Customer Churn Prediction

Somyanath Mohanty

14 September 2018

Contents

Chapter 1

Introduction

Problem Statement

It is more expensive for a company to acquire a new customer than to keep your existing one from leaving. Churn is basically loss of customers to competition. The aim of the project is to build a system which can predict customer behaviour i.e. whether the customer will churn or not.

Data

Our task is to build a classification model which will classify whether the customer will churn or not depending on multiple factors. Given below is a sample of the dataset that we are using to predict the churn:

Table 1.1: Customer Sample Data (Columns 1-7)

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 1.2: Customer Sample Data (Columns 8-13)

total day minutes	total day calls	total day charges	total eve minutes	total eve calls	total eve charges
265.1	110	45.07	197.4	99	16.78
161.6	123	27.47	195.5	103	16.62
243.4	114	41.38	121.2	110	10.3
299.4	71	50.90	61.9	88	5.26
166.7	113	28.34	148.3	122	12.61

Table 1.3: Customer Sample Data (Columns 14-19)

total night minutes	total night calls	total night charges	total intl minutes	total intl calls	total intl charges
244.7	91	11.01	10	3	2.7
254.4	103	11.45	13.8	3	3.7
162.6	104	7.32	12.2	5	3.29
196.9	89	8.86	6.6	7	1.78
186.9	121	8.41	10.1	3	2.73

Table 1.4: Customer Sample Data (Columns 20-21)

number customer service calls	Churn
1	91
1	103
0	104
2	89
3	121

As you can see in the table below we have the following 20 variables, using which we have to correctly predict the customer churn:

Table 1.5: Predictor Variables

S.No.	Predictor
1	account length
2	area code
3	phone number
4	international plan
5	voice mail plan
6	number vmail messages
7	total day minutes
8	total day calls
9	total day charge
10	total eve minutes
11	total eve calls
12	total eve charge
13	total night minutes
14	total night calls
15	total night charge
16	total intl minutes
17	total intl calls
18	total intl charge
19	number customer service calls
20	Churn

Chapter 2

Methodology

Pre-Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called **Exploratory Data Analysis**. To start this process, we will first drop and modify some variables just by observing.

```
# Drop the unimportant variables which does not contribute to the model at all
# As, phone number does not contribute any significant information to the
# prediction model at all
customer_churn$phone.number = NULL

# Convert the area code from integer class to factor
# As, there are only 3 areas so it is better to convert the variable's data
# in the form of a factor as it improves the efficiency of the code
customer_churn$area.code = factor(customer_churn$area.code)
```

Here, "*phone number*" is a variable which is unique for every other customer. So, there is no significant information in it regarding the Churn. Also, we can modify the variable "*area code*" into factor as it has only 3 values. So instead of treating it as number we can convert it into factor and improve the efficiency of the code a bit.

Now, we can check the histograms of all the numerical variables in order to be sure about the normality of the variables. So, we can plot the probability density functions of the numerical variables.

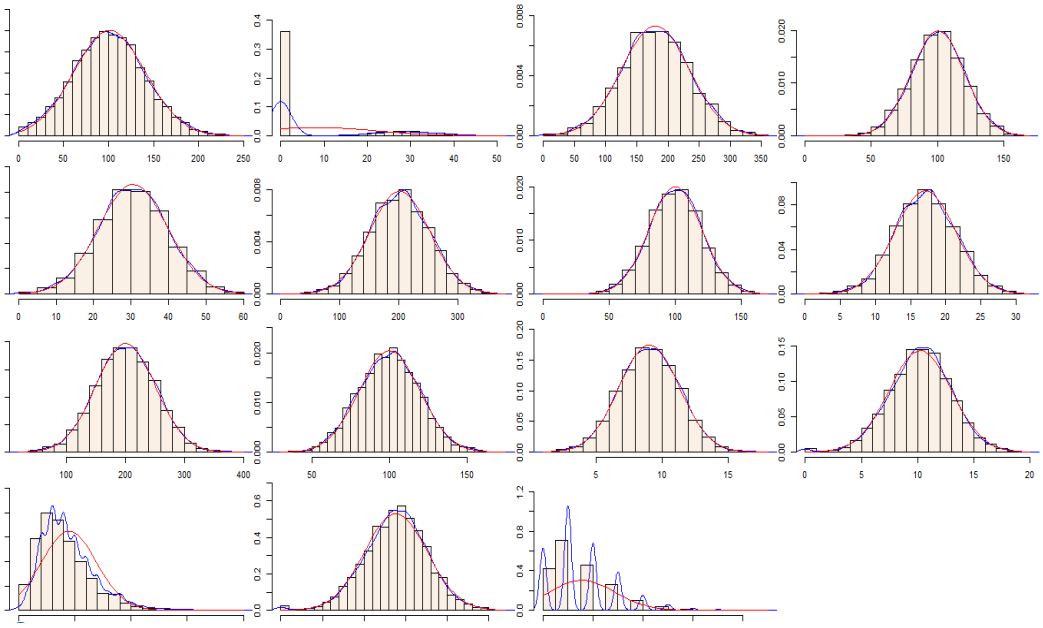


Fig 2.1: Multiple Probability Density Function Plots

In fig 2.1, the red lines represent the normal distribution. So, as you can see in the figure most of the variables are normally distributed. But few of them are skewed like “number vmail messages”, “total international calls” and “number customer service calls”.

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 can opt for correlational analysis.

