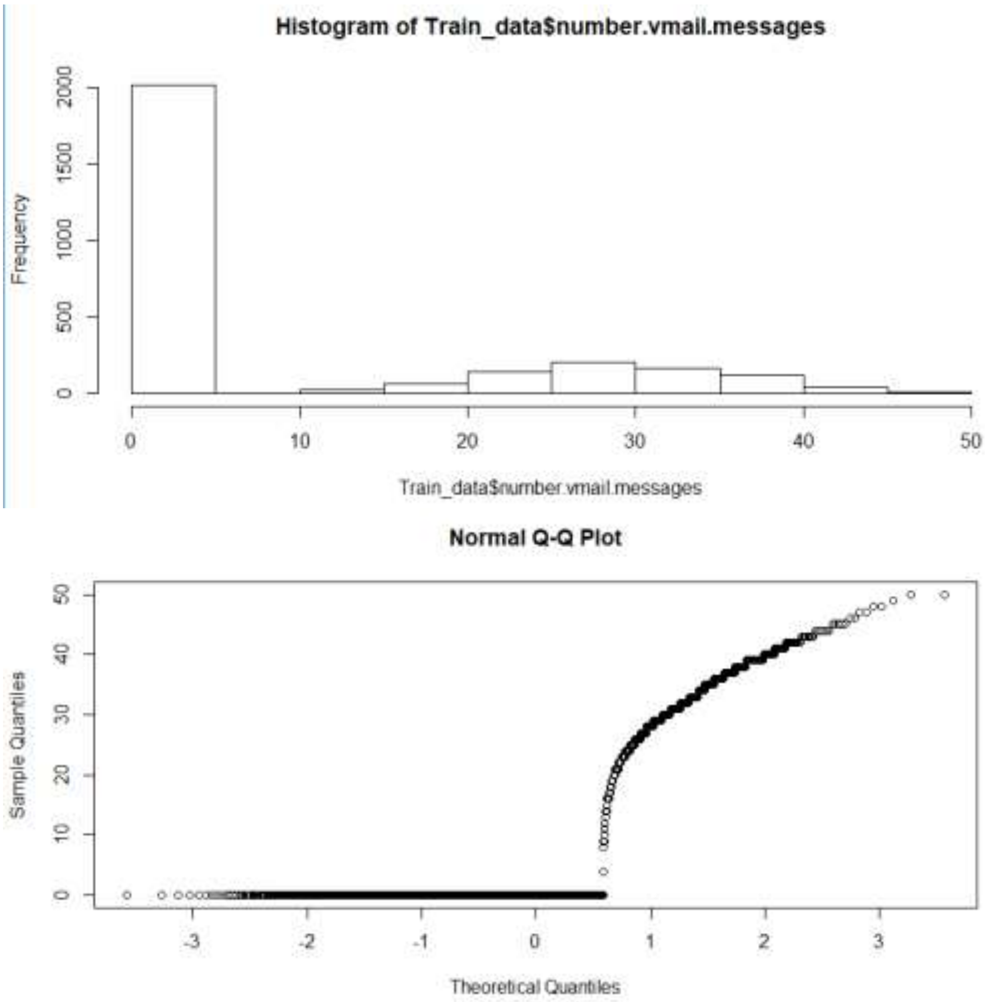


Fig 2.1.2 (b) Variable range before Normalization

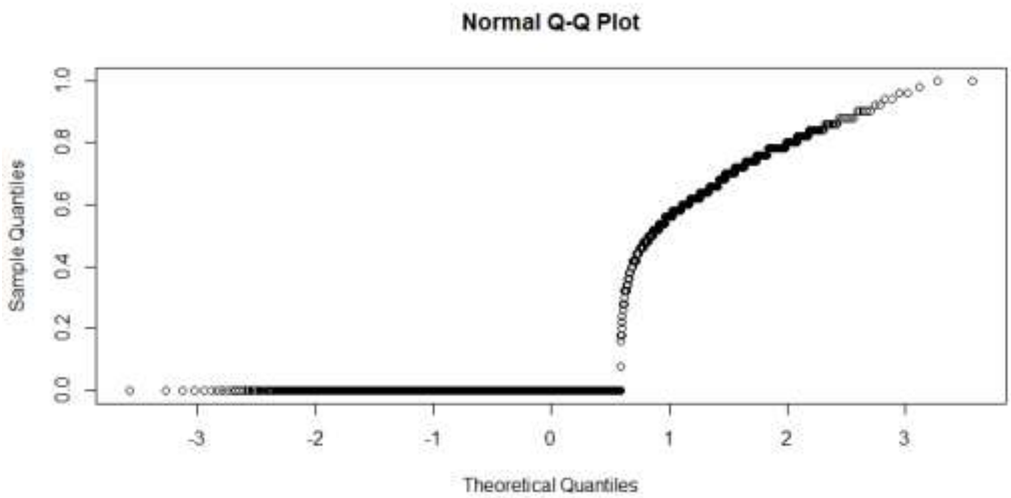


Fig 2.1.2 (c) Variable range after Normalization

Feature scaling is done only for continuous numerical variable, Categorical variables doesn’t require feature scaling.

2.2 Modeling

2.2.1 Model Selection:

From our data set and pre-processing method we come to know that this is a classification problem. We cannot use regression algorithms on it.

There is a two class in our Target variable (Churn) False and True.

2.2.2 Decision Tree Classification:

Decision Tree Algorithm Pseudocode:

- Place the best attribute of the dataset at the root of the tree.
- Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
- Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

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

C5.0 [Release 2.07 GPL Edition]    Wed Jun 27 01:13:20 2018

-------------------------------

Class specified by attribute `outcome'

Read 2797 cases (16 attributes) from undefined.data

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

Rules:

Rule 0/1: (777/46, lift 1.1)

        voice.mail.plan = 2

        ->  class 1  [0.940]

Rule 0/2: (2020/258, lift 1.0)

        voice.mail.plan = 1

        ->  class 1  [0.872]

Rule 0/3: (65, lift 9.1)

