Predicting Bike Rental Count

Akshay Dinkar Patil

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. The objective of this case is to predict customer behaviour.

1.2 Data

Task is to predict the churn score i.e. if the customer has moved (1=yes; 0 = no)
The Predictors provided are as follows:

- account length
- international plan
- voicemail plan
- number of voicemail messages
- total day minutes used
- day calls made
- total day charge
- total evening minutes
- total evening calls
- total evening charge
- total night minutes
- total night calls
- total night charge
- total international minutes used
- total international calls made
- total international charge
- number of customer service calls made

The Target variable is 'move' if the customer has moved (1=yes; 0 = no)

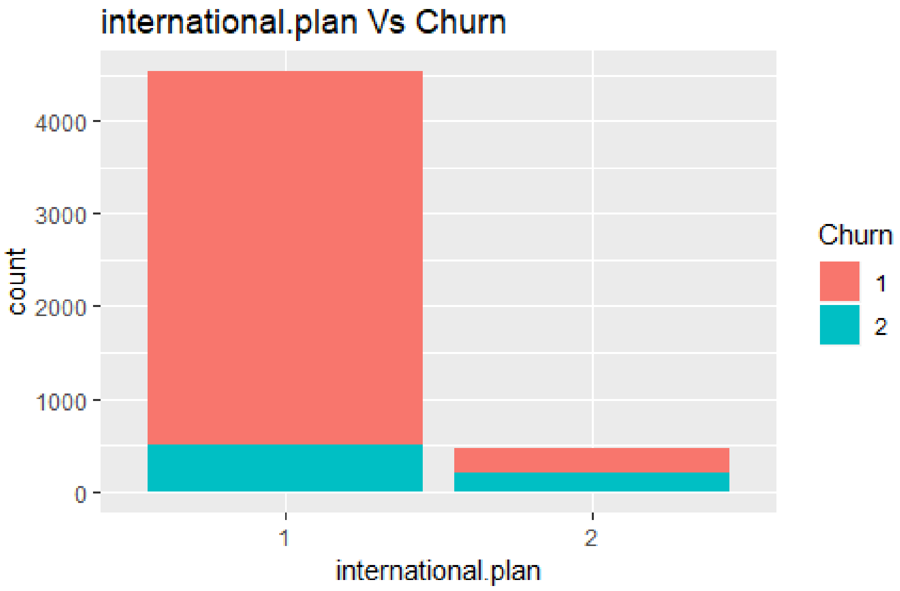
Chapter 2

Methodology

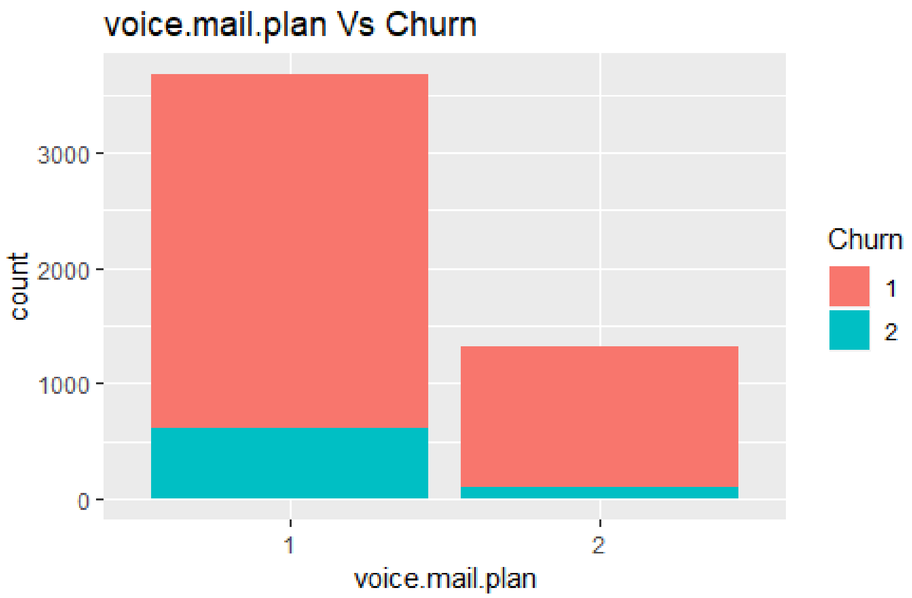
2.1 Pre Processing

Data pre processing is the first stage of any type of project. In this stage we get the feel of the data. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, Imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.
Further we will look into what pre processing steps do this project was involved in.

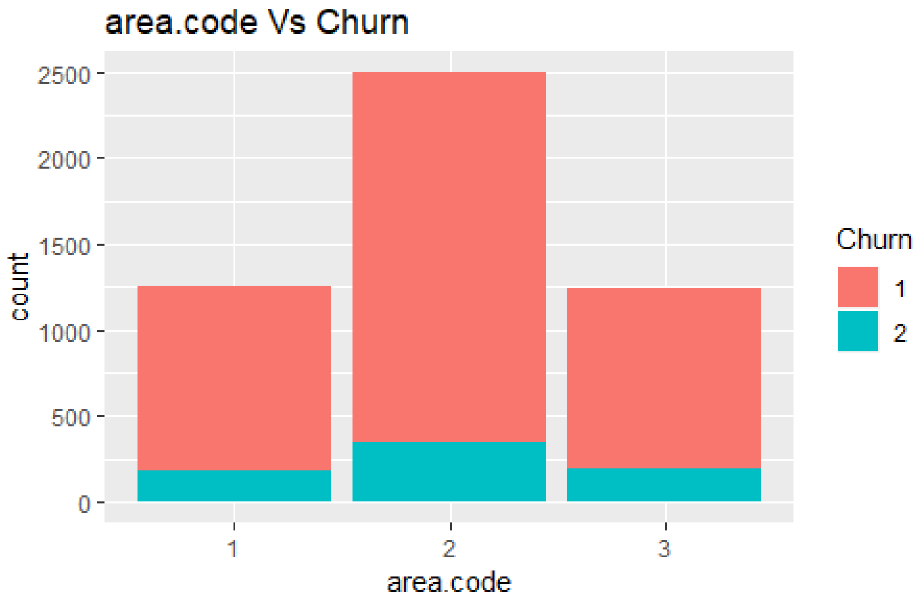
Getting feel of data via visualization:
# churning of customer w.r.t international.plan



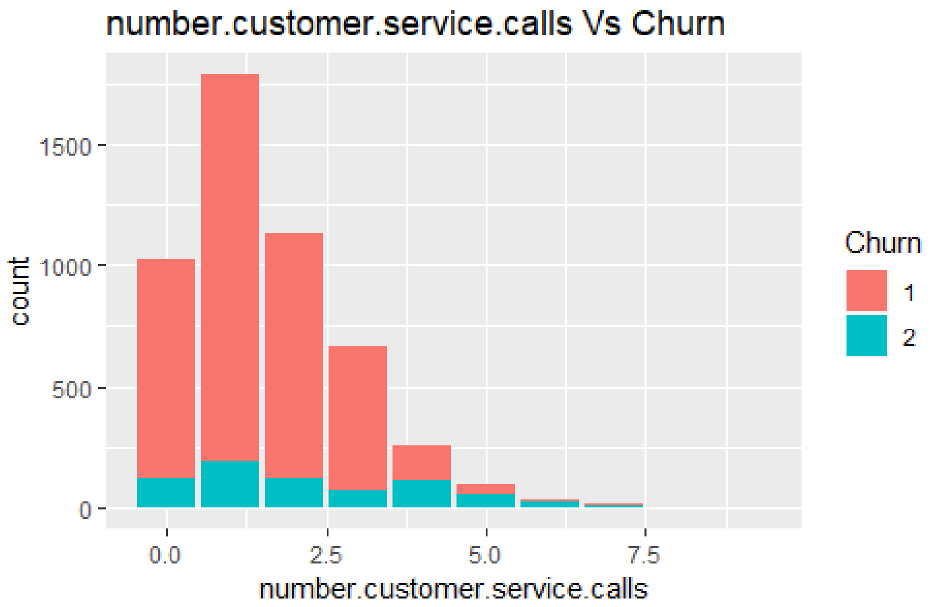
# churning of customer w.r.t voice.mail.plan