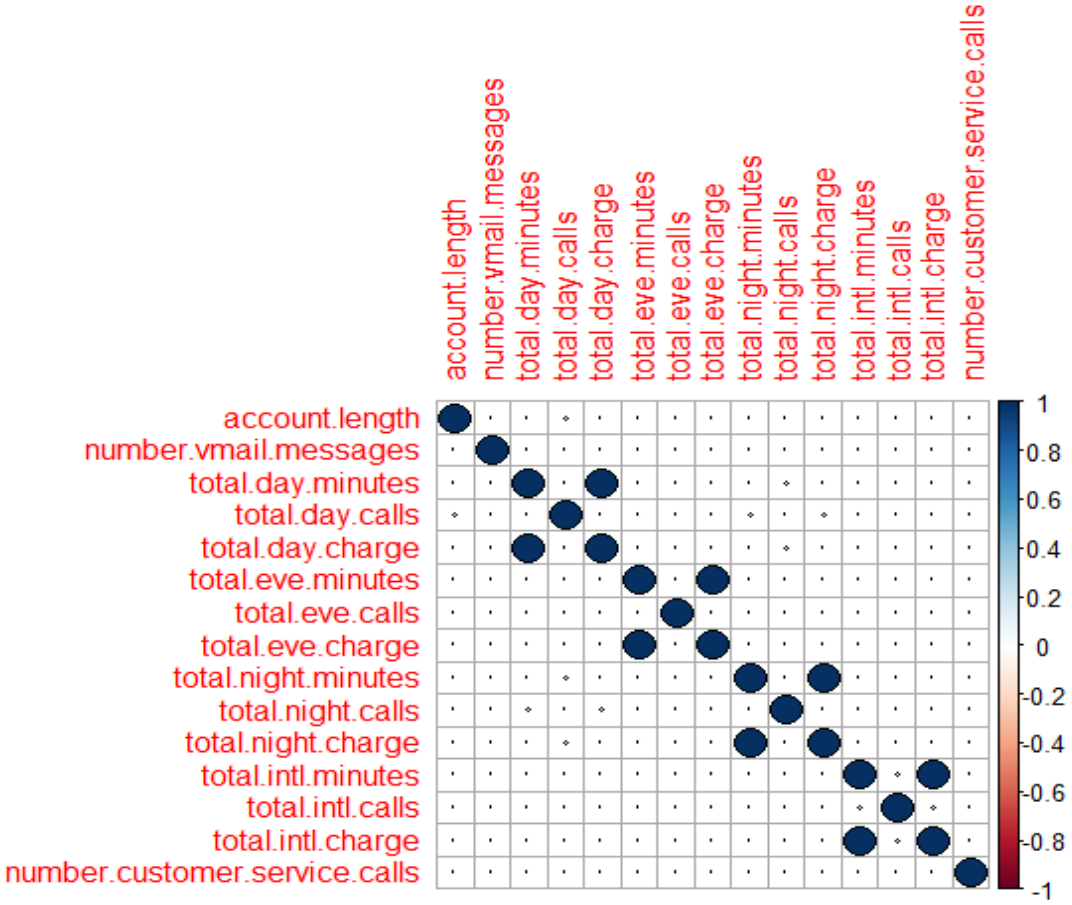


Fig 2.1.1: Correlation Analysis Plot

```
# Creating the correlation analysis plot to analyse the dependecies of numeric
# variables
library(corrplot)
cor_plot = cor(customer_churn[,sapply(customer_churn, is.numeric)])
corrplot(cor_plot, method = "circle", addgrid.col = "darkgray",
         outline = TRUE, number.digits = 2)
#The correlation plot tells us that the numerical variables where there is
#"minutes" and "charges" are correlated. So, we can drop one of the two from every pair of variables
```

Fig 2.1.2: Code Snippet of Correlation Analysis

```r
# Now we check the churn with international plan subscribers
intlchurn_count = count(customer_churn, c('international.plan', 'Churn'))
# percentage of international plan subscribers and their false churn
paste0("International Plan and False Churn ", (intlchurn_count[3,3]/sum(intlchurn_count[c(3,4),3]))*100, " %")

# percentage of international plan subscribers and their true churn
paste0("International Plan and False Churn ", (intlchurn_count[4,3]/sum(intlchurn_count[c(3,4),3]))*100, " %")

# percentage of non-international plan subscribers and their false churn
paste0("Non-International Plan and False Churn ", (intlchurn_count[1,3]/sum(intlchurn_count[c(1,2),3]))*100, " %")

# percentage of non-international plan subscribers and their true churn
paste0("Non-International Plan and True Churn ", (intlchurn_count[2,3]/sum(intlchurn_count[c(1,2),3]))*100, " %")
# According to the findings, the churn is very high when the customer is a international
# plan subscriber i.e. above 40%.
# So, the International Plan is a very important variable in determining the churn
```

Fig 2.1.2: Code Snippet for International Plan variation with Churn

One more method of checking the dependency between the categorical variables is **Chi-Square Test**.

```r
#Chi-Squared test of independence
factor_index = sapply(customer_churn, is.factor)
#In the previous we selected only the categorical variables

factor_data = customer_churn[,factor_index]

for (i in 1:5){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn, factor_data[,i])))
}
# According, to the chi-square test the "area.code" variable is not significant and can be
# rejected as the p-value is very high.

#Dimension reduction
customer_churn_del = subset(customer_churn,
                            select = -c(area.code, total.day.charge, total.eve.charge, total.night.charge,
                                        total.intl.charge))
```

Fig 2.1.3: Code Snippet for Chi-Square Test

```
[1] "state"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 83.044, df = 50, p-value = 0.002296

[1] "area.code"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 0.17754, df = 2, p-value = 0.9151

[1] "international.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 222.57, 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 = 34.132, df = 1, p-value = 5.151e-09

[1] "Churn"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 3324.9, df = 1, p-value < 2.2e-16

Warning message:
In chisq.test(table(factor_data$Churn, factor_data[, i])) :
  Chi-squared approximation may be incorrect
```

Fig 2.1.4: Chi-Square Test result

In figure 2.1.1 we have plotted the correlation analysis plot of all the numerical variables. The plot shows that there is a dependency between every pair of "minutes" and "charges" variable. So, we can drop one of the two from every pair.

We can also check the variation of the data between the predictor numerical variables and our target categorical variable. In fig 2.1.2 our check asserts that the international plan variable is very important and does have significant information regarding the variation of Churn.

Also, according to chi-square test we can reject the "area code" variable as the p-value is very high.

Modelling

Model Selection

The dependant variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependant variable, in our case is "Churn", is Nominal the only predictive analysis that we can perform is **Classification**, and if the dependant variable is Interval or Ratio the normal method is to do a **Regression** analysis, or classification after binning. But the dependant variable we are dealing with is *Binomial* (type of Nominal), for which we'll be doing classification.

Here we'll start our model building from the simplest to more complex model. So, for our classification problem the first model will be logistic regression.

Logistic Regression

```
##### LOGISTIC REGRESSION #####
# Logistic Regression
# glm is built-in function which helps us to build logistic regression on top of the dataset
LR_model = glm(Churn ~ ., data = train, family = "binomial"(link="logit"))

#summary of the model
summary(LR_model)
# The logistic regression test's result suggests us that the most important variables are
# "international.plan - yes", "total.day.minutes", "total.eve.minutes", "total.intl.minutes"
# "number.customer.service.calls" follwed by "total.night.minutes" and "voice.mail/plan - yes"

#anova
anova(LR_model, test = "Chisq")
# Here, we can analyse the drop in the deviance when adding each variable. "international.plan"
# and "number.customer.service.calls" followed by "total.day.minutes" significantly reduces the
# residual deviance. But, the variables "voice.mail.plan", "total.eve.minutes","total.intl.minutes"
# seems to improve the model very less even thought they all have two p-values
```

Fig 2.3.2.1: Code Snippet for Logistic Regression

```r
test$Churn = as.character(test$Churn)
test$Churn[test$Churn == " True."] = "1"
test$Churn[test$Churn == " False."] = "0"

#predict using logistic regression
#Type = response gives us the probabilities
LR_predictions = predict(LR_model, newdata = test, type = "response")

#Convert probabilities
#Converting the probabilities into classes 1 & 0
LR_predictions = ifelse(LR_predictions > 0.5, 1, 0)

misClassifierError = mean(LR_predictions != test$Churn)

print(paste('Logistic Regression Accuracy', 1 - misClassifierError))

#Evaluate the performance of the classification model
ConfMatrix_LR = table(test$Churn, LR_predictions > 0.5)
View(ConfMatrix_LR)
#Accuracy = 87.16%
output = cbind(test, LR_predictions)
write.csv(output,"LR_Output.csv", row.names = F)
```

Fig 2.3.2.2: Code Snippet for Logistic Regression Prediction

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9298  -0.4991  -0.3117  -0.1654   3.0532