        voice.mail.plan = 1
```

total.day.charge > 44.88

total.eve.charge > 17.02

total.night.charge > 5.78

-> class 2 [0.985]

Rule 0/4: (51, lift 9.0)

international.plan = 2

total.intl.calls <= 2

-> class 2 [0.981]

Rule 0/5: (44, lift 9.0)

international.plan = 2

total.intl.charge > 3.51

-> class 2 [0.978]

Rule 0/6: (56/1, lift 8.9)

voice.mail.plan = 1

total.day.charge > 46.82

total.eve.charge > 14.23

-> class 2 [0.966]

Rule 0/7: (49/1, lift 8.8)

voice.mail.plan = 1

total.day.charge > 40.14

total.eve.charge > 17.02

total.night.charge > 9.9

-> class 2 [0.961]

Rule 0/8: (22, lift 8.8)

voice.mail.plan = 1

total.day.charge > 46.82

number.customer.service.calls <= 0

-> class 2 [0.958]

Rule 0/9: (40/1, lift 8.8)

voice.mail.plan = 1

total.day.charge > 37.71

total.eve.charge > 22.72

-> class 2 [0.952]

Rule 0/10: (24/1, lift 8.5)

account.length > 0.5490196

voice.mail.plan = 1

total.day.charge > 46.82

-> class 2 [0.923]

Rule 0/11: (56/4, lift 8.4)

voice.mail.plan = 1

total.day.charge > 40.14

total.eve.charge > 20.6

total.intl.charge > 2.05

-> class 2 [0.914]

From our decision tree model summery we see some rules. In Rule 0/11 we can see that it is predicting True with confidence of 91.4% and its lift value is 8.4 which is quite good. According to the industry standard Rules should have:

Lift value >1

Support value > 20%

Confidence value > 80%

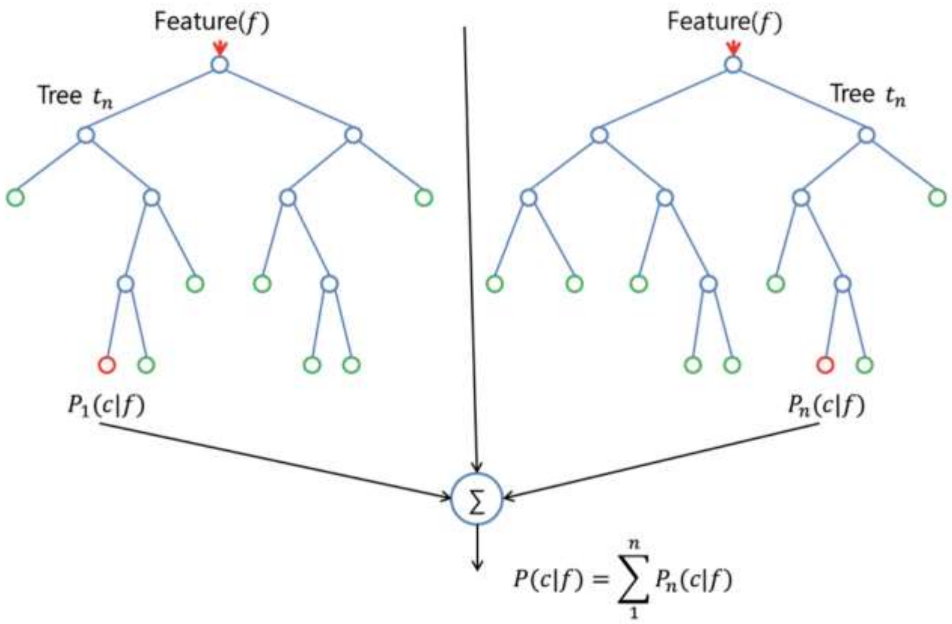
Attribute usage:

100.00%	state
100.00%	account.length
100.00%	international.plan
100.00%	voice.mail.plan
100.00%	total.day.calls
100.00%	total.day.charge
100.00%	total.eve.charge
100.00%	total.night.charge
100.00%	total.intl.calls
100.00%	total.intl.charge
99.86%	total.night.calls
99.57%	total.eve.calls
98.32%	number.vmail.messages
97.46%	area.code
94.82%	number.customer.service.calls

Attribute usage showing how much amount of amount of percentage each variable helping us to explain the target variables.

2.2.3 Random Forest Classification

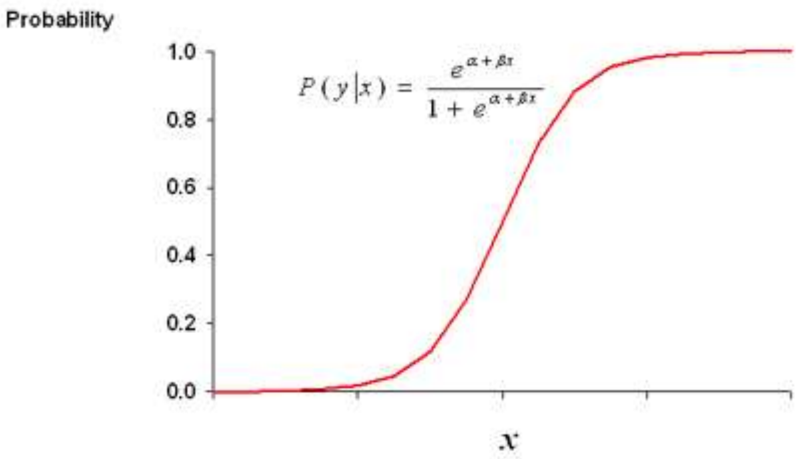
Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.



For making this model I have used randomForest library and iterating the model to the 1000 time to get the optimum result.

2.2.4 Logistic Regression

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits. These values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.



For logistic regression we assume p = 0.5, classification probability above this is 1 and vice-versa.

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0710  -0.3871  -0.2162  -0.0977   3.5472
```

2.2.5 SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. I have use Grid search for SVM for optimum value for the hyper parameter and I get optimum result for `SVM_model$bestTune`

```
        sigma C
3 0.007821692 1
```

2.2.6 Naive Bayes

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)}$$

where
P(H) is the probability of hypothesis H being true. This is known as the prior probability.
P(E) is the probability of the evidence(regardless of the hypothesis).
P(E|H) is the probability of the evidence given that hypothesis is true.
P(H|E) is the probability of the hypothesis given that the evidence is there.

This method is preferred for the categorical variables for continuous numerical variables data should be normally distributed.

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. There are several criteria that exist for evaluating and comparing models. We can compare the models usingany of the following criteria:

1. Accuracy
2. False negative Rate
3. Computational Efficiency

In our case *Computation Efficiency*, do not hold much significance. Therefore we will use *Accuracy* and *False negative Rate* as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with ROC curve.