# churning of customer w.r.t area.code

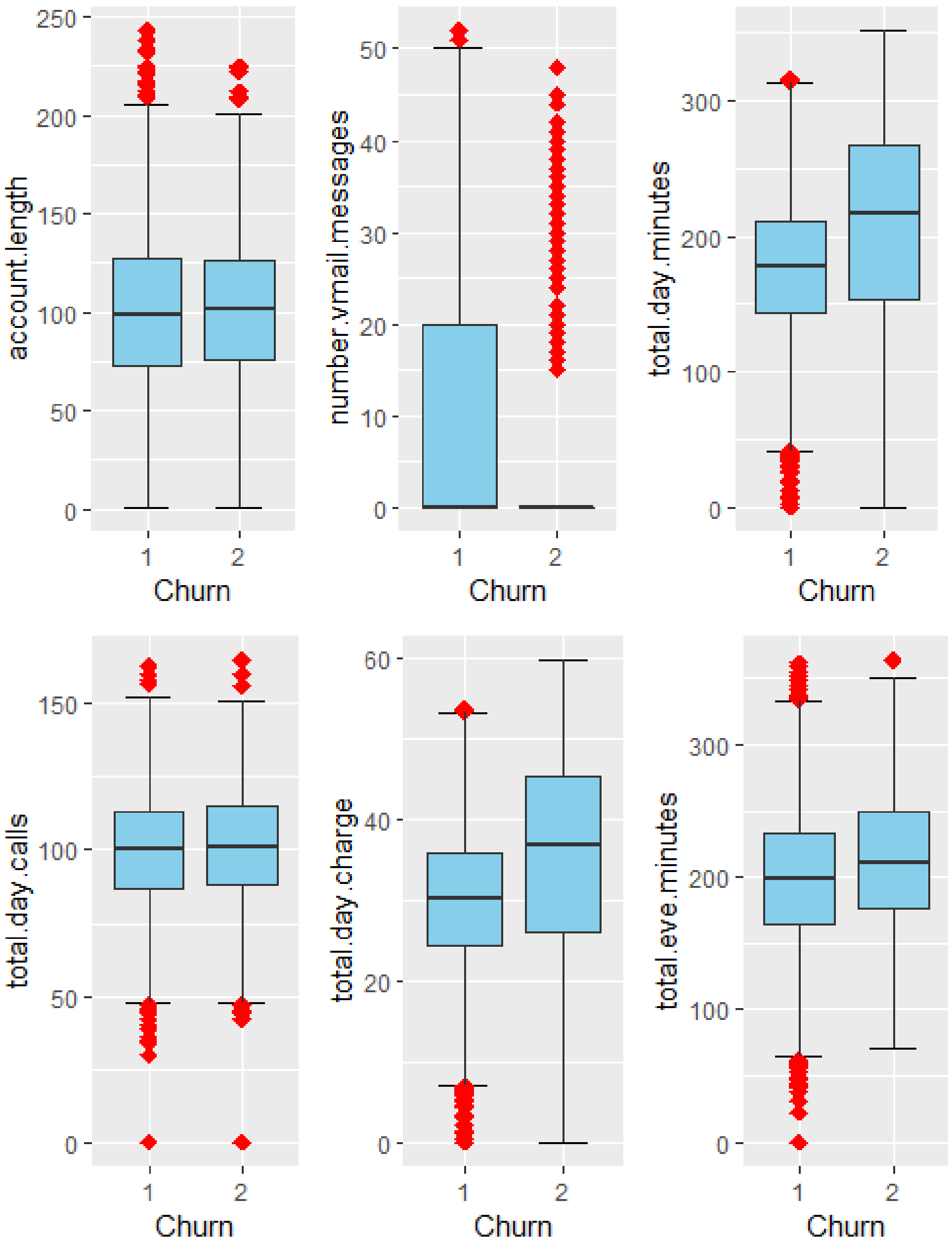


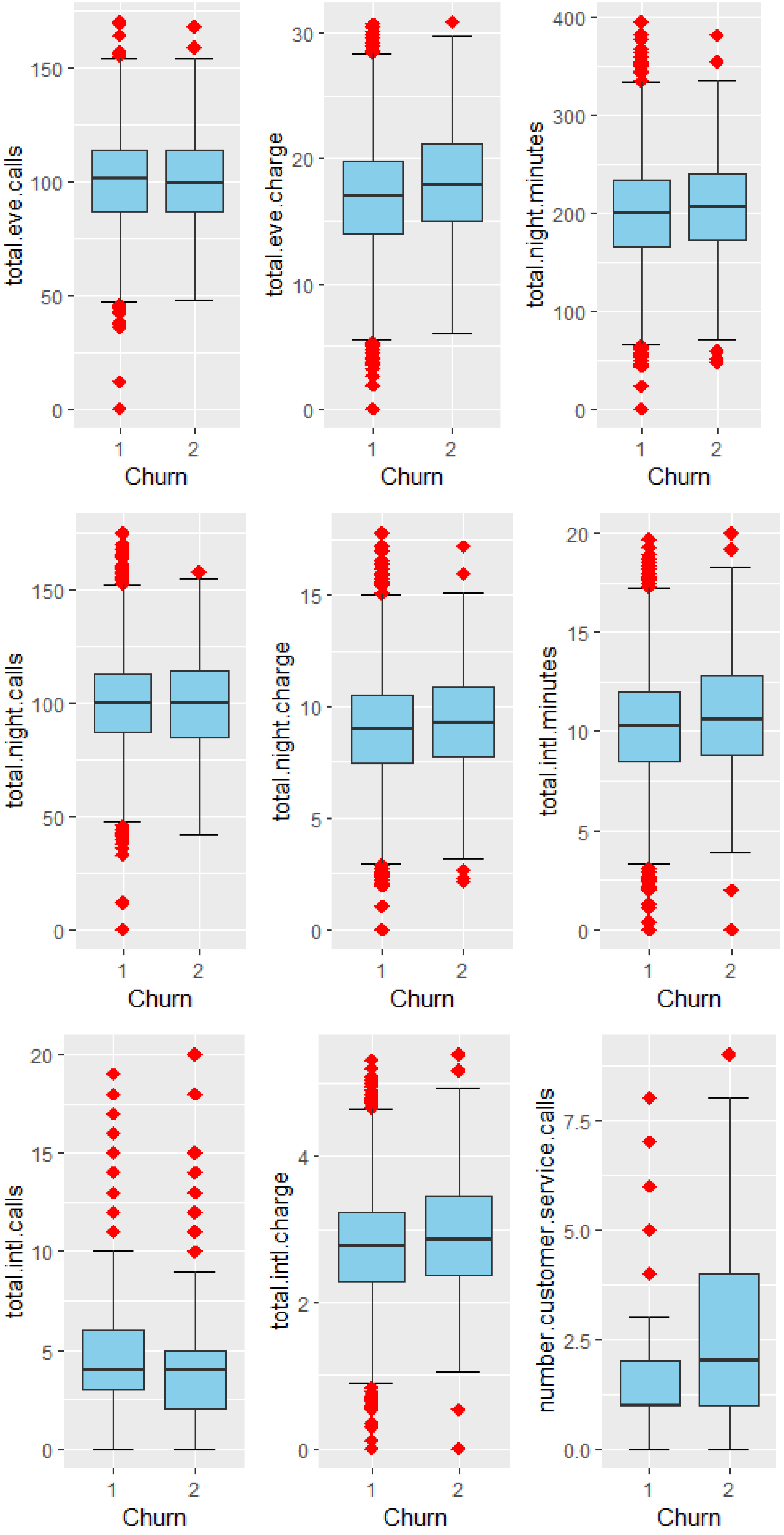
# churning of customer w.r.t number.customer.service.calls



2.1.1 Outlier Analysis

We look for outlier in the dataset by plotting Boxplots. There are outliers present in the data. But if we understand and think about the business model here, we get that the dataset is related to customer usage pattern. A customer can do anything he wants with his calling plan. He can talk any time he wants. For example, ‘total day minutes used’ is the variable in our dataset, it can range from any value to any value. So, I have come to the conclusion that to leave the dataset with outliers to get the most out of the dataset and predict our target variable ‘Churn’.




total.eve.calls
0
50
100
150
total.eve.charge
0
10
20
30
total.night.minutes
0
100
200
300
400
total.night.calls
0
50
100
150
total.night.charge
0
5
10
15
total.intl.minutes
0
5
10
15
20
total.intl.calls
0
5
10
15
20
total.intl.charge
0
2
4
number.customer.service.calls
0.0
2.5
5.0
7.5
1
2
Churn

2.1.2 Missing value Analysis

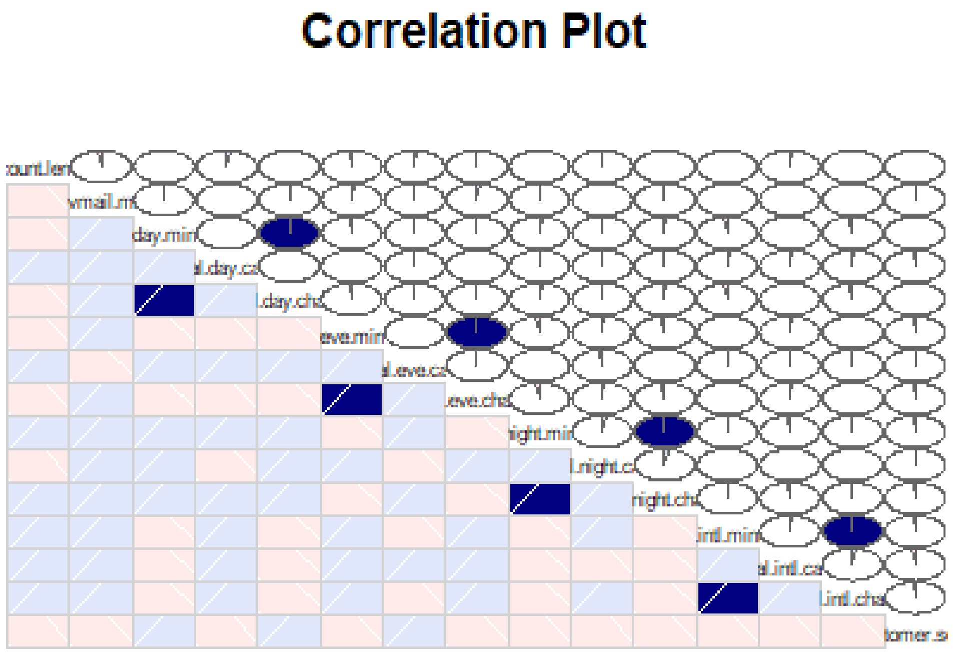
In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks.
Some missing values are in form of NA. missing values left behind after outlier analysis, missing values can be in any form. Unfortunately, in this dataset we haven't found any missing values. Therefore, we will continue to next step.

2.1.3 Feature Selection

In this step we would allow only to pass relevant features to further steps. We remove irrelevant features from the dataset. We do this by understanding the domain knowledge of our features like we look for features which will not be helpful in predict the target variables. In this dataset we have to predict the churn based on usage pattern of the customers, features which excludes this list are – state, area code, account length and phone numbers. Further there are 2 types of test involved for feature selection:

1. **Correlation analysis** – this requires only numerical variables. Therefore, we will filter out only numerical variables and feed it to correlation analysis. We do this by plotting correlation plot for all numerical variables. There should be no correlation between independent variables but there should be high correlation between independent variable and dependent variable. So, we plot the correlation plot. we can see that in correlation plot dark blue indicates that 2 variables are highly correlated with each other here, total.day.minutes is highly correlated with total.day.charge total.eve.minutes is highly correlated with total.eve.charge total.night.minutes is highly correlated with total.night.charge total.intl.minutes is highly correlated with total.intl.charge now instead of selecting all variables for our modelling, we would select one of each of them lets select all charges features-total.day.charge, total.eve.charge, total.night.charge, total.intl.charge and drop all minutes features-total.day.minutes, total.eve.minutes, total.night.minutes, total.intl.minutes.

Correlation Plot:



2 **Chi-Square test of independence** – Unlike correlation analysis we will filter out only categorical variables and pass it to Chi-Square test. Chi-square test compares 2 categorical variables in a contingency table to see if they are related or not.

I. Assumption for chi-square test: Dependency between Independent variable and dependent variable should be high and there should be no dependency among independent variables.

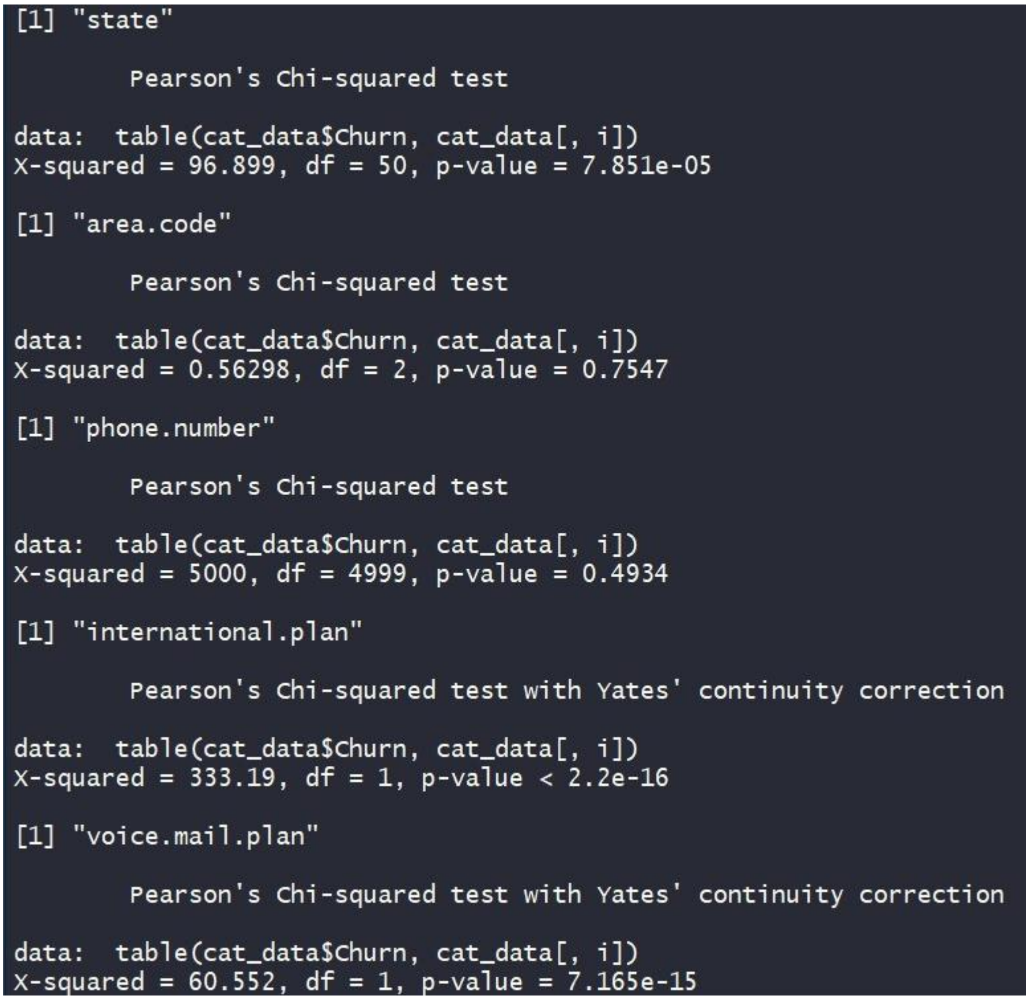
II. Before proceeding to calculate chi-square statistic, we do the hypothesis testing:

Null hypothesis: 2 variables are independent.
Alternate hypothesis: 2 variables are not independent.
The interpretation of chi-square test:

I. For theorical or excel sheet purpose: If chi-square statistics is greater than critical value then reject the null hypothesis saying that 2 variables are dependent and if it’s less, then accept the null hypothesis saying that 2 variables are independent.