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -9.7669077  0.9759178 -10.008  < 2e-16 ***
stateAL                  0.3492010  0.7633606   0.457 0.647346
stateAR                  0.9200442  0.7527457   1.222 0.221613
stateAZ                  0.1334899  0.8440299   0.158 0.874333
stateCA                  1.8328524  0.7823563   2.343 0.019143 *
stateCO                  0.6898136  0.7618690   0.905 0.365241
stateCT                  1.0326011  0.7259443   1.422 0.154903
stateDC                  0.7048030  0.8090440   0.871 0.383669
stateDE                  0.7668968  0.7493844   1.023 0.306133
stateFL                  0.5985951  0.7614983   0.786 0.431823
stateGA                  0.6788217  0.7783502   0.872 0.383138
stateHI                 -0.2018366  0.8954250  -0.225 0.821661
stateIA                  0.2336517  0.9027984   0.259 0.795783
stateID                  0.8833513  0.7475505   1.182 0.237340
stateIL                 -0.1949848  0.8326624  -0.234 0.814853
stateIN                  0.4530439  0.7533107   0.601 0.547571
stateKS                  1.0794222  0.7299408   1.479 0.139199
stateKY                  0.8198837  0.7649873   1.072 0.283827
stateLA                  0.5731898  0.8358993   0.686 0.492892
stateMA                  1.1849778  0.7435629   1.594 0.111015
stateMD                  1.1532706  0.7167221   1.609 0.107597
stateME                  1.3555648  0.7281491   1.862 0.062651 .
stateMI                  1.3986613  0.7133687   1.961 0.049921 *
stateMN                  1.1717740  0.7158869   1.637 0.101669
stateMO                  0.6099525  0.7741786   0.788 0.430772
stateMS                  1.3644687  0.7278053   1.875 0.060824 .
stateMT                  1.8803754  0.7167764   2.623 0.008706 **
stateNC                  0.6192720  0.7539849   0.821 0.411457
stateND                  0.1645378  0.7968298   0.206 0.836408
stateNE                  0.3352039  0.8052295   0.416 0.677202
stateNH                  1.2024379  0.7677747   1.566 0.117317
```

```
stateNJ                        1.6001917  0.7094790   2.255 0.024105 *
stateNM                        0.4887959  0.7868408   0.621 0.534459
stateNV                        1.2607016  0.7253495   1.738 0.082200 .
stateNY                        1.1794006  0.7198730   1.638 0.101350
stateOH                        0.7011917  0.7463404   0.940 0.347471
stateOK                        0.8874962  0.7542691   1.177 0.239343
stateOR                        0.7851522  0.7361140   1.067 0.286145
statePA                        1.1683048  0.7795473   1.499 0.133952
stateRI                       -0.1069775  0.8200324  -0.130 0.896206
stateSC                        1.7841508  0.7372629   2.420 0.015522 *
stateSD                        0.8353637  0.7621829   1.096 0.273072
stateTN                        0.2910411  0.8209935   0.354 0.722965
stateTX                        1.6629143  0.7079187   2.349 0.018823 *
stateUT                        1.0517529  0.7441842   1.413 0.157569
stateVA                       -0.4255407  0.8234364  -0.517 0.605305
stateVT                        0.1096876  0.7778714   0.141 0.887862
stateWA                        1.4356744  0.7241044   1.983 0.047402 *
stateWI                        0.2904388  0.7805840   0.372 0.709834
stateWV                        0.5889018  0.7335685   0.803 0.422096
stateWY                        0.3076557  0.7550166   0.407 0.683654
account.length                 0.0009985  0.0014337   0.696 0.486127
international.plan yes         2.1838009  0.1529433  14.278  < 2e-16 ***
voice.mail.plan yes           -2.1045683  0.5931401  -3.548 0.000388 ***
number.vmail.messages          0.0373885  0.0186144   2.009 0.044582 *
total.day.minutes              0.0131222  0.0011088  11.834  < 2e-16 ***
total.day.calls                0.0040276  0.0028574   1.410 0.158669
total.eve.minutes              0.0077660  0.0011838   6.560 5.38e-11 ***
total.eve.calls                0.0009977  0.0028872   0.346 0.729664
total.night.minutes            0.0039335  0.0011516   3.416 0.000636 ***
total.night.calls              0.0001597  0.0029237   0.055 0.956435
total.intl.minutes             0.0836664  0.0210892   3.967 7.27e-05 ***
total.intl.calls              -0.0900820  0.0256879  -3.507 0.000454 ***
number.customer.service.calls  0.5362351  0.0409086  13.108  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2758.3  on 3332  degrees of freedom
Residual deviance: 2072.2  on 3269  degrees of freedom
AIC: 2200.2

Number of Fisher Scoring iterations: 6
```

Fig 2.3.2.3: Output of the summary of the Logistic Regression

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: Churn

Terms added sequentially (first to last)


                          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)

NULL                                       3332     2758.3

state                     50   83.184      3282     2675.1 0.0022252 *
*
account.length             1    1.221      3281     2673.9 0.2691078

international.plan         1  175.740      3280     2498.2 < 2.2e-16 *
**
```

```
voice.mail.plan              1   43.206       3279     2454.9 4.926e-11 *
**
number.vmail.messages        1    3.772       3278     2451.2 0.0521245 .

total.day.minutes            1  130.448       3277     2320.7 < 2.2e-16 *
**
total.day.calls              1    1.316       3276     2319.4 0.2512753

total.eve.minutes            1   31.714       3275     2287.7 1.786e-08 *
**
total.eve.calls              1    0.177       3274     2287.5 0.6737787

total.night.minutes          1    8.936       3273     2278.6 0.0027966 *
*
total.night.calls            1    0.027       3272     2278.6 0.8684015

total.intl.minutes           1   12.492       3271     2266.1 0.0004087 *
**
total.intl.calls             1   13.665       3270     2252.4 0.0002185 *
**
number.customer.service.calls  1 180.146       3269     2072.2 < 2.2e-16 *
**
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Fig 2.3.2.4: Output of Anova Chi Square test over the Logistic Regression Model

In fig 2.3.2.3, the logistic regression test's result suggests us that the most important variables are "*international.plan - yes*", "*total.day.minutes*", "*total.eve.minutes*", "*total.intl.minutes*" "*number.customer.service.calls*" followed by "*total.night.minutes*" and "*voice.maiL.plan - yes*".

In fig 2.3.2.4, we can analyse the drop in the deviance when adding each variable. "*international.plan*" and "*number.customer.service.calls*" followed by "*total.day.minutes*" significantly reduces the residual deviance. But, the variables "*voice.mail.plan*", "*total.eve.minutes*","*total.intl.minutes*" seems to improve the model very less even though they all have low p-values.

Var1	Var2	Freq	
1	0	FALSE	1397
2	1	FALSE	168
3	0	TRUE	46
4	1	TRUE	56

Fig 2.3.2.5: Contingency Table for Logistic Regression Predictions

According to the contingency table in fig 2.3.2.5 we can have found that the accuracy of our model is coming out to be 87.16% which is quite good. But considering our significant error metric to be False Negative Rate, it is coming out to be 75% which is very high.

So, we will opt for some other algorithm where we can find the error metric to be less.

Naïve Bayes

```
##### NAIVE BAYES #####
library(e1071)

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

#Predict the test cases
#type = class will internally divide the whole extracted probabilities into yes or no
#if we want the probabilities we can write type = raw
NB_predictions = predict(NB_model, test[,1:14], type = 'class')

#Look at the confusion matrix
ConfMatrix_NB = table(observed = test[,15], predicted = NB_predictions)

View(ConfMatrix_NB)
#Accuracy = 88.12%
output = cbind(test, NB_predictions)
write.csv(output,"NB_Output.csv", row.names = F)
```

Fig 2.3.3.1: Code Snippet for Naïve Bayes

	observed	predicted	Freq
1	0	False.	1405
2	1	False.	160
3	0	True.	38
4	1	True.	64

Fig 2.3.3.2: Contingency table of Naïve Bayes Predictions

According to the contingency table in fig 2.3.3.2 we can have found that the accuracy of our model is coming out to be 88.12% which is a slight improvement than our logistic regression model but the False Negative Rate is coming out to be 71.42% which is still very high.

So, we will opt for some other algorithm where we can find the error metric to be less.

Decision Trees

```
############ DECISION TREES ##########

#Decision Tree for classification
#Develop Model on training data
#1st arg is of the target var, here "Churn ~." full stop implies that except responded
#all the other variables are to be considered independent, we can also write the var
#names followed by + explicitly. 2nd arg is of the dataset. 3rd arg trails = 100
#implies that the compiler needs to build 100 trees one by one and select the best
#tree out of all trees. 4th arg rules = TRUE implies that the compiler needs
#to exctract the business rules out of the decision tree.
c50_model = C5.0(Churn ~., train, trials = 100, rules = TRUE)

#write rules into disk
write(capture.output(summary(c50_model)), "c50Rules.txt")

#Lets predict the test cases
#Predict is a function which helps in predicting the new tests cases with the help of
#existing model. 1st arg is of the mdoel name. 2nd arg is the reference to the data but here we
#want to remove the target variable so for that [,-17]. 3rd arg type = class gives the
#class to which the target variable comes out to be, means here or target variable
#is binary so for that we need to predict the test cases also in the form of binary numbers
c50_predictions = predict(c50_model, test[,-15], type = "class")

#Now all we have to do is compare the predicted value with the actual values of the target
#variable, then we have to look at how the variable can able to perform on the test data
#it mean that how accurate the model is.