3.1.1 Accuracy of the Model

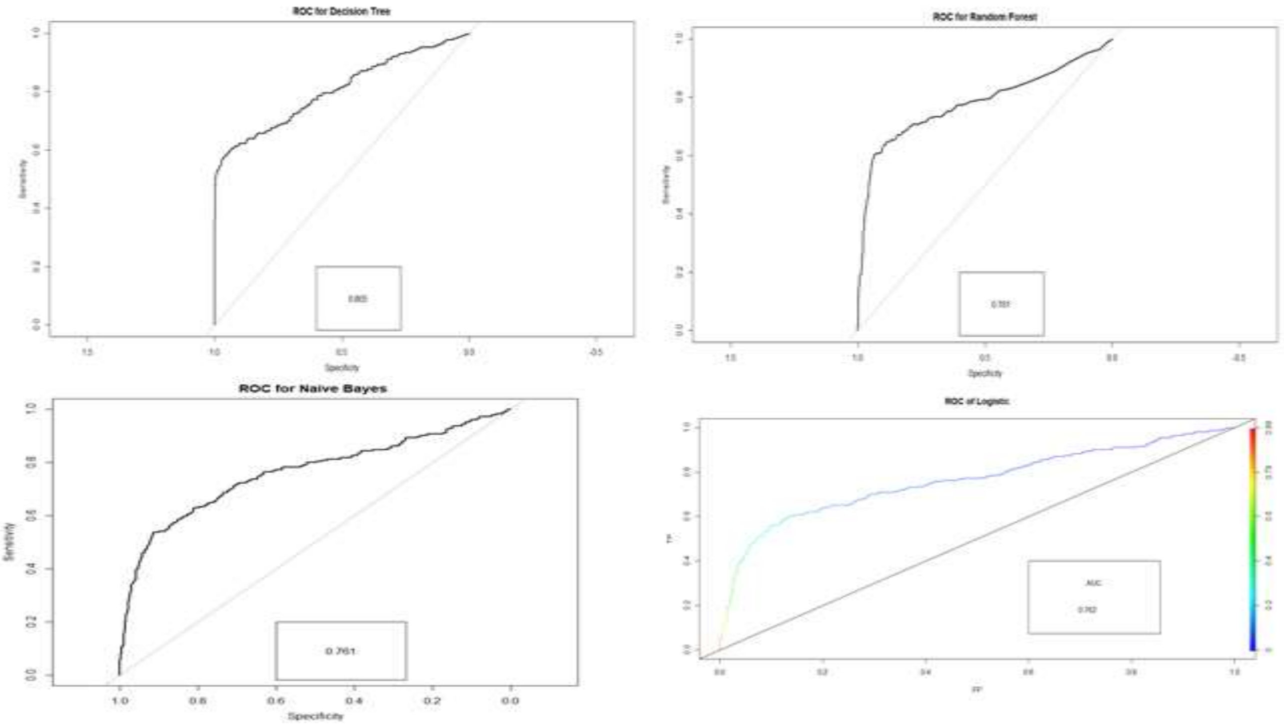


Fig 3.1.1(a) ROC Curve.

ACU of models:

- Decision Tree = 0.805
- Random Forest = 0.781
- Naïve Bayes = 0.761
- Logistic Regression = 0.762

Accuracy of Decision Tree= 91.9%

False Negative rate of Decision Tree= 59.8%

Accuracy of Random Forest = 88.12%
False negative Rate of Random Forest= 72.76%

Accuracy of Logistic Regression = 87.94%
False Negative Rate of Logistic Regression = 75.0%

Accuracy of SVM= 86.66

Accuracy of Naive Bayes= 88.12
F-N rate of Naive Bayes = 79.01

3.1.2 Models Selection

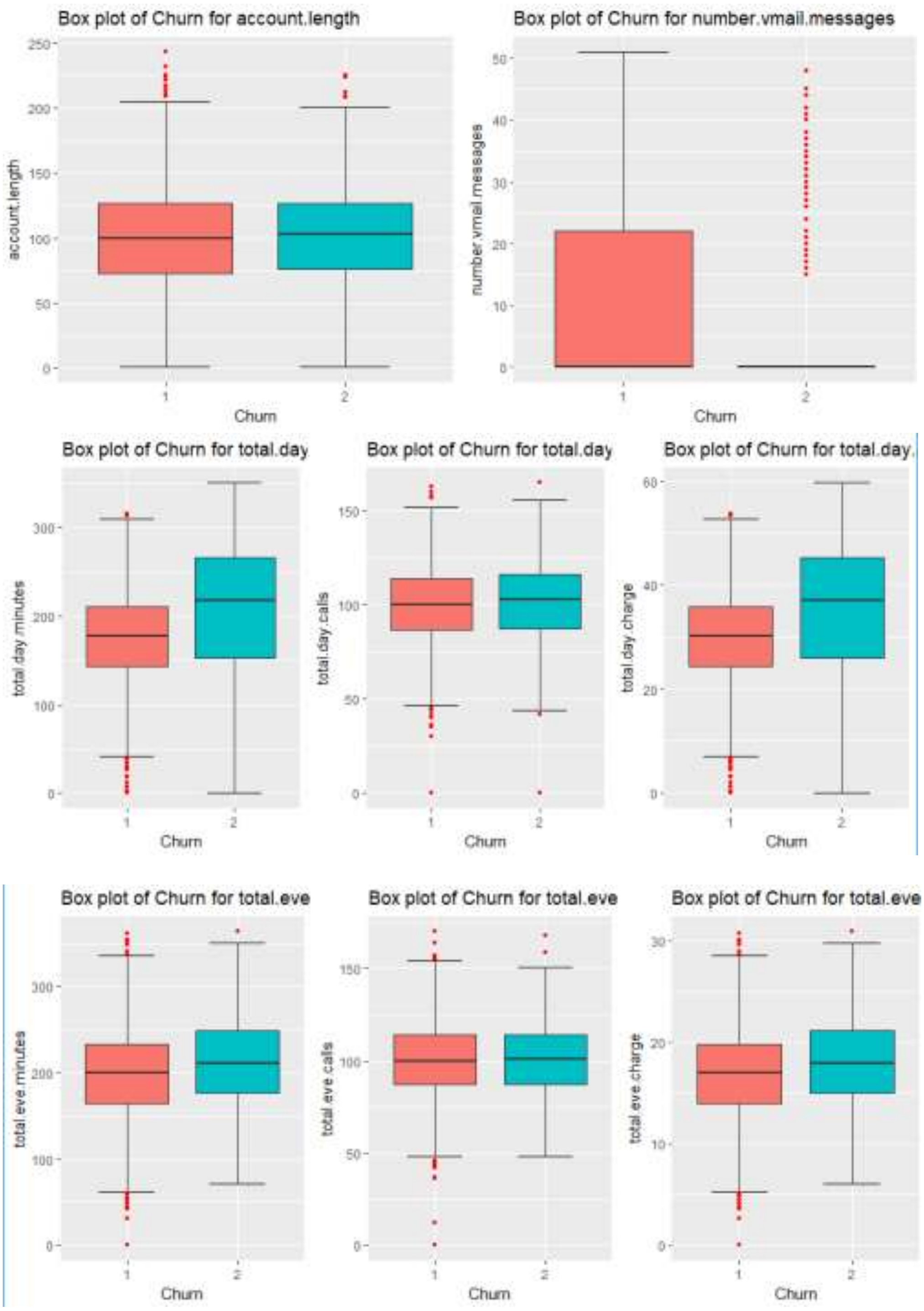
As we can see Accuracy of the Decision Tree is highest, low False Negative Rate and also Area under the curve (AUC) is highest. So Decision Tree would be the best model among all these.

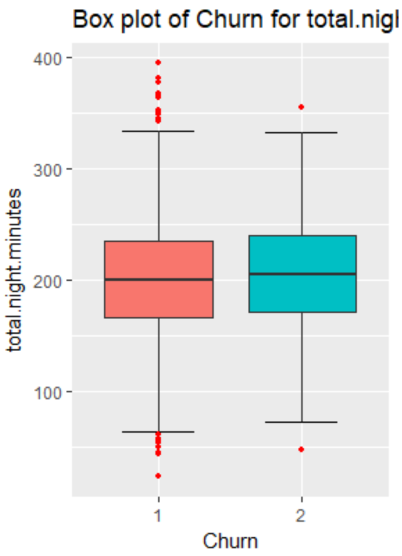
Table 3.1.2 Decision Tree predicted result