II. While programming: If p-value is less than 0.05 then we reject the null hypothesis saying that 2 variables are dependent and if p-value is greater than 0.05 then we accept the null hypothesis saying that 2 variables are independent.

- If p-value<0.05 then select the variable,
- If p-value>0.05 then ignore the variable as this independent variable doesn’t carry any information to explain the dependent variable therefore it will not add any information while modelling. After analysing p value of all categorical features, we come to a conclusion that, feature- phone.number, area.code and account.length will be removed.



```
[1] "state"

        Pearson's Chi-squared test

data:  table(cat_data$Churn, cat_data[, i])
X-squared = 96.899, df = 50, p-value = 7.851e-05

[1] "area.code"

        Pearson's Chi-squared test

data:  table(cat_data$Churn, cat_data[, i])
X-squared = 0.56298, df = 2, p-value = 0.7547

[1] "phone.number"

        Pearson's Chi-squared test

data:  table(cat_data$Churn, cat_data[, i])
X-squared = 5000, df = 4999, p-value = 0.4934

[1] "international.plan"

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

data:  table(cat_data$Churn, cat_data[, i])
X-squared = 333.19, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

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

data:  table(cat_data$Churn, cat_data[, i])
X-squared = 60.552, df = 1, p-value = 7.165e-15
```

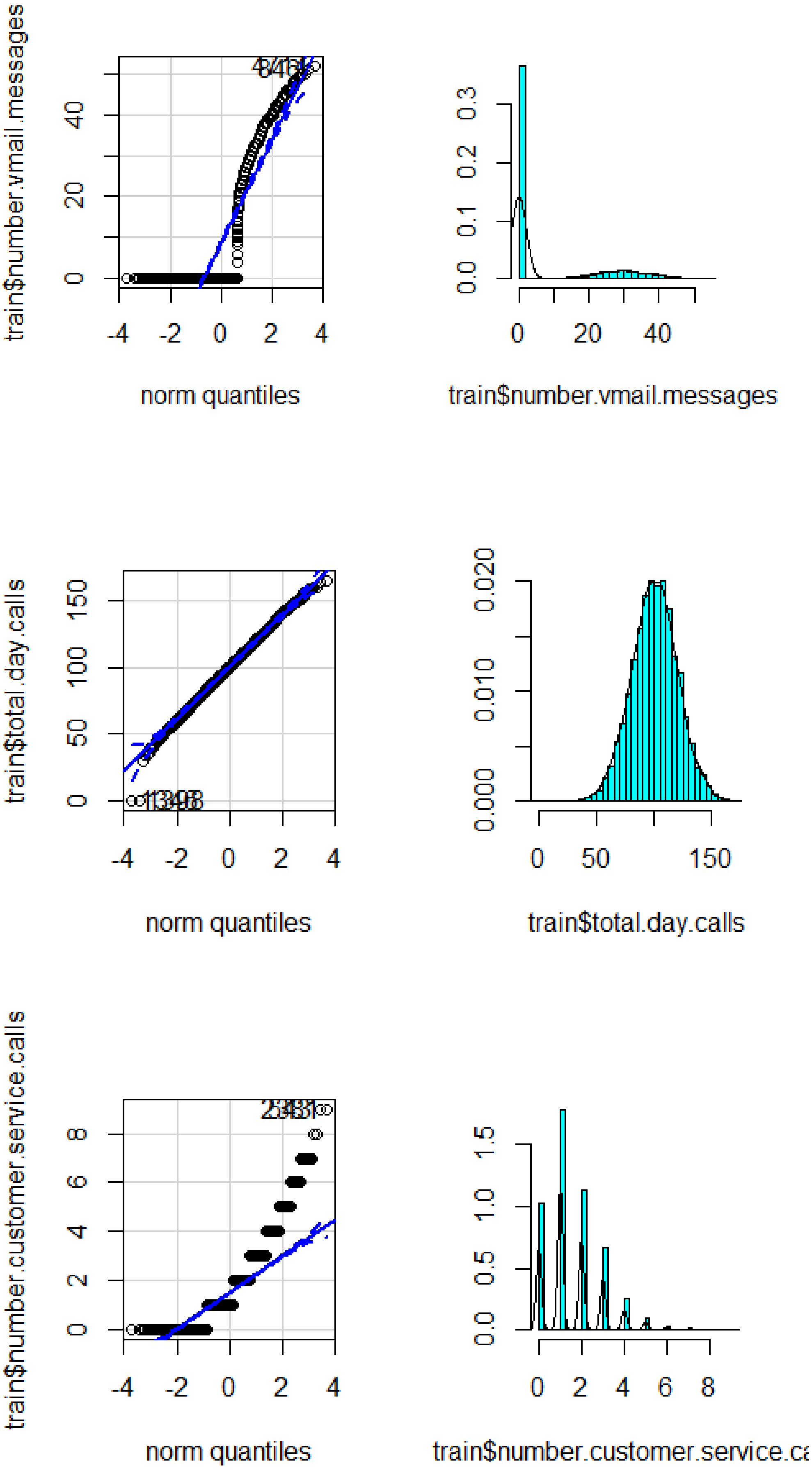
2.1.4 Feature Scaling

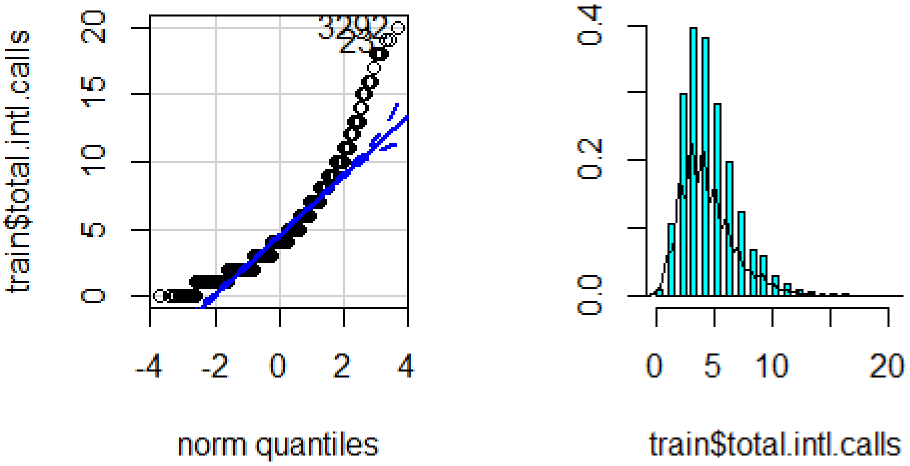
Data Scaling methods are used when we want our variables in data to scaled on common ground. It is performed only on continuous variables.

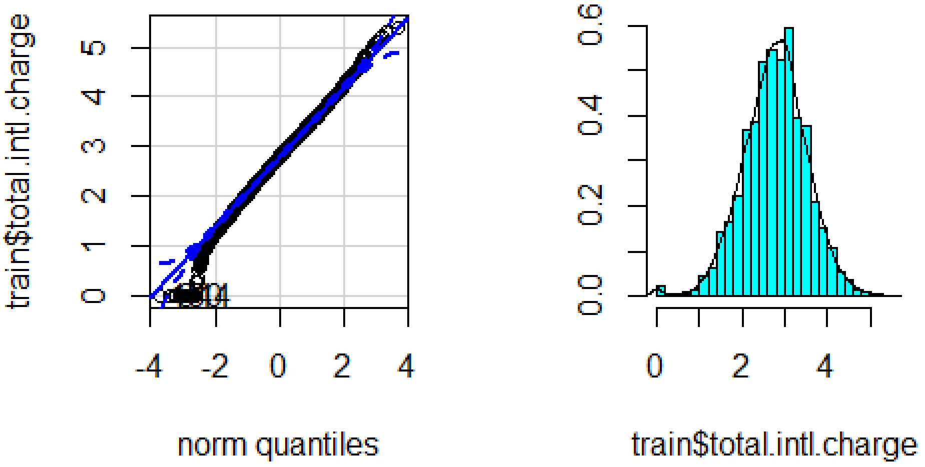
- **Normalization**: Normalization refer to the dividing of a vector by its length. normalization normalizes the data in the range of 0 to 1. It is generally used when we are planning to use distance method for our model development purpose such as KNN. Normalizing the data improves convergence of such algorithms. Normalisation of data scales the data to a very small interval, where outliers can be loosed.
- **Standardization**: Standardization refers to the subtraction of mean from individual point and then dividing by its SD. Z is negative when the raw score is below the mean and Z is positive when above mean. When the data is distributed normally you should go for standardization.

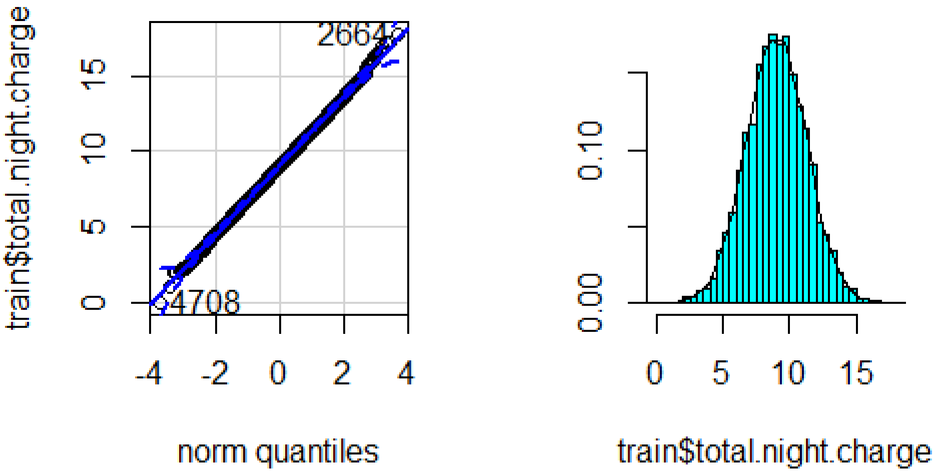
After plotting all the graphs, we can say that most of the variables are normally distributed. Therefore, we will go for standardization over normalization as our most of the data is distributed normally.

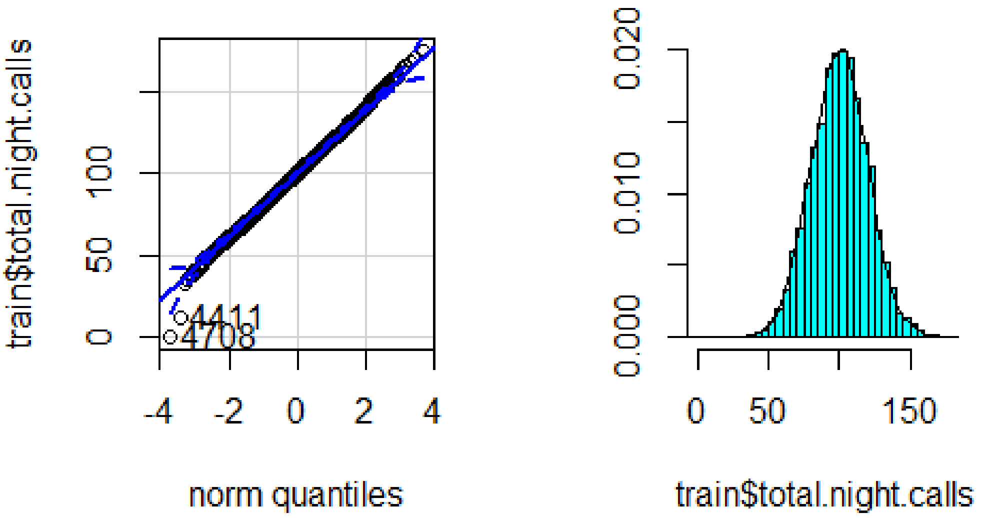
Graphs based on which standardization was chosen:
Note: All are continuous variables.