#Now as the the target variable is categorical we can't do regression here as for regression
# we need a target variable which is continuos.

#Evaluate the performance of the classification model
#Here we are building a contingency table(confusion matrix). 1st arg is of the
#actual values. 2nd arg is of the predcited values
confMatrix_c50 = table(test$Churn, c50_predictions)

#Here now we decide on the basis of the business problem that what is the best error metric
#Accuracy = 95.80%
#FNR = 29.01%
output = cbind(test, c50_predictions)
write.csv(output,"c50_Output.csv", row.names = F)
```

Fig 2.3.4.1: Code Snippet for Decision Tree

1	0	False.	1438
2	1	False.	65
3	0	True.	5
4	1	True.	159

Fig 2.3.4.2: Contingency table of Decision Tree Predictions

According to the contingency table in fig 2.3.4.2 we can found that the accuracy of our model is coming out to be 95.80% which is very high than any of our previous models. Also, the False Negative Rate is coming out to be 29.01% which is very less as compared to our previous models but still it is considered to be quite significant.

So, we will opt for some other algorithm where we can find the error metric to be less.

Random Forest

```
############ RANDOM FOREST CLASSIFIER #############

#Here we want to reduce the error rate
#RANDOM FOREST
#1st arg is of the target variable and saying that all the remaining variables are indep
#2nd arg is of dataset. 3rd arg importance = true suggests that apart from giving me
#output predictions laso give me the important variables. 4th arg is no. of trees.
#In this 1st take 100 trees, then take 500 and see if it improves the model.
RF_model = randomForest(Churn ~., train, importance = TRUE, ntree = 500)
print(RF_model)
plot(RF_model)

#Now, predict the test data using the random forest model
#1st arg is fo the RF model. 2nd arg is of the reference to the independent vars
RF_Predictions = predict(RF_model, test[,-15])