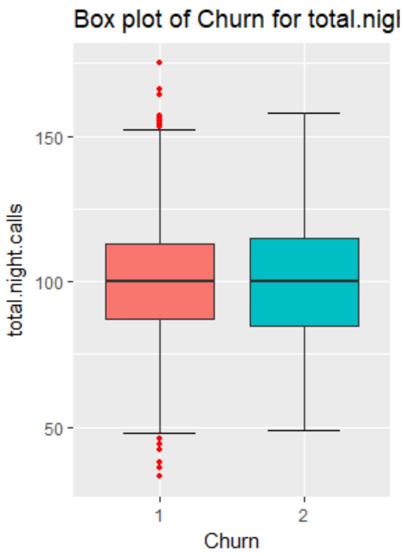
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	state	account l	area cod	phone n	internati	voice ma	number	total day	total day	total day	total eve	total eve	total eve	total nig	total nig	total nig	total intl	total intl	total intl	number	Churn	DT_Prediction_output		
2	HI	101	510	354-8815	no	no	0	70.9	123	12.05	211.9	73	18.01	236	73	10.62	10.6	3	2.86	3	False.	1		
3	MT	137	510	381-7211	no	no	0	223.6	86	38.01	244.8	139	20.81	94.2	81	4.24	9.5	7	2.57	0	False.	1		
4	OH	103	408	411-9481	no	yes	29	294.7	95	50.1	237.3	105	20.17	300.3	127	13.51	13.7	6	3.7	1	False.	1		
5	NM	99	415	418-910C	no	no	0	216.8	123	36.86	126.4	88	10.74	220.6	82	9.93	15.7	2	4.24	1	False.	1		
6	SC	108	415	413-3643	no	no	0	197.4	78	33.56	124	101	10.54	204.5	107	9.2	7.7	4	2.08	2	False.	1		
7	IA	117	415	375-618C	no	no	0	226.5	85	38.51	141.6	68	12.04	223	90	10.04	6.9	5	1.86	1	False.	1		
8	ND	63	415	348-8073	no	yes	32	218.9	124	37.21	214.3	125	18.22	260.3	120	11.71	12.9	3	3.48	1	False.	1		
9	LA	94	408	359-9881	no	no	0	157.5	97	26.78	224.5	112	19.08	310.8	106	13.99	11.1	6	3	0	False.	1		
10	MO	138	510	353-6954	no	no	0	89.1	117	15.15	126.8	46	10.78	190.5	71	8.57	9.9	4	2.67	2	False.	1		
11	TX	128	415	403-4933	no	yes	43	177.8	100	30.23	147.3	89	12.52	194.2	92	8.74	11.9	1	3.21	0	False.	1		
12	AR	113	510	360-3811	no	yes	39	209.8	77	35.67	164.1	90	13.95	159.7	100	7.19	9	4	2.43	1	False.	1		
13	TX	140	415	353-1755	no	no	0	93.2	109	15.84	197.6	116	16.8	219.8	94	9.89	10.5	2	2.84	1	False.	1		
14	ME	102	415	372-8233	no	no	0	228.1	86	38.78	156	97	13.26	227.9	124	10.26	10.6	9	2.86	1	False.	1		

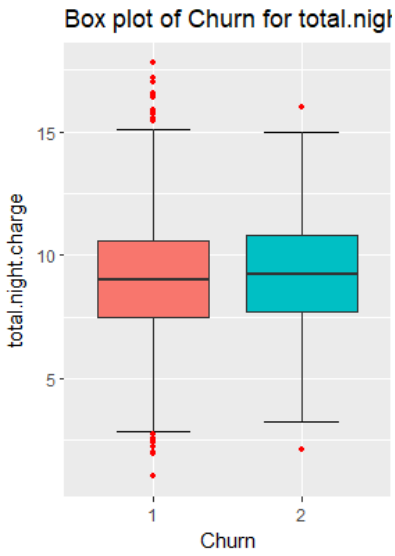
Appendix A – Extra Figure

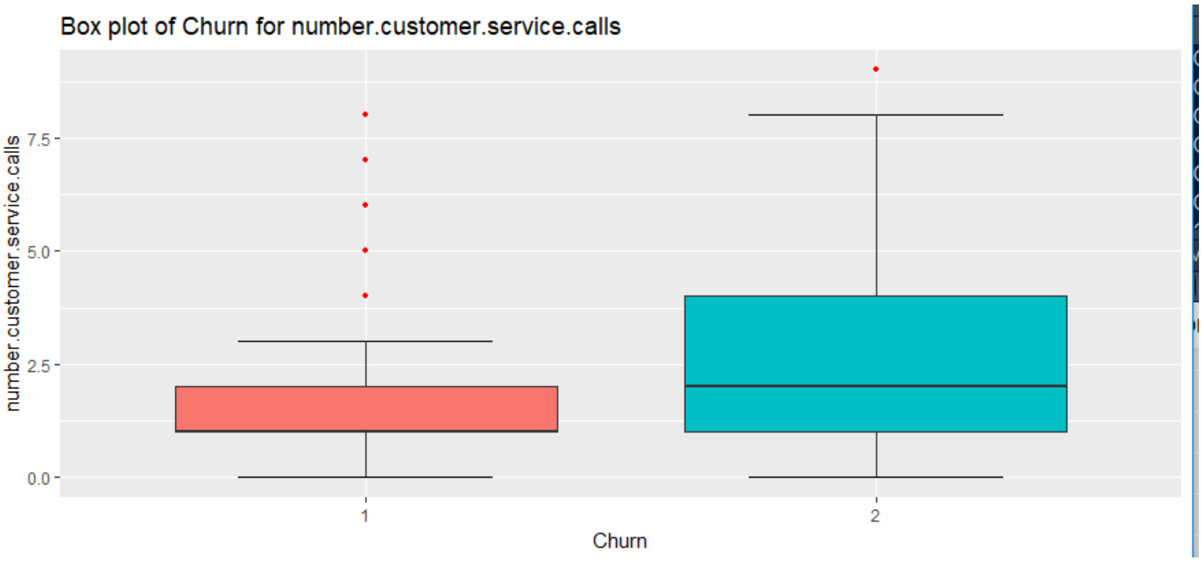
BOX Plot Before Outlier Analysis :