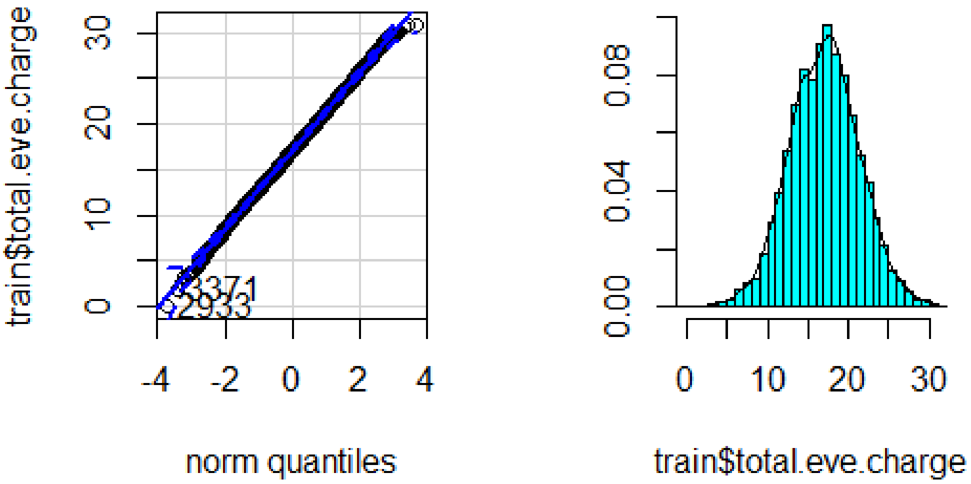

train$number.vmail.messages
0
20
40
-4
-2
0
2
4
norm quantiles
0.0
0.1
0.2
0.3
0
20
40
train$number.vmail.messages
train$total.day.calls
0
50
100
150
-4
-2
0
2
4
norm quantiles
0.000
0.010
0.020
0
50
150
train$total.day.calls
train$number.customer.service.calls
0
2
4
6
8
-4
-2
0
2
4
norm quantiles
0.0
0.5
1.0
1.5
0
2
4
6
8
train$number.customer.service.ca

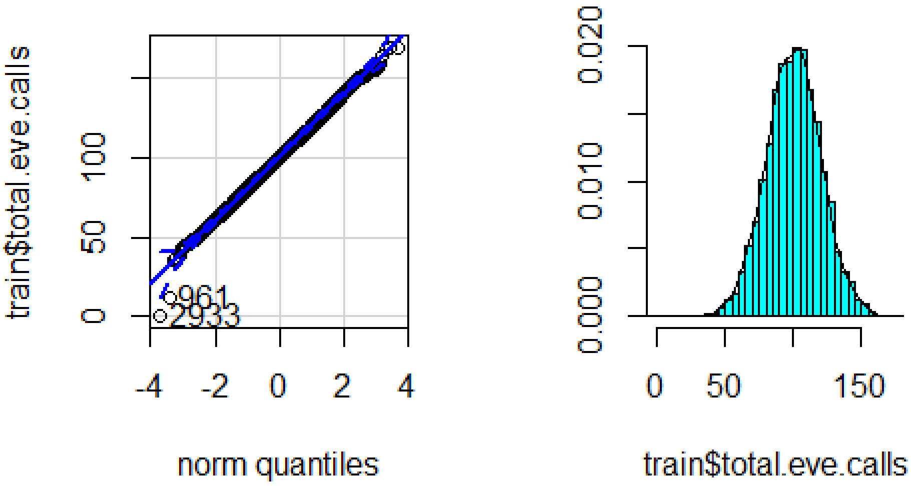

train$total.intl.calls
0
5
10
15
20
3292
23
-4
-2
0
2
4
norm quantiles
0.0
0.2
0.4
0
5
10
20
train$total.intl.calls

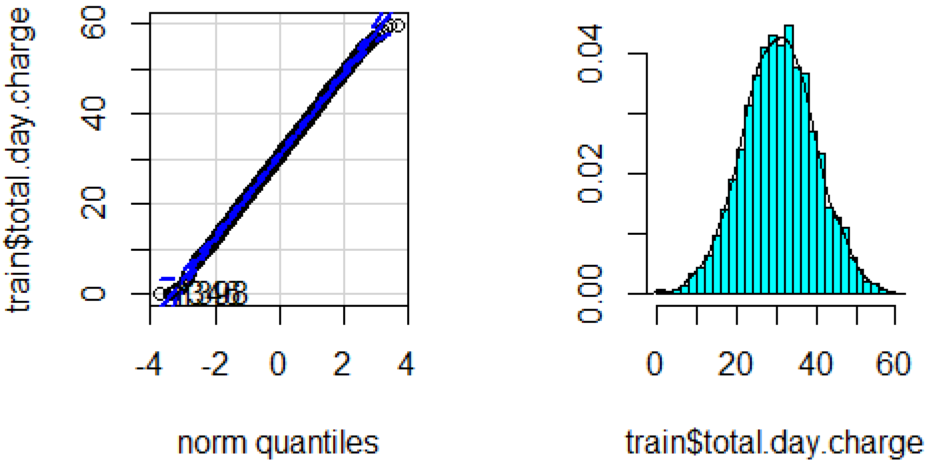

train$total.intl.charge
0
1
2
3
4
5
-4
-2
0
2
4
norm quantiles
0.0
0.2
0.4
0.6
0
2
4
train$total.intl.charge


train$total.night.charge
0
5
10
15
2664
4708
-4
-2
0
2
4
norm quantiles
0.00
0.10
0
5
10
15
train$total.night.charge


train$total.night.calls
0
50
100
4411
4708
-4
-2
0
2
4
norm quantiles
0.000
0.010
0.020
0
50
150
train$total.night.calls


train$total.eve.charge
0
10
20
30
3371
2933
-4
-2
0
2
4
norm quantiles
0.00
0.04
0.08
0
10
20
30
train$total.eve.charge





2.2 Dealing with Target Class imbalance Problem

What is imbalance in data:

It is a dataset in which the observations related to a class variable is not in right proportion. For example, Consider the following table:

Classes in a Variable	Amt. of observation for each class
Class_1	85%
Class_2	10%
Class_3	5%

In this table a variable has 3 classes and observations related to those classes in the dataset are not in proportionate to each other. This type of dataset is imbalanced.

What is target class imbalance problem:

If in a classification problem, our model is trained with unbalanced data, in that scenario our model will predict the class of new test case but it will not be a correct classification because our model is biased towards the class which had higher number of observation/samples in our training data compared to other classes. As imbalance is in predicting the target class, it is called as Target class imbalance problem.

Fig shows what proportion of Churn variable fall into each category class (True or False). We can clearly see that target data is imbalanced. 4293 records fall under False category and 707 records fall under True category. According to the problem statement, our Client is interested in reducing the Churn. We can reduce Churn by predicting them correctly. If we fail to do so, our customer will be loosed to the competition. If we predicted it correctly then our client will provide some extra discounts or features t that specific customer and thus will save that customer from Churning out.

Here False category is our negative class and True is our Positive Class.

In this project, we dealt with this problem by sampling methods and using specific error metrics like - Accuracy, False negative rate, Sensitivity.

Sampling methods used –

We had applied Sampling methods only on training data and it is the best practice to do so because –

i. Your goal is to better train your model by giving it balanced amounts of data.
ii. Your goal is not to predict samples which are created by sampling methods.
iii. Always make sure your test data is free of duplicates or synthetic data, such that you can test your model on real data only.

1. Random Under Sampling: It aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out. Before Random under sampling we had 3320 in negative class and 531 in positive class. After Random under sampling we got 919 in negative class and 531 in positive class.

- **Advantages**
  - It can help improve run time and storage problems by reducing the number of training data samples when the training data set is huge.

- **Disadvantages**
  - It can discard potentially useful information which could be important for building rule classifiers.
  - The sample chosen by random under sampling may be a biased sample. And it will not be an accurate representative of the population. Thereby, resulting in inaccurate results with the actual test data set.

2. Random Over Sampling: Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample. Before Random Over sampling we had 3320 in negative class and 531 in positive class. After Random Over sampling we got 3220 in negative class and 1609 in positive class.

- **Advantages**
    - Unlike under sampling this method leads to no information loss.
    - Outperforms under sampling
- **Disadvantages**
    - It increases the likelihood of overfitting since it replicates the minority class events.

3.Using both Under and Over Random sampling: In this technique we will use both Random Over Sampling and Random Sampling. This is done to overcome disadvantages from both sampling methods. Before Both Random under and Over sampling we had 3320 in negative class and 531 in positive class. After Both Random under and Over sampling we got 2698 in negative class and 1302 in positive class.

4.Synthetic Minority Over Sampling Technique: This technique is followed to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to the original dataset. The new dataset is used as a sample to train the classification models. Before SMOTE we had 3320 in negative class and 531 in positive class. After SMOTE we got 3220 in negative class and 2124 in positive class.

- **Advantages**
    - Mitigates the problem of overfitting caused by random oversampling as synthetic examples are generated rather than replication of instances
    - No loss of useful information
- **Disadvantages**
    - While generating synthetic examples SMOTE does not take into consideration neighbouring examples from other classes. This can result in increase in overlapping of classes and can introduce additional noise
    - SMOTE is not very effective for high dimensional data

Although we can see here Accuracy and FNR is good, we know that our model is trained on unbalanced dataset, and here accuracy is misleading because our model is predicting the majority class, as classifier tend to favour majority class. Therefore, we will use AUC and ROC, which is better performance metric.

Chapter 3

Model Development

Our problem statement wants us to predict the Churn categories i.e. True or False. This is a Classification problem. So, we are going to build classification models on training data and predict it on test data. In this project I have build models using 5 Classification Algorithms:

I. Decision Tree
II. Random Forest
III. Logistic Regression
IV. KNN
V. Naïve Bayes

The complete dataset is split into train and test using stratified sampling
We have,
train and test – Using Stratified Sampling
train_over – Using Random Over Sampling
train_under – Using Random Under Sampling
train_both – Using Random Both Over and Under Sampling
train_smote – Using Synthetic Minority Over Sampling Technique
We will evaluate performance on test dataset generated using Stratified Sampling. As we are interested in Correctly predicting True cases i.e. Positive cases. We will deal with specific error metrics like - Accuracy, False negative rate, Sensitivity.

2.3.1 Model Performance

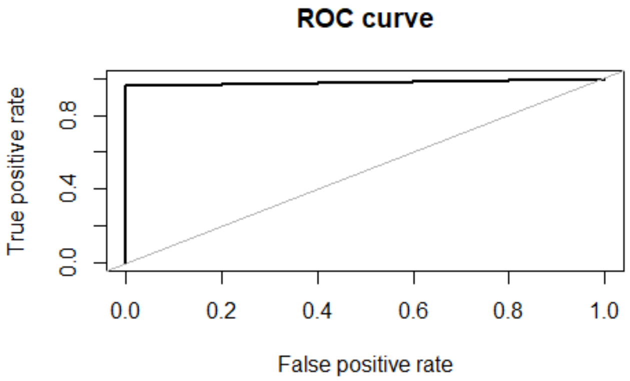
Here, we will evaluate the performance of different Classification models based on which training dataset they are trained with.