#Evaluate the performance of the classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
#Accuracy = 94.60%
#FNR = 30.35%
output = cbind(test, RF_Predictions)
write.csv(output,"RF_Output.csv", row.names = F)
```

Fig 2.3.5.1: Code Snippet for Random Forest Classifier

1	0	False.	1423
2	1	False.	68
3	0	True.	20
4	1	True.	15

Fig 2.3.5.2: Contingency table of Random Forest Classifier Predictions

According to the contingency table in fig 2.3.5.2 we can found that the accuracy of our model is coming out to be 96.60% which is slightly less than our decision trees' accuracy. The False Negative Rate is coming out to be 30.35% which is very less as compared to our previous models but still it is considered to be quite significant.

Chapter 3

Conclusion

Model Evaluation

Now that we have four models to predict 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 using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In Our case of Customer Churn Prediction, the latter two, Interpretability and Computational Efficiency, do not hold much significance as the most important aspect in our business problem is to predict the churn and modify the factors which in turn results in more churn, not to build a model which can do faster predictions or interpret what has been done already.

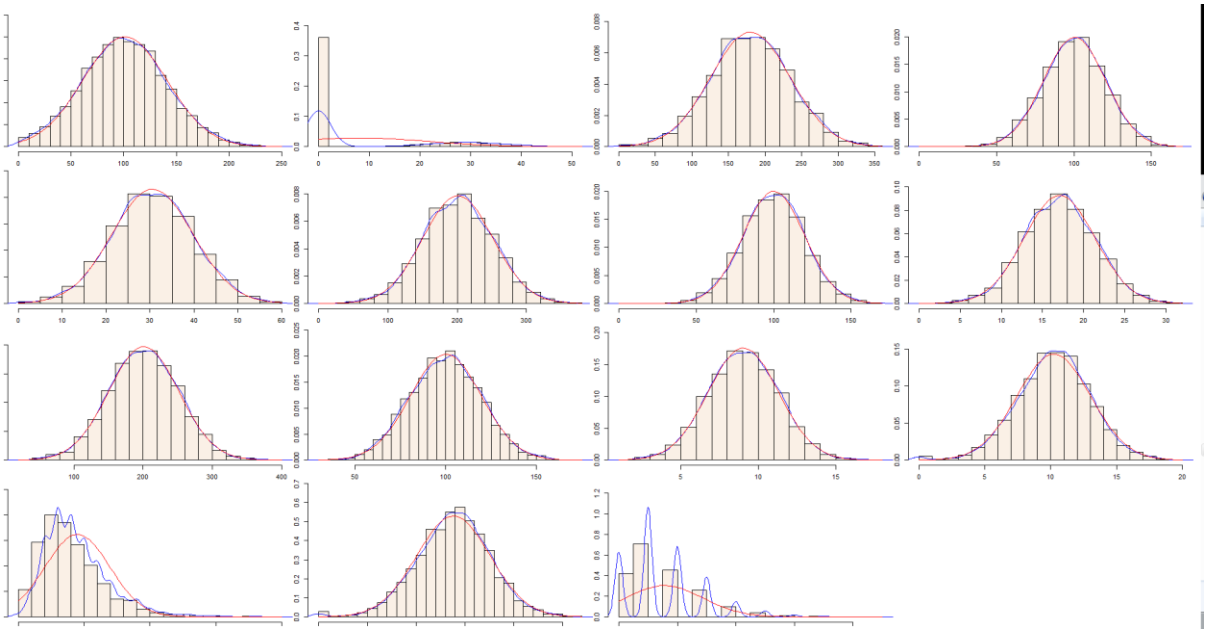