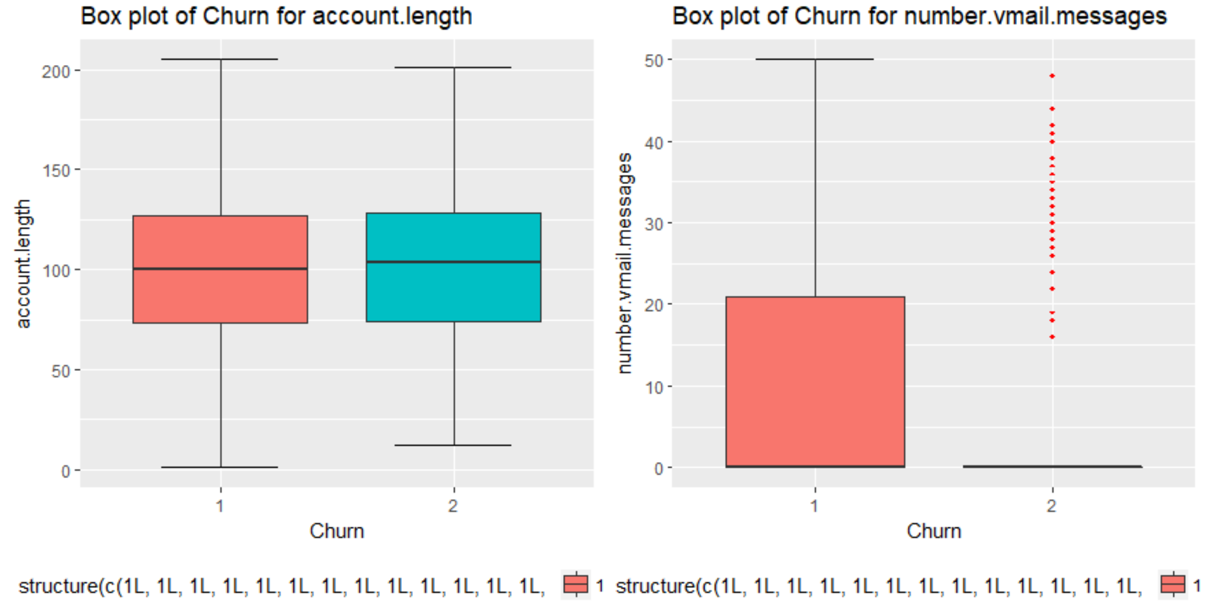

Box plot of Churn for total.nigl
total.night.minutes
400
300
200
100
1
2
Churn

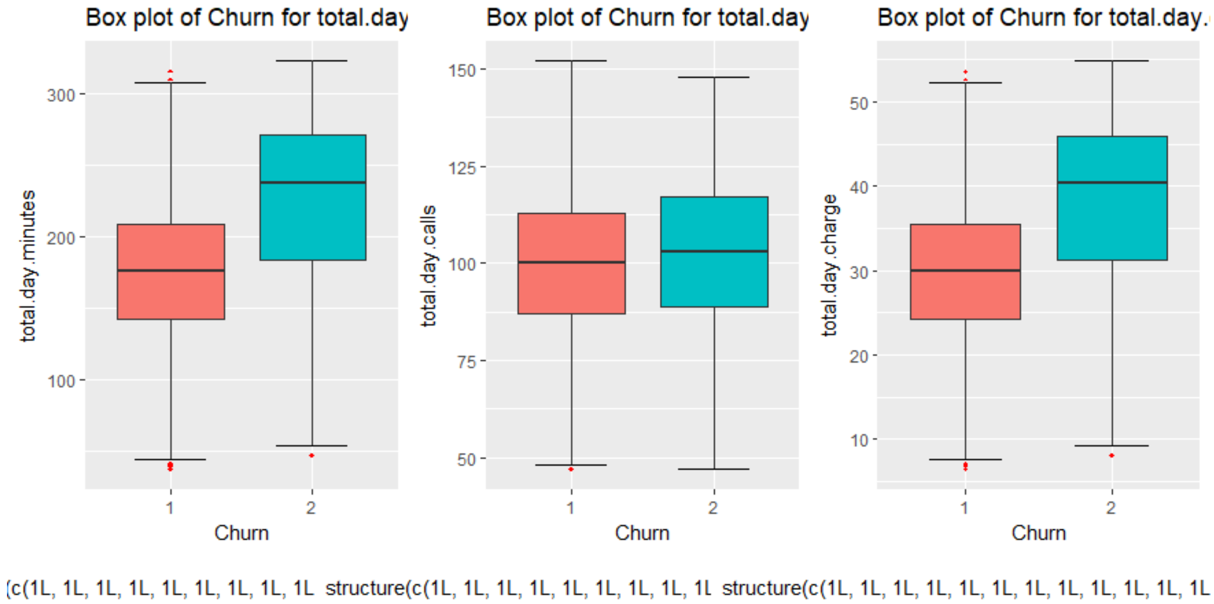

Box plot of Churn for total.nigl
total.night.calls
150
100
50
1
2
Churn

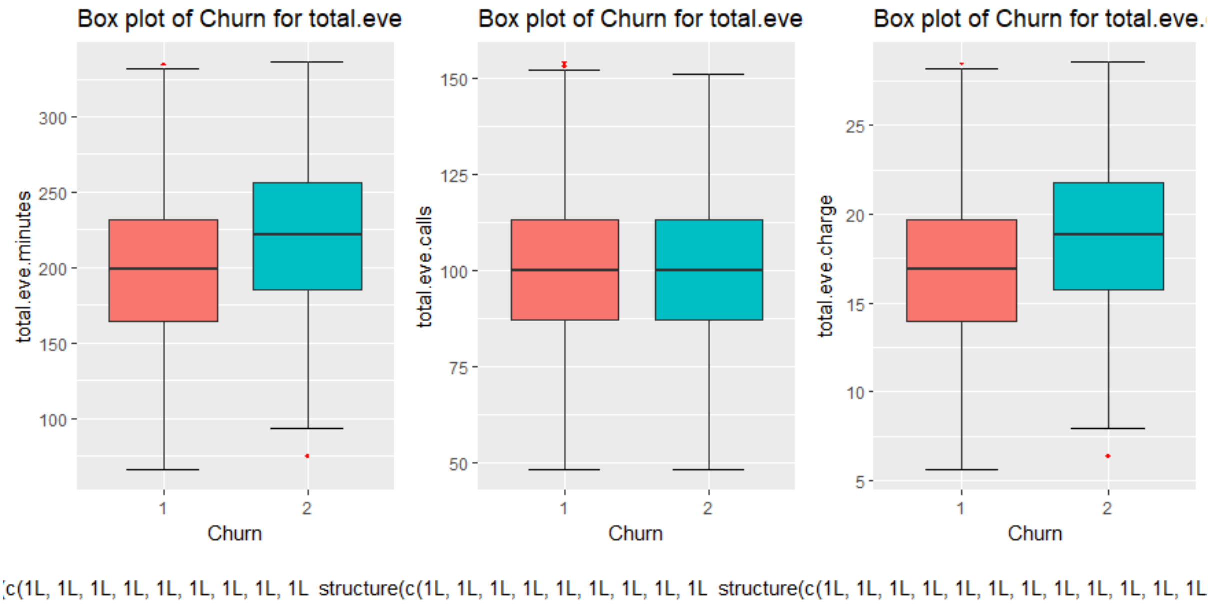

Box plot of Churn for total.nigl
total.night.charge
15
10
5
1
2
Churn

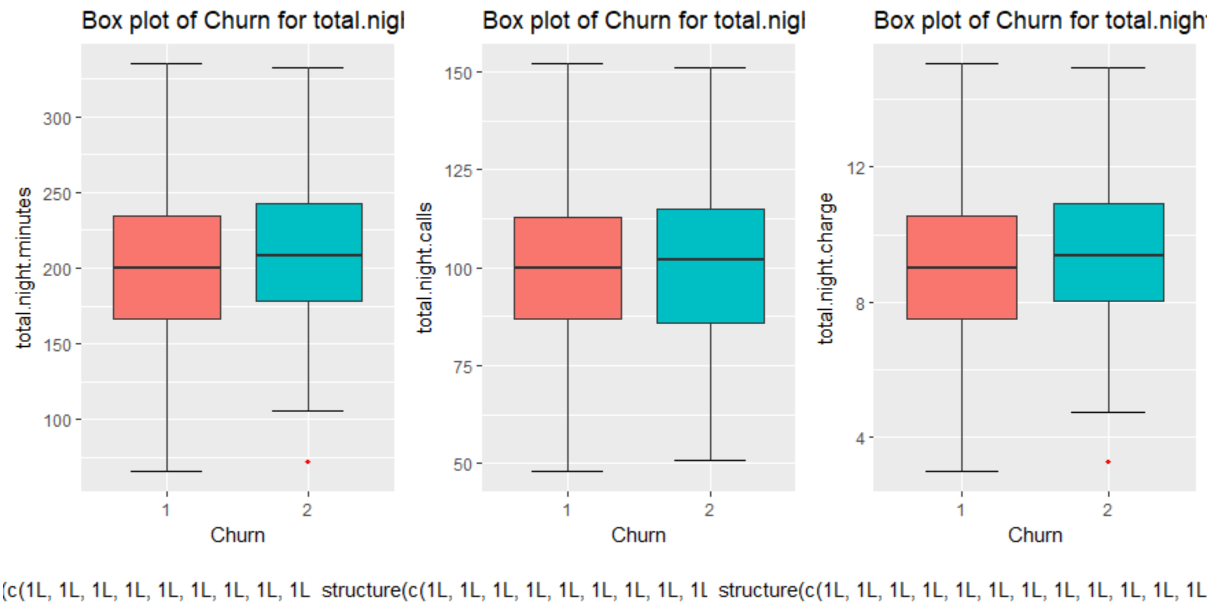

Box plot of Churn for number.customer.service.calls
number.customer.service.calls
7.5
5.0
2.5
0.0
1
2
Churn