I. Decision Tree:

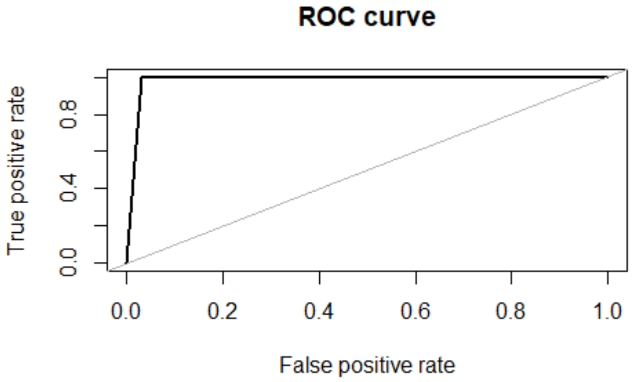
Error Metrics	Stratified Sampling	Random Over Sampling	Random Under Sampling	Random Both Over and Under Sampling	Synthetic Minority Over Sampling Technique
Accuracy	0.9945	0.9989	0.9727	0.9912	1
False Negative Rate	0.037	0	0.1592	0.0507	0
Sensitivity/Recall/TPR	1	0.9924	1	0.9924	1
AUC	0.9811	0.996	0.9841	0.992	1

Decision Tree ROC for different training sample:

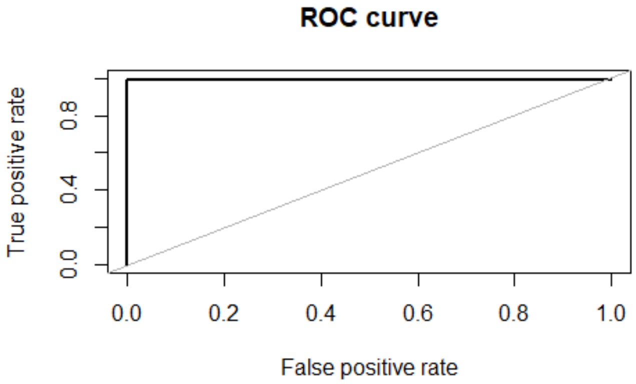
Stratified Sampling



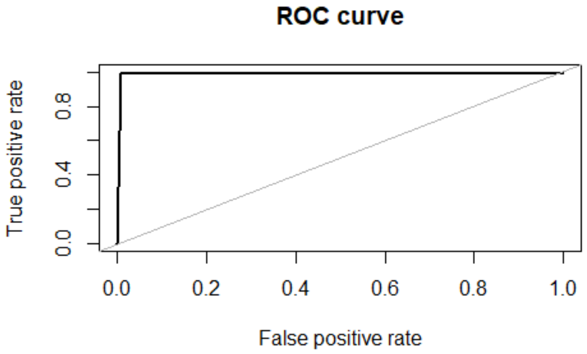
Random Under Sampling



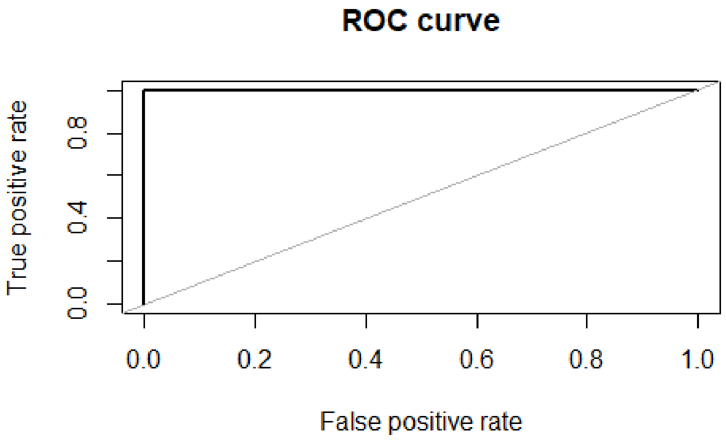
Random Over Sampling



Random Both Over and Under Sampling



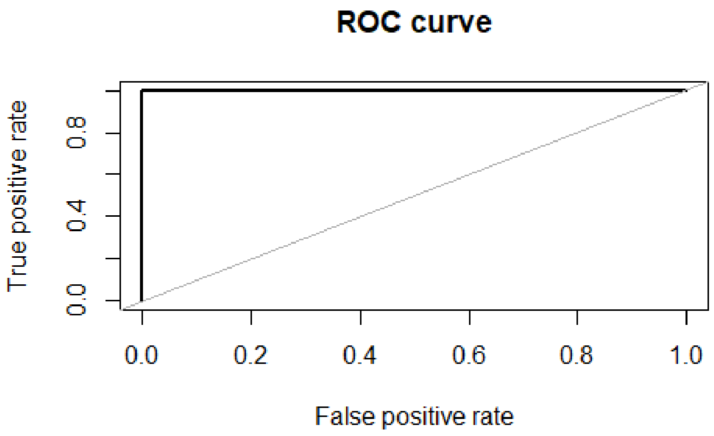
Synthetic Minority Over Sampling Technique



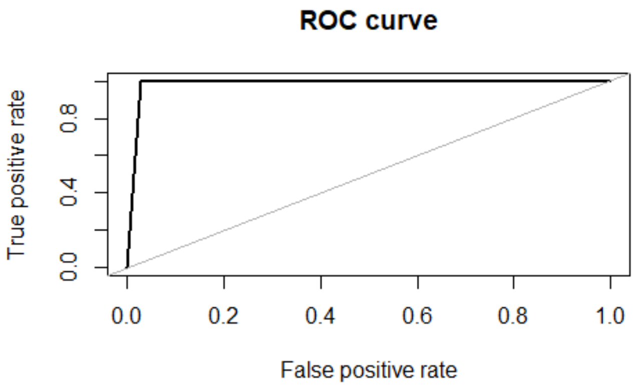
II. Random Forest: For ntrees = 500

Error Metrics	Stratified Sampling	Random Over Sampling	Random Under Sampling	Random Both Over and Under Sampling	Synthetic Minority Over Sampling Technique
Accuracy	1	0.9989	0.976	0.9890	1
False Negative Rate	0	0.0075	0	0.0075	0
Sensitivity/Recall/TPR	1	1	0.8571	0.9357	1
AUC	1	0.996	0.986	0.990	1

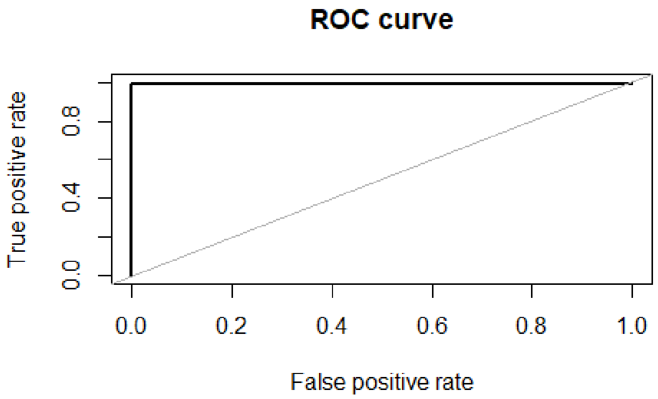
Random Forest ROC for different training sample:
Stratified Sampling



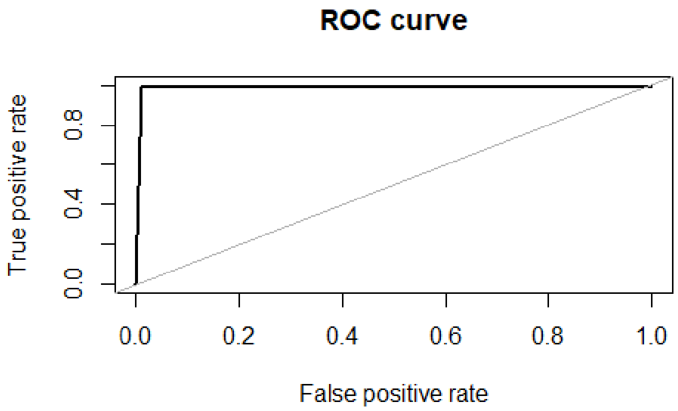
Random Under Sampling



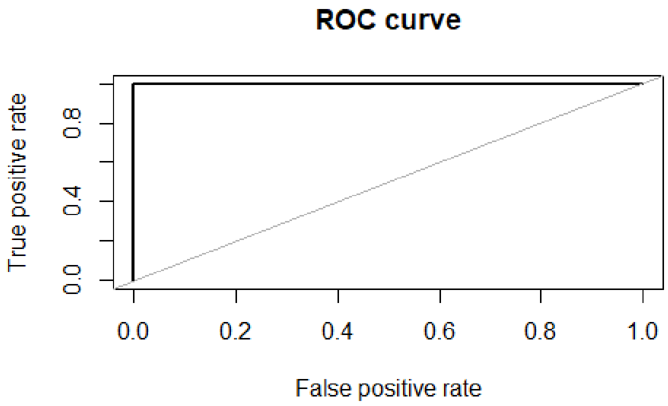
Random Over Sampling



Random Both Over and Under Sampling



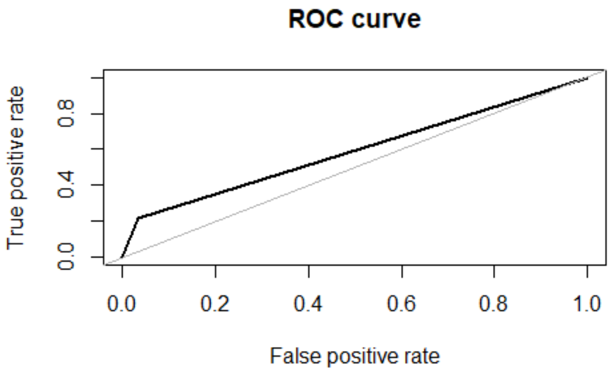
Synthetic Minority Over Sampling Technique



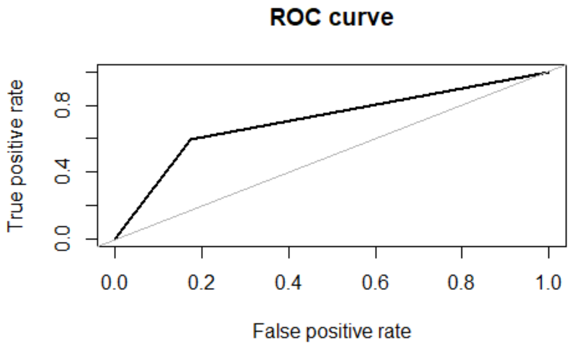
III. Logistic Regression: For p = 0.5

Error Metrics	Stratified Sampling	Random Over Sampling	Random Under Sampling	Random Both Over and Under Sampling	Synthetic Minority Over Sampling Technique
Accuracy	0.8593	0.8233	0.7938	0.8233	0.7808
False Negative Rate	0.7803	0.4469	0.4015	0.4469	0.3409
Sensitivity/Recall/TPR	0.5272	0.4147	0.3674	0.4147	0.3580
AUC	0.593	0.711	0.713	0.711	0.730

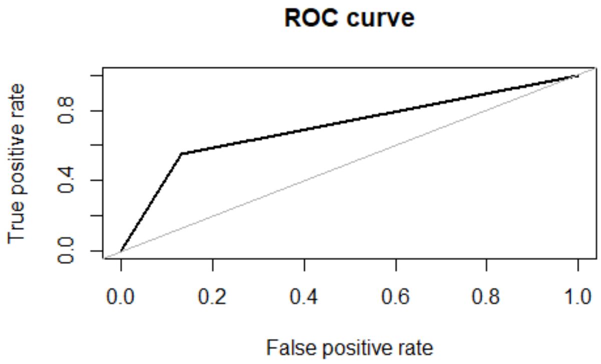
Logistic Regression ROC for different training sample:
Stratified Sampling