Therefore, we will use Predictive Performance to compare the two models.

Predictive Performance

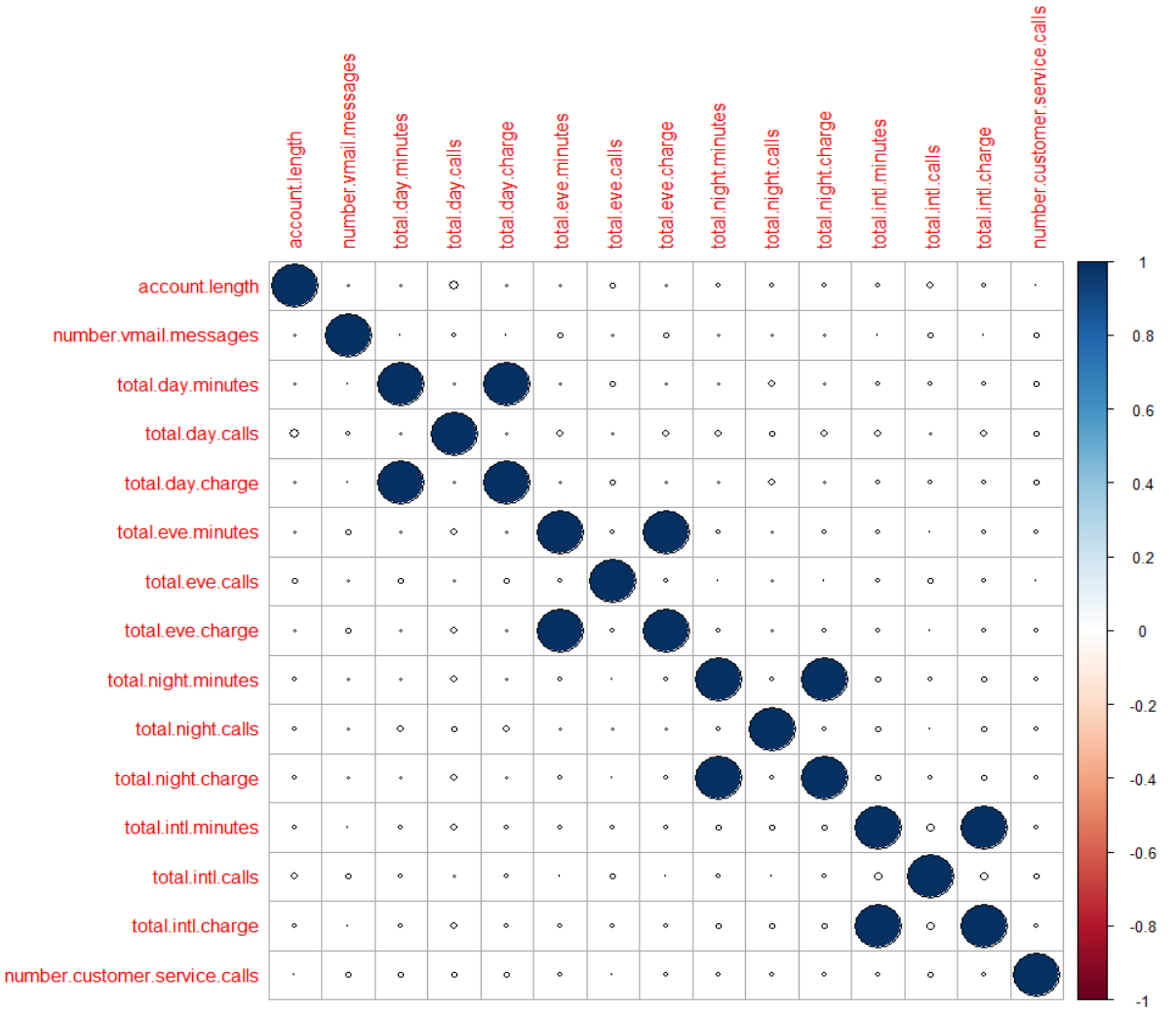
The only thing we check here is the accuracy of the model and the considered error metric to train and make the predictions for the test data. We can easily see that the *decision tree* model is nearly 7% more accurate and less prone to *false negative rate* than *Logistic Regression or Naïve Bayes* but the difference between *Random Forest* and Decision Tree is very less i.e. nearly 1%.

So, we'll go with the model which is giving us the highest accuracy and lowest error which here is *decision trees*.

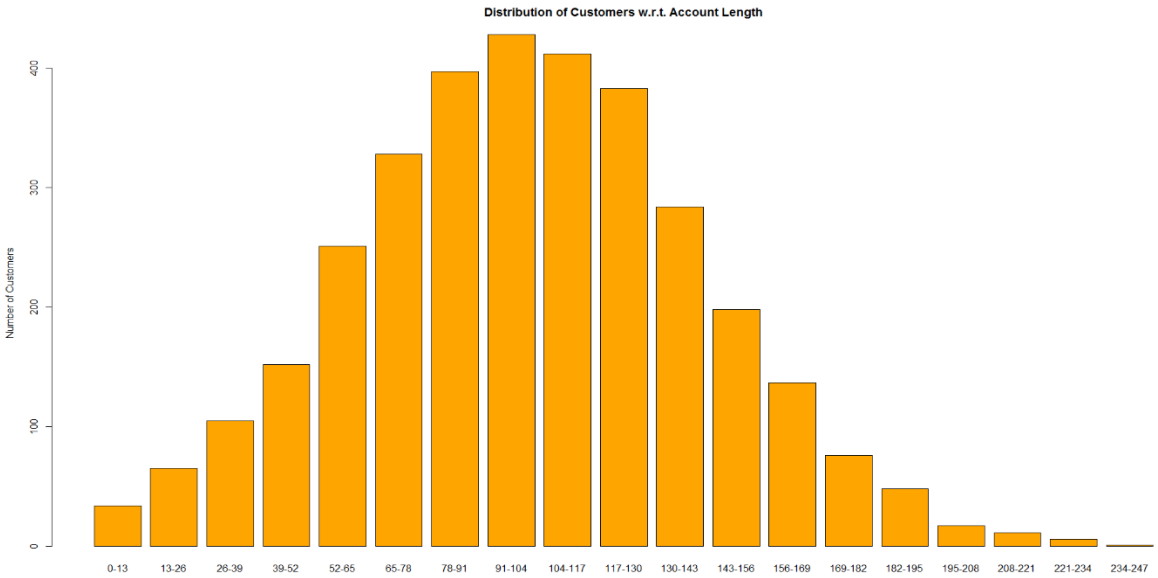
Appendix A – Extra Figures



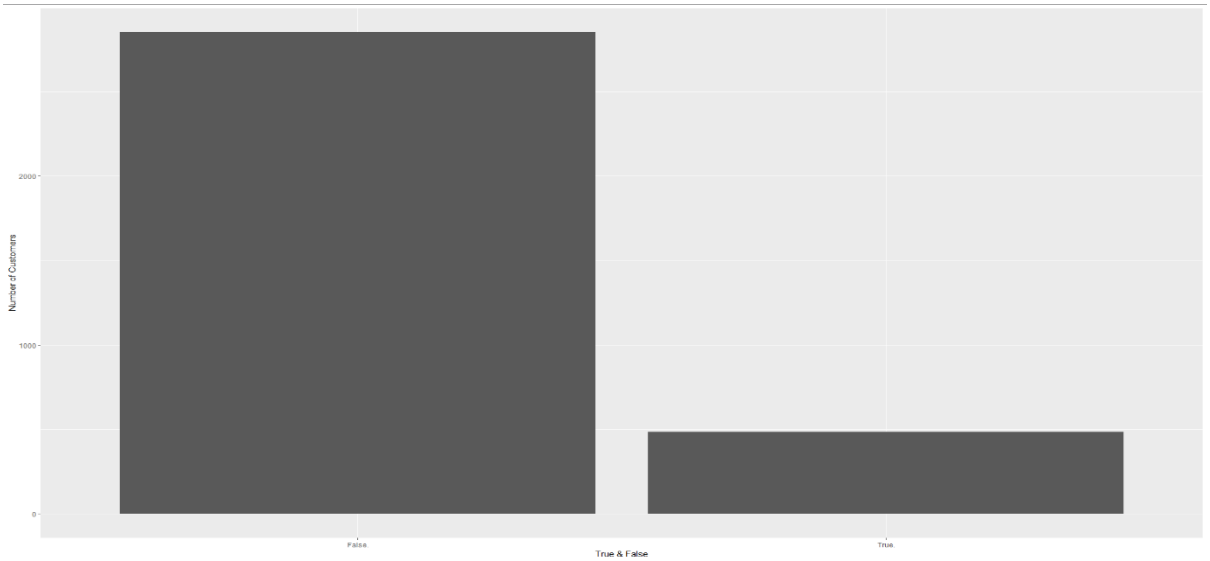
Probability Density Functions of all the Numerical Predictor Variables



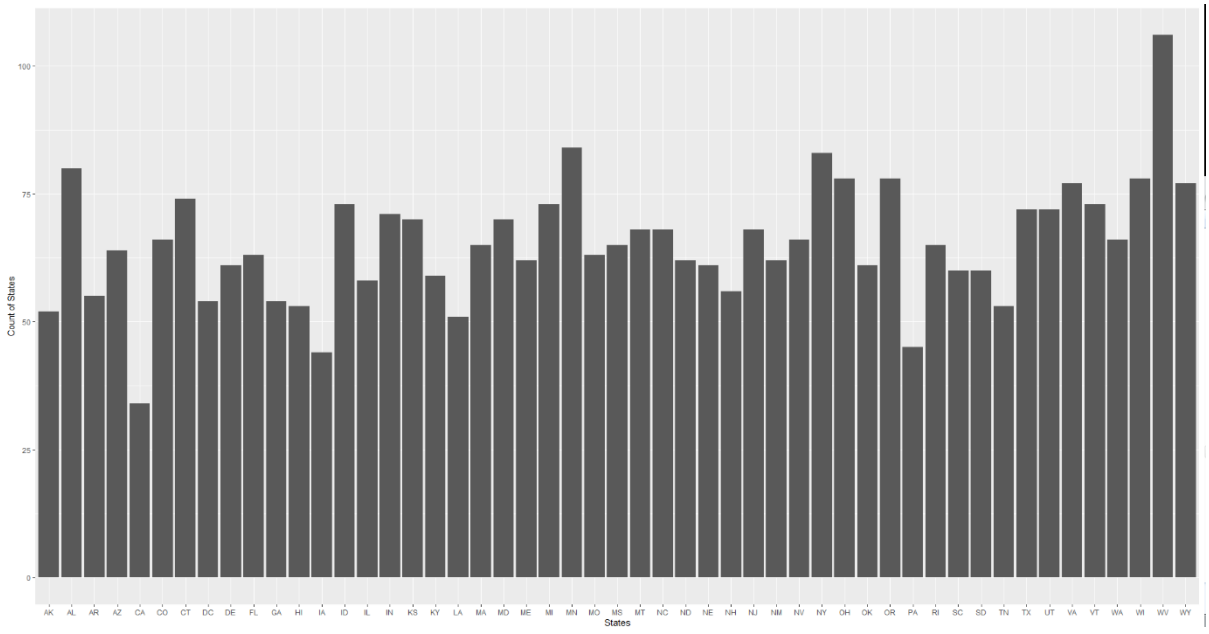
Correlation Analysis Plot between Numerical Variables



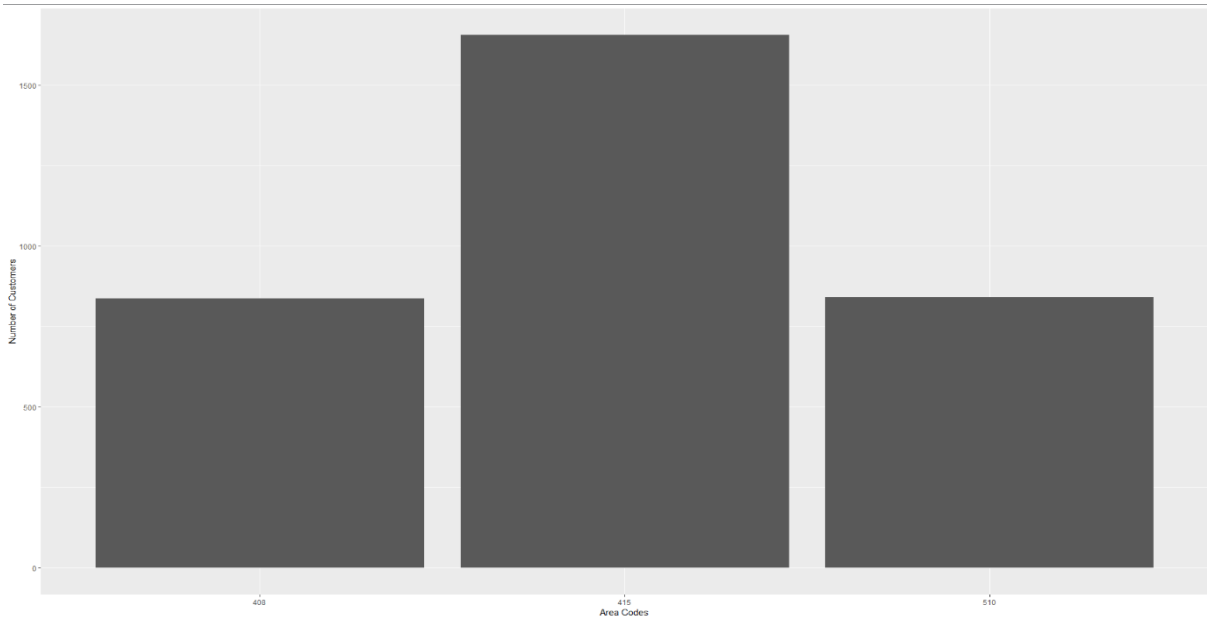
Variation of Customers w.r.t. Account Length



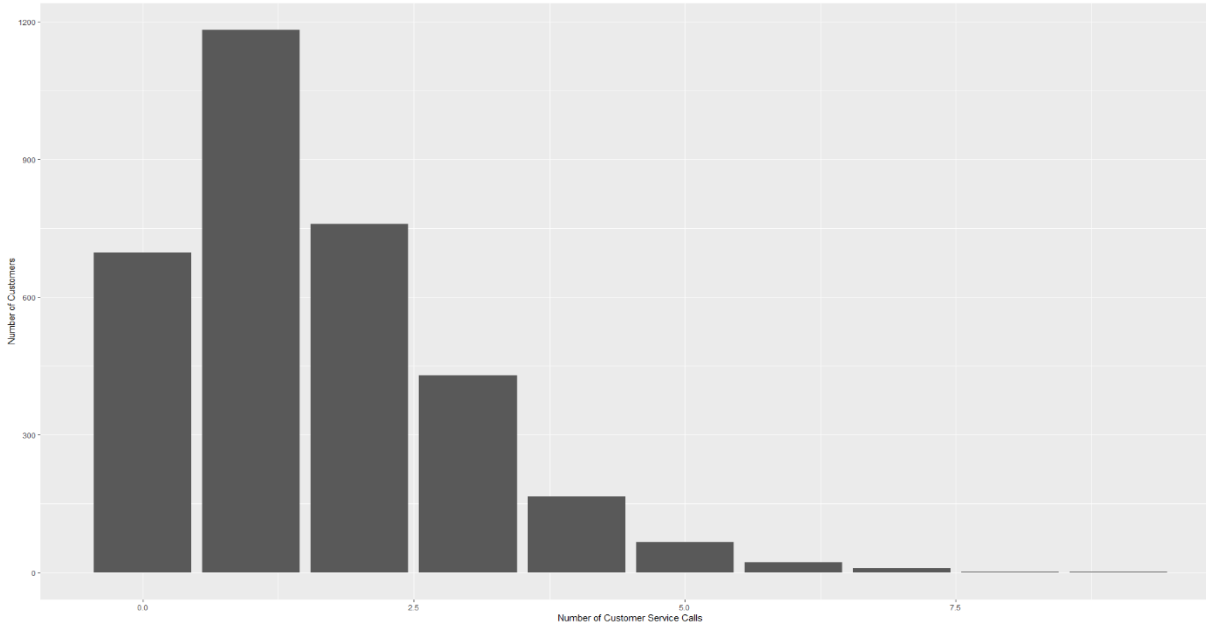
Variation of Customers w.r.t. Churn



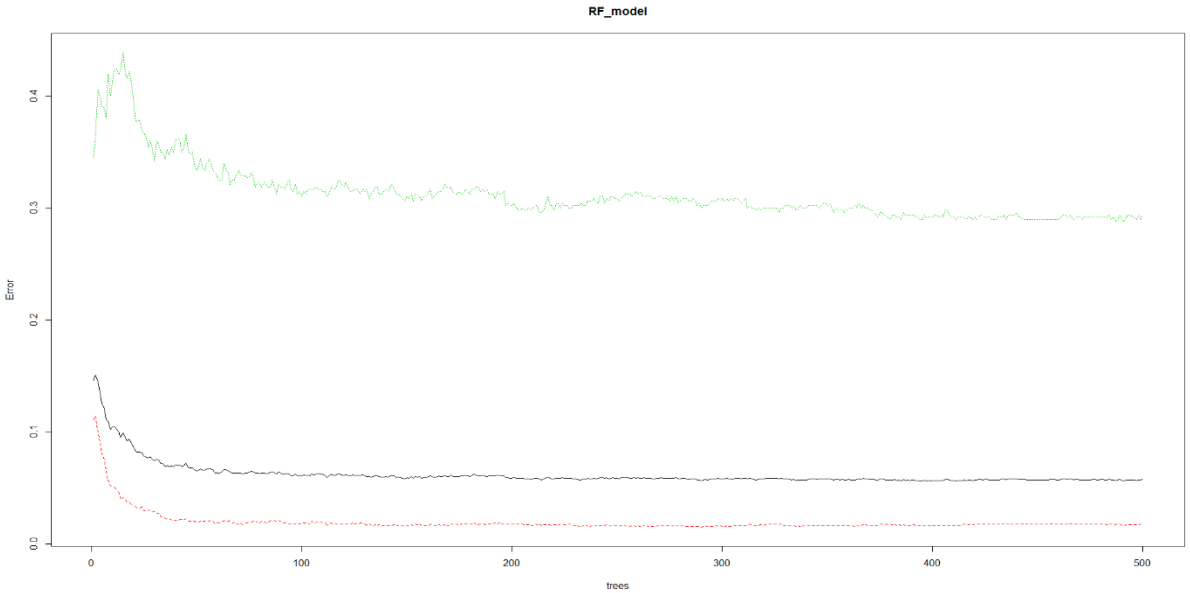
Variation of Customer w.r.t. States



Variation of Customers w.r.t. Area Codes



Variation of Customers w.r.t. Number of Customer Service Calls



Random Forest Classifier Plot

Appendix B – R Code