BOX Plot After Outlier Analysis:






Box plot of Churn for total.eve
total.eve.minutes
Churn
Box plot of Churn for total.eve
total.eve.calls
Churn
Box plot of Churn for total.eve.
total.eve.charge
Churn
c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L
structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L
structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L


Box plot of Churn for total.nigl
total.night.minutes
Churn
Box plot of Churn for total.nigl
total.night.calls
Churn
Box plot of Churn for total.nigh
total.night.charge
Churn
c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L
structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1l
structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L

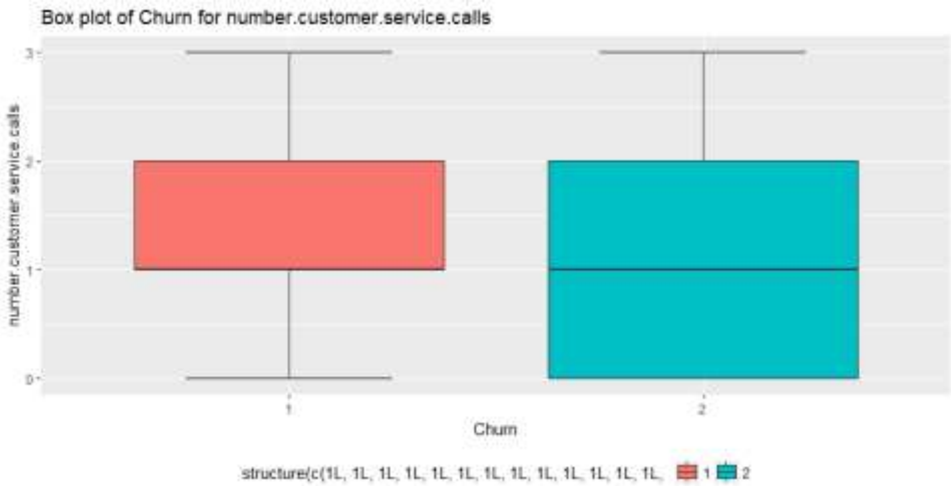
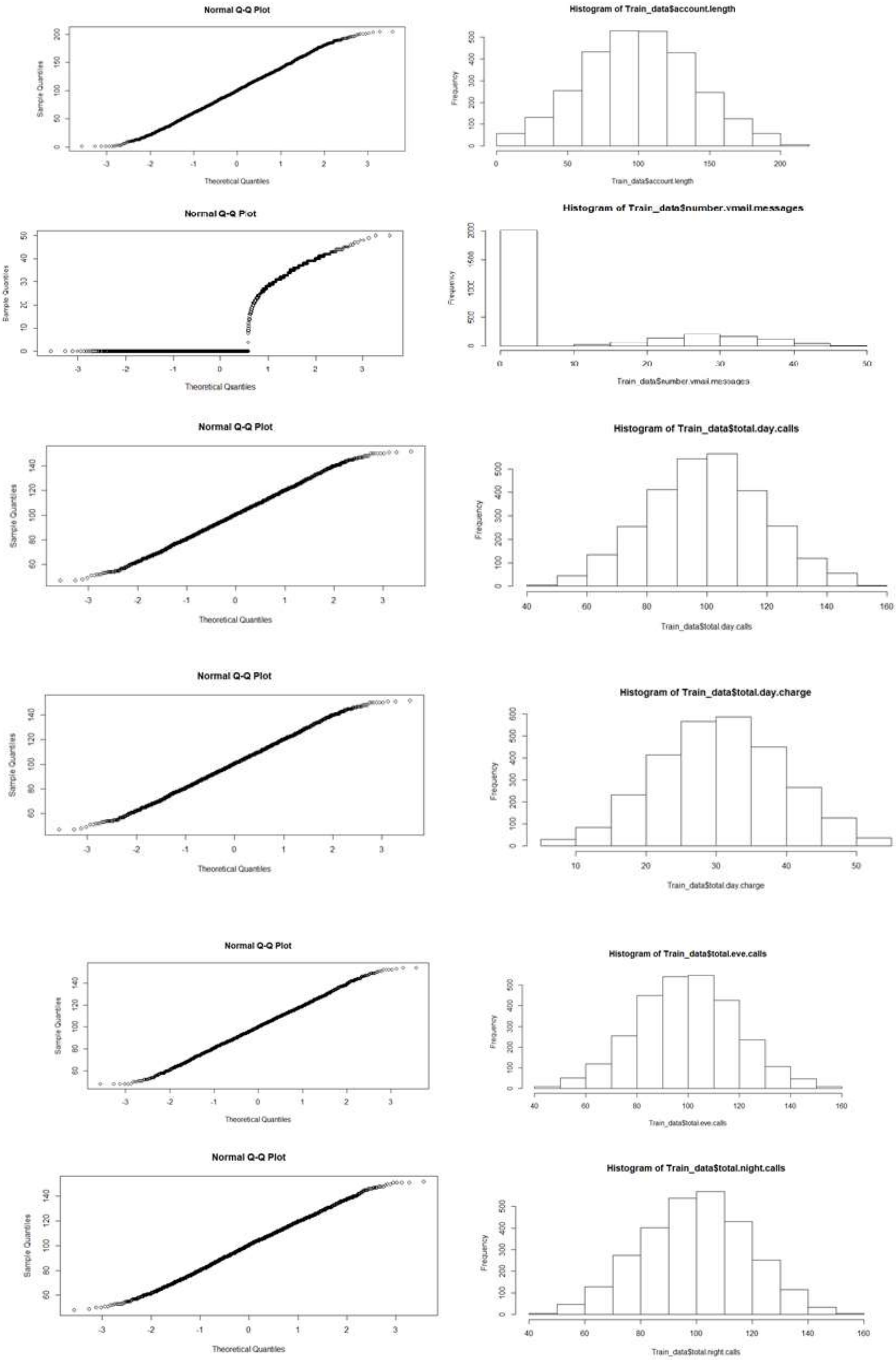
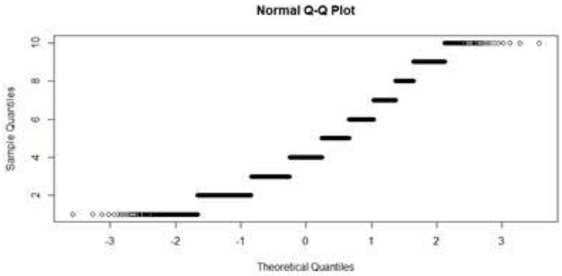
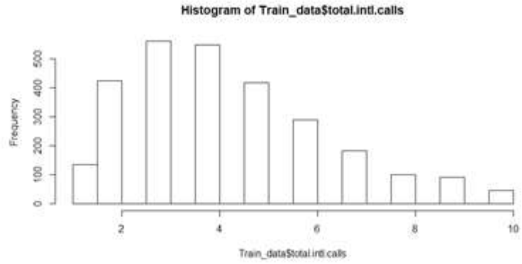
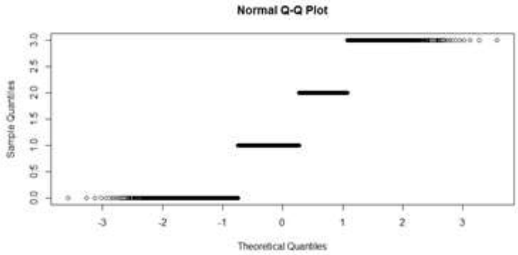