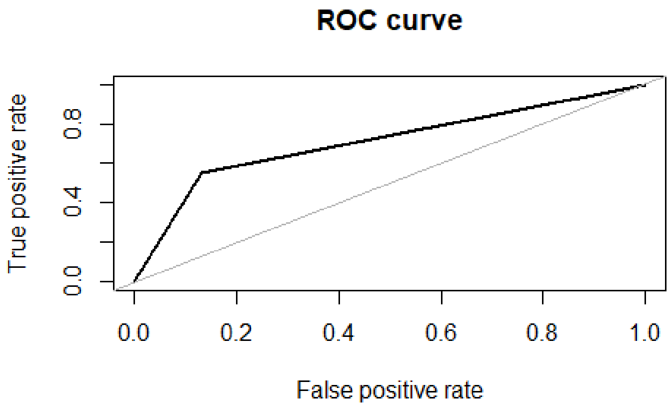
Random Under Sampling



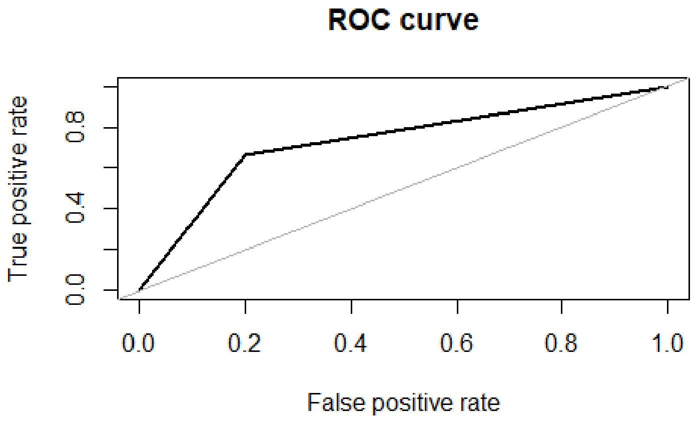
Random Over Sampling



Random Both Over and Under Sampling



Synthetic Minority Over Sampling Technique

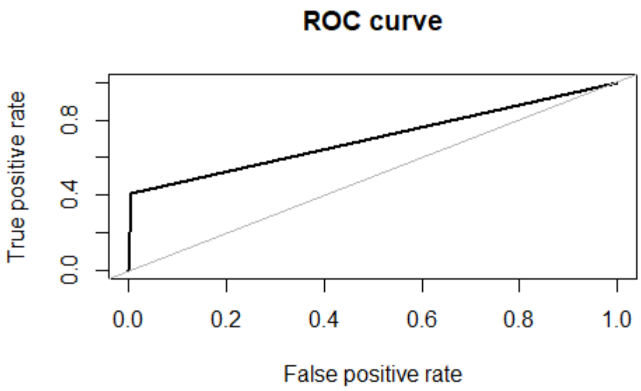


IV. KNN: K = 7

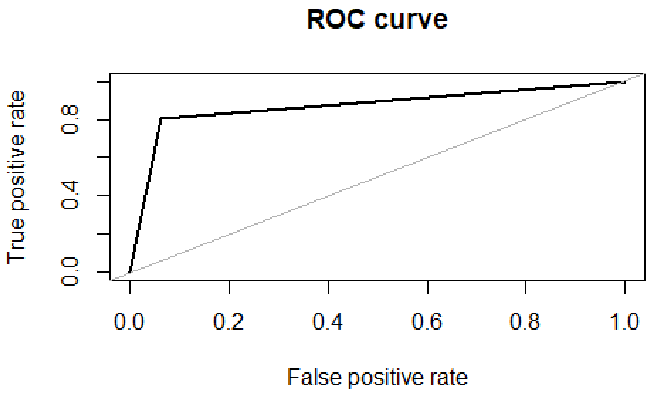
Error Metrics	Stratified Sampling	Random Over Sampling	Random Under Sampling	Random Both Over and Under Sampling	Synthetic Minority Over Sampling Technique
Accuracy	0.9116	0.8865	0.9193	0.8943	0.8964
False Negative Rate	0.5909	0.1893	0.1969	0.2348	0.015
Sensitivity/Recall/TPR	0.9473	0.5752	0.6883	0.6047	0.5829
AUC	0.703	0.855	0.871	0.841	0.933

KNN ROC for different training sample:

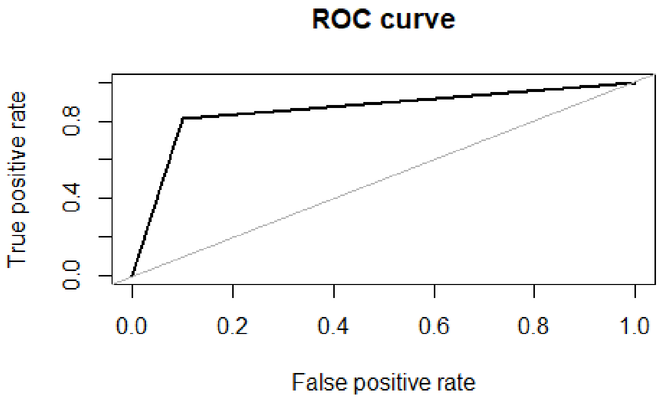
Stratified Sampling



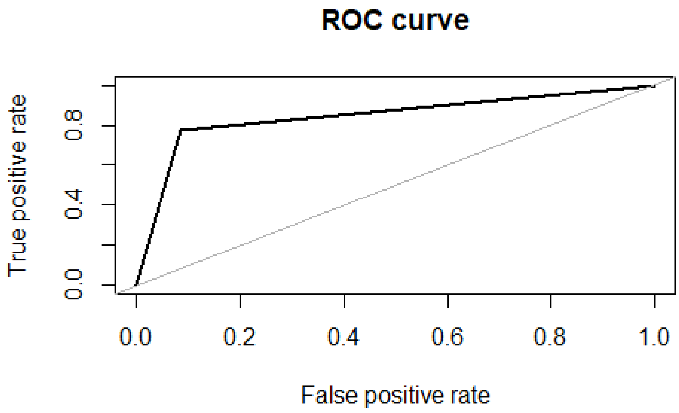
Random Under Sampling



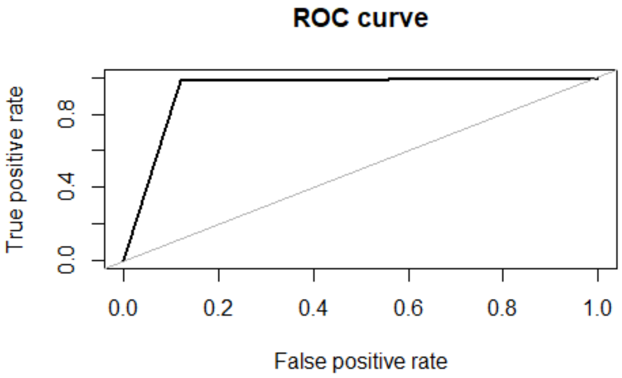
Random Over Sampling



Random Both Over and Under Sampling



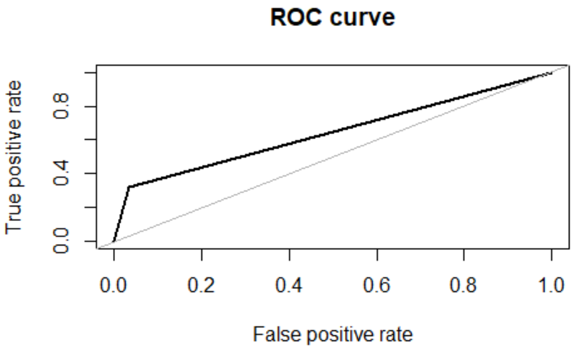
Synthetic Minority Over Sampling Technique



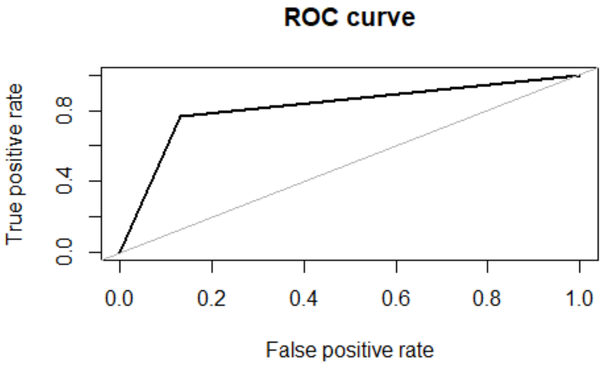
V. Naïve Bayes:

Error Metrics	Stratified Sampling	Random Over Sampling	Random Under Sampling	Random Both Over and Under Sampling	Synthetic Minority Over Sampling Technique
Accuracy	0.8724	0.8756	0.8538	0.8680	0.8200
False Negative Rate	0.6818	0.2575	0.2348	0.2272	0.2651
Sensitivity/Recall/TPR	0.6086	0.5505	0.4950	0.5284	0.4273
AUC	0.642	0.820	0.817	0.828	0.785

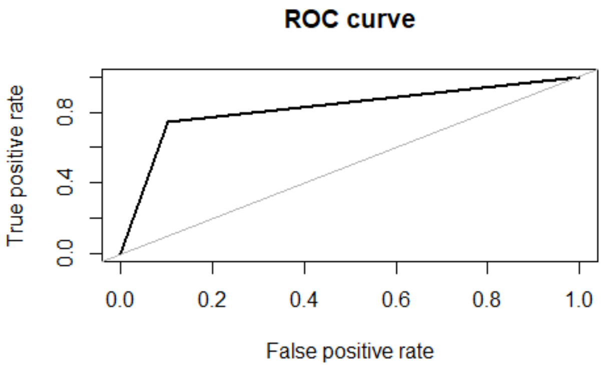
Naïve Bayes ROC for different training sample:
Stratified Sampling



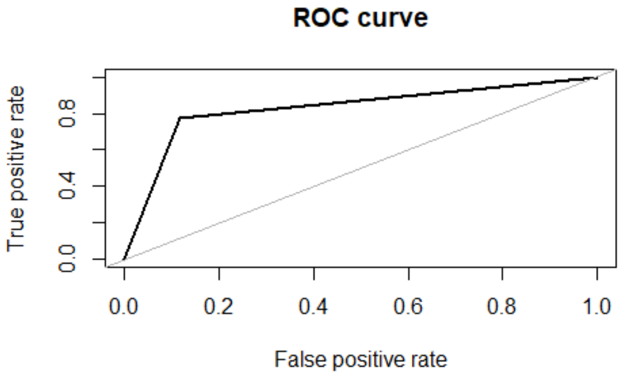
Random Under Sampling



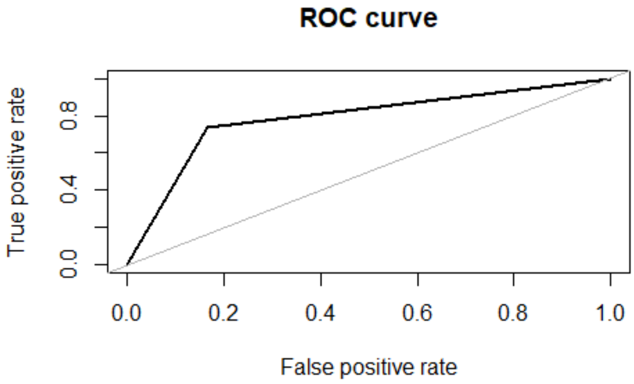
Random Over Sampling



Random Both Over and Under Sampling



Synthetic Minority Over Sampling Technique



Chapter 4

Conclusion

For Decision Tree - Synthetic Minority Over Sampling Technique performs well,

Accuracy	1
False Negative Rate	0
Sensitivity/Recall/TPR	1
AUC	1

For Random Forest both – Stratified Sampling and- Synthetic Minority Over Sampling Technique performs well,

Accuracy	1
False Negative Rate	0
Sensitivity/Recall/TPR	1
AUC	1

For Logistic Regression – - Synthetic Minority Over Sampling Technique performs well,

Accuracy	0.7808
False Negative Rate	0.3409
Sensitivity/Recall/TPR	0.3580
AUC	0.730