```
# Clear the environment
rm(list = ls())

#importing the required libraries
library("ggplot2")
library("corrgram")
library("randomForest")
library("C50")

# Load the file
customer_churn = read.csv("Train_data.csv")

# Explore the data
str(customer_churn)

# Check for missing values
sapply(customer_churn, function(x) sum(is.na(x)))

##### PRE PROCESSING ####

# Drop the unimportant variables which does not contribute to the model at all
# As, phone number does not contribute any significant information to the
# prediction model at all
customer_churn$phone.number = NULL

# Convert the area code from integer class to factor
# As, there are only 3 areas so it is better to convert the variable's data
# in the form of a factor as it improves the efficiency of the code
customer_churn$area.code = factor(customer_churn$area.code)

# We can also, plot the probability density functions of the numerical variables
# to check their distribution and also skewness
old_par = par("mar")
par(mar = c(1,1,1,1))
library(psych)
multi.hist(customer_churn[,sapply(customer_churn, is.numeric)], main = NA, dcol = c("blue", "red"),
           dlty = c("solid", "solid"), bcol = "linen")
par(mar = old_par)


##### EDA & Feature Selection #####

# Creating the correlation analysis plot to analyse the dependecies of numeric
# variables
library(corrplot)
cor_plot = cor(customer_churn[,sapply(customer_churn, is.numeric)])
corrplot(cor_plot, method = "circle", addgrid.col = "darkgray",
         outline = TRUE, number.digits = 2)
#The correlation plot tells us that the numerical variables where there is
#"minutes" and "charges" are correlated. So, we can drop one of the two from every pair of variables


# Also,we can perform binning on the account length variable in order to check
# its distribution. As it is very large amount of data.
summary(customer_churn$account.length)
# BUt after checking its summary we can say that we can divide it into intervals
# so that we can check its distribution across the total population
```

```
# set up boundaries for bins
breaks <- c(0,13,26,39,52,65,78,91,104,117,130,143,156,169,182,195,208,221,234,247)
# specify interval/bin labels
labels <- c("0-13", "13-26", "26-39", "39-52", "52-65", "65-78", "78-91", "91-104",
            "104-117", "117-130", "130-143", "143-156", "156-169", "169-182",
            "182-195", "195-208", "208-221", "221-234", "234-247")
# bucketing data points into bins
bins <- cut(customer_churn$account.length, breaks, include.lowest = T, right=FALSE, labels=labels)
# inspect bins
summary(bins)
# Plotting the account length distribution
plot(bins, main="Distribution of Customers w.r.t. Account Length", xlab="Account Length",
ylab="Number of Customers",col="orange")


# Checking the distribution of churn in the observations
library(plyr)
churn_count = count(customer_churn, 'Churn')
ggplot(churn_count, aes(x=Churn, y = freq)) + xlab("True & False") +
  geom_histogram(stat="identity") + ylab("Number of Customers")

# percentage of the false churn
paste0("The percentage of false churn is ", (churn_count$freq[1] / 3333)*100, " %")

# percentage of the true churn
paste0("The percentage of true churn is ", (churn_count$freq[2] / 3333)*100, " %")
# This churn rate implies that nearly 15% of the customers left the telecom
# company's services. Now the reason for the leave acan be anything from random
# chance to any specific problem in the company's policies and charges.

# In order to be more sure about the reason for churn we can check the dependency
# of churn with different variables.


# Checking the distribution of states in the total observations
state_count = count(customer_churn, 'state')
ggplot(state_count, aes(x=state, y = freq)) + xlab("States") +
  geom_histogram(stat="identity") + ylab("Count of States")
# The distribution gives us an idea that there is majority of only "WEST VIRGINIA"
# in the total observations


# International Plan count
Intplan_count = count(customer_churn, 'international.plan')

# percentage of the non-international plan subscribers
paste0("The percentage of customers not subscribed in international plan is ", (Intplan_count$freq[1] /
3333)*100, " %")

# percentage of the international plan subscribers
paste0("The percentage of customers subscribed in international plan is ", (Intplan_count$freq[2] /
3333)*100, " %")

# Voice-Mail Plan count
VoiceMailplan_count = count(customer_churn, 'voice.mail.plan')

# percentage of the non-voice-mail plan subscribers
paste0("The percentage of customers not subscribed in voice-mail plan is ", (VoiceMailplan_count$freq[1]
/ 3333)*100, " %")

# percentage of the non-voice-mail plan subscribers
```

```
paste0("The percentage of customers subscribed in voice-mail plan is ", (VoiceMailplan_count$freq[2] /
3333)*100, " %")

# Checking the distribution of different areas in the observations
# Area Code count
AreaCode_count = count(customer_churn, 'area.code')
ggplot(AreaCode_count, aes(x=area.code, y = freq)) + xlab("Area Codes") +
  geom_histogram(stat="identity") + ylab("Number of Customers")

# Checking the distribution of number customer service calls in the observations
ServiceCall_count = count(customer_churn, 'number.customer.service.calls')
ggplot(ServiceCall_count, aes(x=number.customer.service.calls, y = freq)) + xlab("Number of Customer
Service Calls") +
  geom_histogram(stat="identity") + ylab("Number of Customers")
# The number of customer service calls is dominated by 2 followed by 0 & 2 and
# then followed by 3 & 4.