Box plot of Churn for number.customer.service.calls
number.customer.service.calls
Churn
structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
1
2

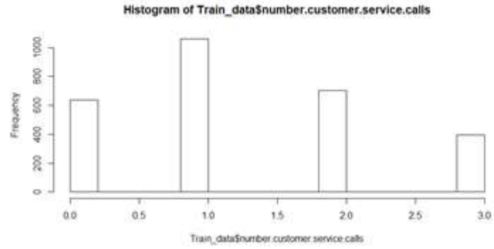
Fig: Normality plot




Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles


Histogram of Train_data$total.intl.calls
Frequency
Train_data$total.intl.calls


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles


Histogram of Train_data$number.customer.service.calls
Frequency
Train_data$number.customer.service.calls

Appendix B – R- Code

```
rm(list=ls(all=T))
setwd("O:/edWisor/Project1")
getwd()
#************************ Read the data***************************************
Train_data = read.csv("Train.csv",header = T, na.strings = c(" ", "", "NA"))
Test_data = read.csv("Test_data.csv" ,header = T, na.strings = c(" ", "", "NA"))
# ********************* Remove'-'symbol from the phone.number dataset***********
Train_data$phone.number = gsub("-","", Train_data$phone.number)
#dataset$Churn = gsub("."," ", dataset$Churn)
#************************** Check datatype ***********************************
str(Train_data)
# 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 : chr " 3824657" " 3717191" " 3581921" " 3759999" ...
# 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 ...
dim(Train_data)
head(Train_data, 5)
# Unique values in a column
# length(unique(dataset$account.length))
# 212
# table(dataset$account.length)
# mean(dataset$account.length)
```

```
# # 101.0648
#
# length(unique(dataset$number.vmail.messages))
# #46
# table(dataset$number.vmail.messages)
# mean(dataset$number.vmail.messages)
# #8.09901
# Converting Catogrical data into factor of Training dataset
Train_data$area.code = as.factor(Train_data$area.code)
for(i in 1:ncol(Train_data)){
if(class(Train_data[,i]) == 'factor'){
Train_data[,i] = factor(Train_data[,i], labels=(1:length(levels(factor(Train_data[,i])))))
}
}
# Converting Catogrical data into factor of testing dataset
Test_data$area.code = as.factor(Test_data$area.code)
for(i in 1:ncol(Test_data)){
if(class(Test_data[,i]) == 'factor'){
Test_data[,i] = factor(Test_data[,i], labels=(1:length(levels(factor(Test_data[,i])))))
}
}#
For Churn False is denoted by 1 and TRUE is denoted as 2
#************************** Missing Value for Training data ****************************
missing_val = data.frame(apply(Train_data,2,function(x){sum(is.na(x))}))
missing_val
missing_val$Columns = row.names(missing_val)
#************************** Missing Value for Testing data ****************************
missing_val = data.frame(apply(Test_data,2,function(x){sum(is.na(x))}))
missing_val
missing_val$Columns = row.names(missing_val)
#************************* Outlier Analysis **********************
# BoxPlot (Only for Numeric Variables)
library("ggplot2")
numeric_index = sapply(Train_data,is.numeric)
# Select only Numeric Data
numeric_data = Train_data[, numeric_index]
cnames = colnames(numeric_data)
for (i in 1:length(cnames))
{
assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn", fill = Train_data$Churn),
data =
```

```
subset(Train_data))+
stat_boxplot(geom = "errorbar", width = 0.5) +
geom_boxplot(outlier.colour="red" ,outlier.shape=19,
outlier.size=1, notch=FALSE) +
theme(legend.position="bottom")+
labs(y=cnames[i],x="Churn")+
ggtitle(paste("Box plot of Churn for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,gn5,ncol=3)
gridExtra::grid.arrange(gn6,gn7,gn8,ncol=3)
gridExtra::grid.arrange(gn9,gn10,gn11,ncol=3)
gridExtra::grid.arrange(gn12,gn13,gn14=3)
gridExtra::grid.arrange(gn15,ncol=1)
#****************************** Removing Outliers**********************************
df = Train_data
for(i in cnames)
{
print(i)
val = Train_data[,i][Train_data[,i] %in% boxplot.stats(Train_data[,i])$out]
#print(length(val))
Train_data = Train_data[which(!Train_data[,i] %in% val),]
}
#******************************** After removing
Outliers**********************************
for (i in 1:length(cnames))
{
assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn", fill = Train_data$Churn),
data =
subset(Train_data))+
stat_boxplot(geom = "errorbar", width = 0.5) +
geom_boxplot(outlier.colour="red" ,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])))
}
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,gn5,ncol=3)
gridExtra::grid.arrange(gn6,gn7,gn8,ncol=3)
gridExtra::grid.arrange(gn9,gn10,gn11,ncol=3)
```

```
gridExtra::grid.arrange(gn12,gn13,gn14=3)
gridExtra::grid.arrange(gn15,ncol=1)
#******************************Feature Selection*********************************
#Blue colour indicates the extrimely Positive correlation and Dark Red Colour shows
Extremely -ve correlation
library("corrgram")
corrgram(Train_data[,numeric_index], order = F,
upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
# Chose only chtogrical variables
factor_index = sapply(Train_data,is.factor)
factor_data = Train_data[,factor_index]
#Dependence between Independent and dependent variable should be high
# Idealy There should be no dependency between indepenents variables
names(factor_data)
for (i in 1:4)
# Range is 1 to 4 because our factor dataset contain 5 columns only
# among which 4 are independent variable so we need to itrate the loop only for
independent variables
{
print(names(factor_data)[i])
print(chisq.test(table(factor_data$Churn,factor_data[,i])))
}
# p<0.05 Means our dependent variable depend on Independent variable
#******************************* Dimension Reduction
*************************************
# Remove all the highly corelated data
Train_data = subset(Train_data, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,
total.intl.minutes,phone.number))
Test_data = subset(Test_data, select = -
c(total.day.minutes,total.eve.minutes,total.night.minutes,
total.intl.minutes,phone.number))
#********************************* Feature Scaling *************************
# 1) Normalization =(actual - Min. value)/(Max.value - Min.value)
#2) Standardizatoin/ Z-score
#use Z-score Only if data is normally distrubated
#Normalization
#********************************Check Normal distrubation *****************
#qqnorm is used for normality plot
qqnorm(Train_data$account.length)
hist(Train_data$account.length)
```

```
qqnorm(Train_data$number.vmail.messages)
hist(Train_data$number.vmail.messages)
qqnorm(Train_data$total.day.calls)
hist(Train_data$total.day.calls)