For KNN - Synthetic Minority Over Sampling Technique performs well,

Accuracy	0.8680
False Negative Rate	0.2272
Sensitivity/Recall/TPR	0.5284
AUC	0.828

For Naïve bayes - Random Both Over and Under Sampling performs well,

Accuracy	0.8680
False Negative Rate	0.2272
Sensitivity/Recall/TPR	0.5284
AUC	0.828

Chapter 5

R-Code

```
rm(list=ls(all=T))
setwd("C:/Users/admin/Documents/R files")
# Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", 
"Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','dplyr','class')

# install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

# Load the data
# also replacing empty strings if present with NA
train = read.csv("Train_data.csv")
test = read.csv("Test_data.csv")

# comibining train and test so that it will be easier to preprocess it together than separate
train=rbind(train,test)
rmExcept('train')
# Structure of data
str(train)
head(train,10)
# Univariate Analysis
train$area.code = as.factor(train$area.code)
# Since, we have to predict the churn score based on usage pattern
# so features which are not related to usage pattern will be removed from training data
# like "area code", "phone number"ans'state'are not important so we will remove them, we will remove them after chi-
square reduction
str(train)
# Missing Value Analysis
any(is.na(train))                               # It returns a FALSE value. So, there's no missing value.
# Summary gives all Stats and Also Count of NA values after Max.
summary(train)                                  # Also after calling Summary we can see that there are no NA's

##### Visualization #########
ggplot(train, aes(Churn, account.length)) +             # we can see that account.length is not statistically significant
  geom_boxplot()                                # in predicting Churn, thus we can remove it.

# churning of customer w.r.t international.plan
intlplanVSChurn <- train %>%
  select(international.plan,Churn) %>%
  group_by(international.plan)
ggplot(intlplanVSChurn, aes(fill=Churn, x=international.plan)) +
  geom_bar() + ggtitle("international.plan Vs Churn")

# churning of customer w.r.t voice.mail.plan
vmpVSChurn <- train %>%
  select(voice.mail.plan,Churn) %>%
  group_by(voice.mail.plan)
vmpVSChurn
ggplot(vmpVSChurn, aes(fill = Churn, x = voice.mail.plan)) +
  geom_bar()+ggtitle('voice.mail.plan Vs Churn')

# churning of customer w.r.t area.code
acVSChurn <- train %>%
  select(area.code,Churn) %>%
  group_by(area.code)
acVSChurn
ggplot(acVSChurn, aes(fill = Churn, x = area.code)) +
  geom_bar()+ggtitle('area.code Vs Churn')

# churning of customer w.r.t number.customer.service.calls
cscVSChurn <- train %>%
  select(number.customer.service.calls,Churn) %>%
  group_by(number.customer.service.calls)
cscVSChurn
ggplot(cscVSChurn, aes(fill = Churn, x = number.customer.service.calls)) +
  geom_bar()+ggtitle('number.customer.service.calls Vs Churn')

#################### converting our factor variables into numeric factor levels ######################
for(i in c(1:21)){

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

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

  }
}

# let's separate continuous data from categorical data for feature selection
numeric_index = vapply(train,is.numeric,logical(1))           # selecting only numeric(vapply is safer than sapply)
numeric_data = train[,numeric_index]
num_col = names(numeric_data)                                  # names() return colnames in data.frame
print(length(num_col))
################# Outlier detection using Boxplot #################
#outlier analysis works only on numerical variables
for (i in 1:length(num_col)){
  assign(paste0("gn",i),
         ggplot(aes_string(y = (num_col[i]), x = 'Churn'),data = train) +
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "skyblue",
                        outlier.shape=18,outlier.size=3, notch=FALSE) +
           labs(y=num_col[i],x="Churn"))
}
# Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
gridExtra::grid.arrange(gn13,gn14,gn15,ncol=3)

# There are outliers present in the data. But if we understand and think about the business model here, we get that the
dataset is related to customer usage pattern.
```

```
# therefore we come to conclusion that to leave the dataset with outliers to get the most out of the dataset and predict our target variable ‘Churn’.

####### Feature Selection ######
# Let's look at the correlation plot for only continuous data
corrgram(train[,num_col], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
# So we can see that in correlation plot dark blue indicates that 2 variables are highly correlated with each other
# here, total.day.minutes is highly correlated with total.day.charge
#       total.eve.minutes is highly correlated with total.eve.charge
#       total.night.minutes is highly correlated with total.night.charge
#       total.intl.minutes is highly correlated with total.intl.charge
# now instead of selecting all variables for our modelling, we would select one of each of them
# lets select all charges features-total.day.charge, total.eve.charge, total.night.charge, total.intl.charge
# and drop all minutes features-total.day.minutes, total.eve.minutes, total.night.minutes, total.intl.minutes
# we will drop them after chi-sa=quare analysis
# Now Chi-square test of independence for Categorical data
# As chi-square is only for categorical variables so we will select only categorical variables from training data
cat_index = vapply(train,is.factor,logical(1))  # selecting only numeric(vapply is safer than sapply)
cat_data = train[,cat_index]
cat_col = names(cat_data)                    # names() return colnames in data.frame
cat_col
# Applying Chi-square test of independence for Categorical data
for (i in 1:5){
  print(names(cat_data)[i])
  print(chisq.test(table(cat_data$Churn,cat_data[,i])))
}
# If p-value is less than 0.05 then we reject the null hypothesis saying that 2 variables are dependent and if
# p-value is greater than 0.05 then we accept the null hypothesis saying that 2 variables are independent.
# If p-value<0.05 then select the variable,
# If p-value>0.05 then ignore the variable as this independent variable doesn't carry any information to explain the dependent variable therefore it will not add any information while modelling.
# after analyzing p value of all categorical features, we come to a conclusion that, feature- phone.number,area.code will be removed
# and account.length
# Dimensionality Reduction
train = subset(train,
             select = -c(state,account.length,area.code,phone.number,total.day.minutes,total.eve.minutes,total.night.minutes,total.intl.minutes))
str(train)

# Again select only numeric and only categorical variables separately from our new data
numeric_index = vapply(train,is.numeric,logical(1))           # selecting only numeric(vapply is safer than sapply)
numeric_data = train[,numeric_index]
num_col = names(numeric_data)
cat_index = vapply(train,is.factor,logical(1))               # selecting only numeric(vapply is safer than sapply)
cat_data = train[,cat_index]
cat_col = names(cat_data)
###### Feature Scaling ######
# It is performed only on Continuous Variables

library(car)
dev.off()
par(mfrow=c(1,2))
```

```
qqPlot(train$number.vmail.messages)                        # qqPlot, it has a x values derived from gaussian distribution, if data is distributed normally then the sorted data points should lie very close to the solid reference line
truehist(train$number.vmail.messages)                       # truehist() scales the counts to give an estimate of the probability density.
lines(density(train$number.vmail.messages))  # left skewed      # lines() and density() functions to overlay a density plot on histogram

qqPlot(train$total.day.calls)
truehist(train$total.day.calls)
lines(density(train$total.day.calls))       # normal

qqPlot(train$total.day.charge)
truehist(train$total.day.charge)
lines(density(train$total.day.charge))      # normal

qqPlot(train$total.eve.calls)
truehist(train$total.eve.calls)
lines(density(train$total.eve.calls))       # normal

qqPlot(train$total.eve.charge)
truehist(train$total.eve.charge)
lines(density(train$total.eve.charge))      # normal

qqPlot(train$total.night.calls)
truehist(train$total.night.calls)
lines(density(train$total.night.calls))      # normal

qqPlot(train$total.night.charge)
truehist(train$total.night.charge)
lines(density(train$total.night.charge))     # normal

qqPlot(train$total.intl.charge)
truehist(train$total.intl.charge)
lines(density(train$total.intl.charge))      # almost normal

qqPlot(train$total.intl.calls)
truehist(train$total.intl.calls)
lines(density(train$total.intl.calls))       # left skewed

qqPlot(train$number.customer.service.calls)
truehist(train$number.customer.service.calls)
lines(density(train$number.customer.service.calls))# exibits multimodal distribution
# After plotting all the graphs we can say that most of the variables are normally distributed
# therefore, we will go for standardization over normalization as our most of the data is distributed normally
# Standardization
for(i in num_col){
  print(i)
  train[,i] = (train[,i] - mean(train[,i]))/sd(train[,i])
}
summary(train) # we can see now our data is standarized
# Z is negative when the raw score is below the mean and Z is positive when above mean.

############## A check for Target class imbalance problem for categorical Target Variable-Churn ##########
```

```
imbalance = data.frame(table(train$Churn))                    # False : 4293 & True : 707,
colnames(imbalance) = c('Churn_categories','Churn_count')       # i.e. Ratio=4293:707, event
rate=(707/5000)*100=14.14%
ggplot(imbalance, aes(x = Churn_categories, y = Churn_count))+
  geom_col()+ggtitle('Churn Distribution Graph')

# As sampling is only applied on training dataset, so lets divide our whole data into train and test
dim(train)
df =train
#train = df
set.seed(2019)
train.index = createDataPartition(train$Churn, p = .75, list = FALSE)
train = train[ train.index,]
test  = train[-train.index,]
dim(train)
dim(test)
prop.table(table(train$Churn)) # 1=FALSE,2=TRUE,FALSE = 85.84% & TRUE = 14.15%

#1. Random Over Sampling
library(ROSE)
table(train$Churn)
sampling_result=ovun.sample(formula = Churn~. , data=train, method = "over" , N= 3220/(1-0.3333),seed=2019)
# selecting only data from large ovun sample
train_over = sampling_result$data
table(train_over$Churn)
prop.table(table(train_over$Churn))

#2. Random under Sampling
table(train$Churn)
sampling_result=ovun.sample(formula = Churn~. , data=train, method = "under" , N= 483/0.333,seed=2019)
# selecting only data from large ovun sample
train_under = sampling_result$data
table(train_under$Churn)
prop.table(table(train_under$Churn))


# 3. Combining under and over sampling
# Defining desired no. of new cases in balanced dataset and fraction of TRUE cases
n_new = 4000
true_fraction = 0.34
sampling_result=ovun.sample(formula = Churn~. , data=train, method = "both" , N= n_new,p =
true_fraction,seed=2019)
# selecting only data from large ovun sample
train_both = sampling_result$data
table(train_both$Churn)
prop.table(table(train_both$Churn))

# 4. SMOTE-can only be applied on numeric variables since it uses the euclidean distance to determine nearest
neighbors.
library(smotefamily)
table(train$Churn)
# Setting the number of TRUE and FALSE cases, and the desired percentage of FALSE cases
true_cases = 531
false_cases = 3220
desired_precent = 0.6
```

```
# Calculate the value for the 'dup_size' parameter of SMOTE
# ntimes is the desired value such that the over-sampled dataset contains 60% FALSE cases
ntimes <- ((1 - desired_precent) / desired_precent) * (false_cases / true_cases) - 1
# Creating synthetic TRUE cases with SMOTE
names(train)
str(train)
smote_output = SMOTE(X =train[ , -c(1, 2, 13)], target = train$Churn, K = 5, dup_size = ntimes)