# Now, check the churn with different variables

# Now we check the churn with international plan subscribers
intlchurn_count = count(customer_churn, c('international.plan', 'Churn'))
# percentage of international plan subscribers and their false churn
paste0("International Plan and False Churn ", (intlchurn_count[3,3]/sum(intlchurn_count[c(3,4),3]))*100, "
%")

# percentage of international plan subscribers and their true churn
paste0("International Plan and False Churn ", (intlchurn_count[4,3]/sum(intlchurn_count[c(3,4),3]))*100, "
%")

# percentage of non-international plan subscribers and their false churn
paste0("Non-International Plan and False Churn ",
(intlchurn_count[1,3]/sum(intlchurn_count[c(1,2),3]))*100, " %")

# percentage of non-international plan subscribers and their true churn
paste0("Non-International Plan and True Churn ",
(intlchurn_count[2,3]/sum(intlchurn_count[c(1,2),3]))*100, " %")
# According to the findings, the churn is very high when the customer is a international
# plan subscriber i.e. above 40%.
# So, the International Plan is a very important variable in determining the churn

#Chi-Squared test of independence
factor_index = sapply(customer_churn, is.factor)
#In the previous we selected only the categorical variables

factor_data = customer_churn[,factor_index]

for (i in 1:5){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$Churn, factor_data[,i])))
}
# According, to the chi-square test the "area.code" variable is not significant and can be
# rejected as the p-value is very high.

#Dimension reduction
customer_churn_del = subset(customer_churn,
                            select = -c(area.code, total.day.charge, total.eve.charge, total.night.charge,
                                        total.intl.charge))


# Clear the environment except customer_churn_del
library(DataCombine)
```

```
rmExcept(c("customer_churn_del","customer_churn"))

##### MODEL DEVELOPMENT ####
customer_churn_model = customer_churn_del

# Divide the data into train and test using stratified sampling method
# createdatapartition is one more function used to create stratified samples
# Here in createdatapartiotion function 1st arg is of the reference variable on the basis
# of which splitting will take place, 2nd arg is the percentage of observation we need
# in our sample, list = false implies that theer will no repetitive observations
set.seed(1234)
train = customer_churn_model
test = read.csv("Test_data.csv")
test = subset(test, select = -c(phone.number, area.code, total.day.charge, total.eve.charge,
total.night.charge,
                                total.intl.charge))

##### LOGISTIC REGRESSION #####
# Logistic Regression
# glm is built-in function which helps us to build logistic regression on top of the dataset
LR_model = glm(Churn ~ ., data = train, family = "binomial"(link="logit"))

#summary of the model
summary(LR_model)
# The logistic regression test's result suggests us that the most important variables are
# "international.plan - yes", "total.day.minutes", "total.eve.minutes", "total.intl.minutes"
# "number.customer.service.calls" followed by "total.night.minutes" and "voice.mail/plan - yes"

#anova
anova(LR_model, test = "Chisq")
# Here, we can analyse the drop in the deviance when adding each variable. "international.plan"
# and "number.customer.service.calls" followed by "total.day.minutes" significantly reduces the
# residual deviance. But, the variables "voice.mail.plan", "total.eve.minutes","total.intl.minutes"
# seems to improve the model very less even though they all have low p-values

test$Churn = as.character(test$Churn)
test$Churn[test$Churn == " True."] = "1"
test$Churn[test$Churn == " False."] = "0"

#predict using logistic regression
#Type = response gives us the probabilities
LR_predictions = predict(LR_model, newdata = test, type = "response")

#Convert probabilities
#Converting the probabilities into classes 1 & 0
LR_predictions = ifelse(LR_predictions > 0.5, 1, 0)

misClassifierError = mean(LR_predictions != test$Churn)

print(paste('Logistic Regression Accuracy', 1 - misClassifierError))

#Evaluate the performance of the classification model
ConfMatrix_LR = table(test$Churn, LR_predictions > 0.5)
#Accuracy = 87.16%
#FNR = 75%
output = cbind(test, LR_predictions)
write.csv(output,"LR_Output.csv", row.names = F)


##### NAIVE BAYES #####
library(e1071)
```

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

#Predict the test cases
#type = class will internally divide the whole extracted probabilities into yes or no
#if we want the probabilities we can write type = raw
NB_predictions = predict(NB_model, test[,1:14], type = 'class')

#Look at the confusion matrix
ConfMatrix_NB = table(observed = test[,15], predicted = NB_predictions)
#Accuracy = 88.12%
#FNR = 71.42%
output = cbind(test, NB_predictions)
write.csv(output,"NB_Output.csv", row.names = F)

############ DECISION TREES ###########

#Decision Tree for classification
#Develop Model on training data
#1st arg is of the target var, here "Churn ~." full stop implies that except responded
#all the other variables are to be considered independent, we can also write the var
#names followed by + explicitly. 2nd arg is of the dataset. 3rd arg trails = 100
#implies that the compiler needs to build 100 trees one by one and select the best
#tree out of all trees. 4th arg rules = TRUE implies that the compiler needs
#to exctract the business rules out of the decision tree.
c50_model = C5.0(Churn ~., train, trials = 100, rules = TRUE)

#write rules into disk
write(capture.output(summary(c50_model)), "c50Rules.txt")

#Lets predict the test cases
#Predict is a function which helps in predicting the new tests cases with the help of
#existing model. 1st arg is of the mdoel name. 2nd arg is the reference to the data but here we
#want to remove the target variable so for that [,-17]. 3rd arg type = class gives the
#class to which the target variable comes out to be, means here or target variable
#is binary so for that we need to predict the test cases also in the form of binary numbers
c50_predictions = predict(c50_model, test[,-15], type = "class")

#Now all we have to do is compare the predicted value with the actual values of the target
#variable, then we have to look at how the variable can able to perform on the test data
#it mean that how accurate the model is.

#Now as the the target variable is categorical we can't do regression here as for regression
# we need a target variable which is continuos.

#Evaluate the performance of the classification model
#Here we are building a contingency table(confusion matrix). 1st arg is of the
#actual values. 2nd arg is of the predcited values
confMatrix_c50 = table(test$Churn, c50_predictions)

#Here now we decide on the basis of the business problem that what is the best error metric
#Accuracy = 95.80%
#FNR = 29.01%
output = cbind(test, c50_predictions)
write.csv(output,"c50_Output.csv", row.names = F)

########### RANDOM FOREST CLASSIFIER ############

#Here we want to reduce the error rate
#RANDOM FOREST
```

```
#1st arg is of the target variable and saying that all the remaining variables are indep
#2nd arg is of dataset. 3rd arg importance = true suggests that apart from giving me
#output predictions laso give me the important variables. 4th arg is no. of trees.
#In this 1st take 100 trees, then take 500 and see if it improves the model.
RF_model = randomForest(Churn ~., train, importance = TRUE, ntree = 500)
print(RF_model)
plot(RF_model)

#Now, predict the test data using the random forest model
#1st arg is fo the RF model. 2nd arg is of the reference to the independent vars
RF_Predictions = predict(RF_model, test[,-15])

#Evaluate the performance of the classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
#Accuracy = 94.60%
#FNR = 30.35%
output = cbind(test, RF_Predictions)
write.csv(output,"RF_Output.csv", row.names = F)
```