qqnorm(Train_data$total.day.charge)
hist(Train_data$total.day.charge)
qqnorm(Train_data$total.eve.calls)
hist(Train_data$total.eve.calls)
qqnorm(Train_data$total.night.calls)
hist(Train_data$total.night.calls)
qqnorm(Train_data$total.intl.calls)
hist(Train_data$total.intl.calls)
qqnorm(Train_data$number.customer.service.calls)
hist(Train_data$number.customer.service.calls)
cnames
#****************** Storeing all the continious variable for feature scaling *************
cnames = c("account.length","number.vmail.messages","total.day.calls","total.day.charge",
"total.eve.calls","total.eve.charge",
"total.night.calls","total.night.charge",
"total.intl.calls","total.intl.charge","number.customer.service.calls")
cnames1 = c("account.length","number.vmail.messages","total.day.calls","total.day.charge",
"total.eve.calls","total.eve.charge",
"total.night.calls","total.night.charge",
"total.intl.calls","total.intl.charge","number.customer.service.calls")
# Do normalization for continious variables
for(i in cnames)
{
print(i)
Train_data[,i] = (Train_data[,i] - min(Train_data[,i]))/
(max(Train_data[,i] - min(Train_data[,i])))
}
range(Train_data$total.eve.charge)
# Do normalization for continious variables
for(i in cnames1)
{
print(i)
Test_data[,i] = (Test_data[,i] - min(Test_data[,i]))/
(max(Test_data[,i] - min(Test_data[,i])))
}
#**************Divide data into train and test using stratified sampling method
**************
```

```
# library(caTools)
# library(caret)
#
# set.seed(1234)
# train.index = createDataPartition(Train_data$Churn, p = .80, list = FALSE)
# train = Train_data[ train.index,]
# test = Train_data[-train.index,]
# ***************************** Decision Tree *******************
library(C50)
library(caret) # For SVm and confusionMatrix
library(rpart)
library(pROC)
library(ROCR)
C50_model = C5.0(Churn ~., Train_data, trials = 100, rules = TRUE)
#Summary of DT model
summary(C50_model)
#write(capture.output(summary(C50_model)), "c50Rules_Final.txt")
#Lets predict for test cases
C50_Predictions = predict(C50_model, Test_data[,-16], type = "class")
#Test_data$DT_prediction = C50_Predictions
# write.csv(C50_Predictions,file = file.choose(new = T))
##Evaluate the performance of classification model
ConfMatrix_C50 = table(Test_data$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)
table(Train_data$Churn)
#****************** ROC Curve **********************************
DT_pred <- predict(C50_model, Test_data[-16], type = 'prob')
auc <- auc(Test_data$Churn ,DT_pred[,2])
plot(roc(Test_data$Churn, DT_pred [,2]),
main = "ROC for Decision Tree")
auc <- round(auc,3)
legend(0.6,0.2,auc)
Gini = 2*auc - 1
Gini
#Gini coefficient = 61.00%
# FNR = FN/FN+TP
# 134/(134+90)
# Accuracy = 91.9
#False Negative rate = 0.568 i.e 59.8%
#****************************** Random Forest
**************************************
```

```
library(randomForest)
RF_model = randomForest(Churn ~ ., Train_data, importance = TRUE, ntree = 1000)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, Test_data[,-16])
# y_pred1 =
predict(RF_model,c(12,0.4219,3,1,1,0.00,0.706,0.186,0.2671,0.5625,0.4294,0.6185,
# 0.1578,0.5375,0.4285))
##Evaluate the performance of classification model
ConfMatrix_RF = table(Test_data$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
# Accuracy of Decision Tree = 88.12
# False negative Rate = 72.76%
#install.packages("pROC")
#library(pROC)
RF_pred <- predict(RF_model, Test_data[-16], type = 'prob')
auc <- auc(Test_data$Churn ,RF_pred[,2])
plot(roc(Test_data$Churn, RF_pred [,2]),
main = "ROC for Random Forest")
auc <- round(auc,3)
legend(0.6,0.2,auc)
Gini = 2*auc - 1
Gini
#****************************** Logistic Regression ******************************
logit_model = glm(Churn ~ ., data = Train_data, family = "binomial")
#summary of the model
summary(logit_model)
# write(capture.output(summary(logit_model)),"logistic_Model_Summary.txt")
# predict using logistic regression
logit_Predictions = predict(logit_model, newdata = Test_data, type = "response")
# convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)
#Evaluate the performance of classification model
table(Actual = Test_data$Churn, Prediction = logit_Predictions)
# Accuracy of Logistic Regression = 87.94
# FAlse Negative Rate of Logistic Regression = 75.0
#install.packages("nnet")
library(nnet)
mymodel <- multinom(Churn ~. , data = Train_data)
# Misclassification Rate
p <- predict(mymodel,Test_data[,-16])
tab <- table(p, Test_data$Churn)
```

```
library(ROCR)
library(gplots)
logi_pred <- predict(mymodel,newdata = Test_data[,-16], type = 'prob')
logi_pred <- prediction(logi_pred, Test_data$Churn)
eval <- performance(logi_pred , "acc")
plot(eval)
# Best fit
max <- which.is.max(slot(eval,"y.values")[[1]])
acc <- slot(eval,"y.values")[[1]]
cut <- slot(eval,"x.values")[[1]]
print(c(Accuracy = acc, Cutoff = cut))
roc <- performance(logi_pred, "tpr","fpr")
plot(roc, colorize = T, main = "ROC of Logistic", ylab = "TP", xlab = "FP")
abline(a = 0, b = 1)
#AUC
auc <- performance(logi_pred ,"auc")
auc <- unlist(slot(auc,"y.values"))
auc <- round(auc, 3)
legend(0.6,0.4,auc,title = "AUC")
Gini = 2*auc - 1
Gini
# Gini = 0.524
#************************ Grid search for SVM ******************************
SVM_model = train(form = Churn ~., data = Train_data, method = 'svmRadial')
SVM_model$bestTune
y_pred = predict(SVM_model, newdata = Test_data[-16])
cm = table(Actual = Test_data$Churn, Prediction = y_pred)
ConfMatrix = table(Test_data$Churn, y_pred)
confusionMatrix(ConfMatrix)
#Accuracy of SVM= 86.66
#************************** Naive Bayes ********************************
library(e1071)
#Develop model
NB_model = naiveBayes(Churn ~ ., data = Train_data)
#predict on test cases #raw for probability
NB_Predictions = predict(NB_model, Test_data[,1:15], type = 'class')
#Look at confusion matrix
Conf_matrix = table(observed = Test_data[,16], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)
#Accuracy of Naive Bayes= 88.12
# F-N rate of Naive Bayes = 79.01
```

```
NV_pred <- predict(NB_model, Test_data[,-16], type = 'raw')
auc <- auc(Test_data$Churn ,NV_pred[,2])
plot(roc(Test_data$Churn, NV_pred [,2]),
main = "ROC for Naive Bayes ")
auc <- round(auc,3)
legend(0.6,0.2,auc)
Gini = 2*auc - 1
Gini
# Gini Coefficent = 0.522
```

References:

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

https://medium.com/