train_smote <- smote_output$data
str(train_smote)
names(train_smote)[11]<- "Churn"
train_smote$Churn = as.factor(train_smote$Churn)
table(train_smote$Churn)
prop.table(table(train_smote$Churn))                    # now True cases is 0.397% and False is 0.602%

# Model development #
#function for calculating error metrics
error_metric = function(cm){
  TN = cm[1,1]
  FP = cm[1,2]
  FN = cm[2,1]
  TP = cm[2,2]
  print(paste0('Accuracy- ',((TN+TP)/(TN+TP+FN+FP))*100))
  print(paste0('FNR- ',((FN)/(TP+FN))*100))
  print(paste0('Sensitivity/Recall/TPR-  ',((TP)/(TP+FP))*100))
}

######### C5.0 #######

##Decision tree for classification
#Develop Model on training data of sampled train
C50_model = C5.0(Churn ~., train, trials = 100, rules = TRUE)
# Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-13], type = "class")
##Evaluate the performance of classification model
ConfMatrix_C50 = table(actual = test$Churn,predictions = C50_Predictions)
# Because our problem statement in more focused to Churn reduction, we are interested in knowing the customers who are going too churn out
# so our positive class is -  True cases and true case is labelled as 2
# so adding a parameter positive to define our positive class and assigning it to 2.
confusionMatrix(ConfMatrix_C50,positive = "2")
error_metric(ConfMatrix_C50)
# sensitivity = 1, accuracy = 0.9945, FNR =0.037
# according to problem statement we want to correctly identify people who will Churn out i.e Sensitivity/Recall/TPR
# though we can see here accuracy and fnr is good, we know that our model is trained on unbalanced dataset,
# and here accuracy is misleading because our model is predicting the majority class.
# as classifier tend to favour majority class.
# Therefore, we will use AUC and ROC which is better performance metric.
# Area under ROC curve
roc.curve(test$Churn,C50_Predictions)
#Area under the curve for sampled train: 0.9811

#Develop Model on training data of under sampled -train_under
C50_model = C5.0(Churn ~., train_under, trials = 100, rules = TRUE)
# Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-13], type = "class")
```

```
# Evaluate the performance of classification model
ConfMatrix_C50 = table(predictions = C50_Predictions, actual = test$Churn)
confusionMatrix(ConfMatrix_C50,positive = "2")
error_metric(ConfMatrix_C50)
# sensitivity = 1, accuracy = 0.9727, FNR = 0.1592
roc.curve(test$Churn,C50_Predictions)
#Area under the curve for train_under: 0.9841

#Develop Model on training data of over sampled -train_over
C50_model = C5.0(Churn ~., train_over, trials = 100, rules = TRUE)
# Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-13], type = "class")
# Evaluate the performance of classification model
ConfMatrix_C50 = table(predictions = C50_Predictions, actual = test$Churn)
confusionMatrix(ConfMatrix_C50,positive = "2")
error_metric(ConfMatrix_C50)
# sensitivity = 0.9924, accuracy = 0.9989, FNR =0
roc.curve(test$Churn,C50_Predictions)
#Area under the curve for train_over: 0.9962

#Develop Model on training data of both-over and under sampled-train_both
C50_model = C5.0(Churn ~., train_both, trials = 100, rules = TRUE)
# Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-13], type = "class")
# Evaluate the performance of classification model
ConfMatrix_C50 = table(predictions = C50_Predictions, actual = test$Churn)
confusionMatrix(ConfMatrix_C50,positive = "2")
error_metric(ConfMatrix_C50)
# sensitivity = 0.9924, accuracy = 0.9912, FNR =0.050
roc.curve(test$Churn,C50_Predictions)
#Area under the curve for train_both: 0.992

#Develop Model on training data of over sampled synthetic- train_smote
C50_model = C5.0(Churn ~., train_smote, trials = 100, rules = TRUE)
# Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-13], type = "class")
# Evaluate the performance of classification model
ConfMatrix_C50 = table(predictions = C50_Predictions, actual = test$Churn)
confusionMatrix(ConfMatrix_C50,positive = "2")
error_metric(ConfMatrix_C50)
# sensitivity = 1, accuracy = 1, FNR =0
roc.curve(test$Churn,C50_Predictions)
#Area under the curve for train_smote: 1

#Logistic Regression on sampled train-train
logit_model = glm(Churn ~ ., data = train, family = "binomial")
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
# Confusion matrix
ConfMatrix_LR = table(test$Churn,logit_Predictions>0.5)# can adjust p to correctly classify TRUE cases
ConfMatrix_LR
error_metric(ConfMatrix_LR)
# accuracy = 0.8593, sensitivity =  0.5272, FNR =0.7803
roc.curve(test$Churn,logit_Predictions>0.5)
#Area under the curve for train: 0.593
```

```
#Logistic Regression on under sampled train-train_under
logit_model = glm(Churn ~ ., data = train_under, family = "binomial")
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
# Confusion matrix
ConfMatrix_LR = table(test$Churn,logit_Predictions>0.5)# can adjust p to correctly classify TRUE cases
ConfMatrix_LR
error_metric(ConfMatrix_LR)
# accuracy = 0.7938, sensitivity =  0.3674, FNR =0.4015
roc.curve(test$Churn,logit_Predictions>0.5)
#Area under the curve for train_under: 0.713

#Logistic Regression on over sampled train-train_over
logit_model = glm(Churn ~ ., data = train_over, family = "binomial")
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
# Confusion matrix
ConfMatrix_LR = table(test$Churn,logit_Predictions>0.5)# can adjust p to correctly classify TRUE cases
ConfMatrix_LR
error_metric(ConfMatrix_LR)
# accuracy = 0.8233, sensitivity =  0.4147, FNR = 0.4469
roc.curve(test$Churn,logit_Predictions>0.5)
#Area under the curve for train_over: 0.711

#Logistic Regression on both over and under sampled train-train_both
logit_model = glm(Churn ~ ., data = train_both, family = "binomial")
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
# Confusion matrix
ConfMatrix_LR = table(test$Churn,logit_Predictions>0.5)# can adjust p to correctly classify TRUE cases
ConfMatrix_LR
error_metric(ConfMatrix_LR)
# accuracy = 0.8233, sensitivity =  0.4147, FNR = 0.4469
roc.curve(test$Churn,logit_Predictions>0.5)
#Area under the curve for train_both: 0.711

#Logistic Regression on over sampled synthetically train-train_smote
logit_model = glm(as.factor(Churn) ~ ., data = train_smote, family = "binomial")
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "response")
# Confusion matrix
ConfMatrix_LR = table(test$Churn,logit_Predictions>0.5)# can adjust p to correctly classify TRUE cases
ConfMatrix_LR
error_metric(ConfMatrix_LR)
# accuracy = 0.7808, sensitivity =  0.3580, FNR = 0.3409
roc.curve(test$Churn,logit_Predictions>0.5)
#Area under the curve for train_smote: 0.730

# Random Forest on sampled train-train
RF_model = randomForest( Churn~ ., train, importance = TRUE, ntree = 500)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-13])
##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
```

```
confusionMatrix(ConfMatrix_RF,positive = '2')
error_metric(ConfMatrix_RF)
# accuracy = 1, sensitivity =  1,, FNR =0
roc.curve(test$Churn,RF_Predictions)
#Area under the curve for train_:1

# Random Forest on under sampled train-train_under
RF_model = randomForest( Churn~ ., train_under, importance = TRUE, ntree = 500)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-13])
##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF,positive = '2')
error_metric(ConfMatrix_RF)
# accuracy = 0.976, sensitivity =  0.8571, FNR =0
roc.curve(test$Churn,RF_Predictions)
#Area under the curve for train_under:0.986


# Random Forest on over sampled train-train_over
RF_model = randomForest( Churn~ ., train_over, importance = TRUE, ntree = 500)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-13])
##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF,positive = '2')
error_metric(ConfMatrix_RF)
# accuracy = 0.9989, sensitivity =  1, FNR =0.0075
roc.curve(test$Churn,RF_Predictions)
#Area under the curve for train_over:0.996


# Random Forest on both over and under sampled train-train_both
RF_model = randomForest( Churn~ ., train_both, importance = TRUE, ntree = 500)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-13])
##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF,positive = '2')
error_metric(ConfMatrix_RF)
# accuracy = 0.9890, sensitivity =  0.9357, FNR =0.0075
roc.curve(test$Churn,RF_Predictions)
#Area under the curve for train_both:0.990

# Random Forest on over sampled synthetically train-train_smote
RF_model = randomForest( Churn~ ., train_smote, importance = TRUE, ntree = 500)
#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-13])
##Evaluate the performance of classification model
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF,positive = '2')
error_metric(ConfMatrix_RF)
# accuracy = 1, sensitivity =  1, FNR =0
roc.curve(test$Churn,RF_Predictions)
#Area under the curve for train_smote:1
```

```
# KNN on sampled train-train
KNN_Predictions = knn(train[-13], test[-13], train$Churn, k = 7)
#Confusion matrix
ConfMatrix_KNN = table(test$Churn,KNN_Predictions)
confusionMatrix(ConfMatrix_KNN,positive = '2')
error_metric(ConfMatrix_KNN)
# accuracy = 0.9116, sensitivity =  0.9473,FNR = 0.5909
roc.curve(test$Churn,KNN_Predictions)
#Area under the curve for train_smote:0.703

# KNN on under sampled train-train_under
KNN_Predictions = knn(train_under[-13], test[-13], train_under$Churn, k = 7)
#Confusion matrix
ConfMatrix_KNN = table(test$Churn,KNN_Predictions)
confusionMatrix(ConfMatrix_KNN,positive = '2')
error_metric(ConfMatrix_KNN)
# accuracy = 0.9193, sensitivity =  0.6883,FNR = 0.1969
roc.curve(test$Churn,KNN_Predictions)
#Area under the curve for train_under:0.871

# KNN on over sampled train-train_over
KNN_Predictions = knn(train_over[-13], test[-13], train_over$Churn, k = 7)
#Confusion matrix
ConfMatrix_KNN = table(test$Churn,KNN_Predictions)
confusionMatrix(ConfMatrix_KNN,positive = '2')
error_metric(ConfMatrix_KNN)
# accuracy = 0.8865, sensitivity =  0.5752,FNR =0.1893
roc.curve(test$Churn,KNN_Predictions)
#Area under the curve for train_over:0.855

# KNN on both over and under sampled train-train_both
KNN_Predictions = knn(train_both[-13], test[-13], train_both$Churn, k = 7)
#Confusion matrix
ConfMatrix_KNN = table(test$Churn,KNN_Predictions)
confusionMatrix(ConfMatrix_KNN,positive = '2')
error_metric(ConfMatrix_KNN)
# accuracy = 0.8942, sensitivity =  0.6047,FNR =0.2348
roc.curve(test$Churn,KNN_Predictions)
#Area under the curve for train_both:0.843

# KNN on over sampled synthetically train-train_smote
KNN_Predictions = knn(train_smote[-11], test[,-c(1,2,13)], train_smote$Churn, k = 7)
#Confusion matrix
ConfMatrix_KNN = table(test$Churn,KNN_Predictions)
confusionMatrix(ConfMatrix_KNN,positive = '2')
error_metric(ConfMatrix_KNN)
# accuracy = 0.8964, sensitivity =  0.5829,FNR =0.015
roc.curve(test$Churn,KNN_Predictions)
#Area under the curve for train_smote:0.933

# Naive Bayes on sampled train-train
NB_model = naiveBayes(Churn ~ ., data = train)
#predict on test cases
NB_Predictions = predict(NB_model, test[-13], type = 'class')
```

```
#Confusion matrix
ConfMatrix_NB = table(test$Churn,NB_Predictions)
confusionMatrix(ConfMatrix_NB,positive = '2')
error_metric(ConfMatrix_NB)
# accuracy = 0.8724, sensitivity =  0.6086,FNR =0.6818
roc.curve(test$Churn,NB_Predictions)
#Area under the curve for train:0.642

# Naive Bayes on under sampled train-train_under
NB_model = naiveBayes(Churn ~ ., data = train_under)
#predict on test cases #raw
NB_Predictions = predict(NB_model, test[-13], type = 'class')
#Confusion matrix
ConfMatrix_NB = table(test$Churn,NB_Predictions)
confusionMatrix(ConfMatrix_NB,positive = '2')
error_metric(ConfMatrix_NB)
# accuracy = 0.8538, sensitivity =  0.4950,FNR =0.2348
roc.curve(test$Churn,NB_Predictions)
#Area under the curve for train_under:0.817

# Naive Bayes on over sampled train-train_over
NB_model = naiveBayes(Churn ~ ., data = train_over)
#predict on test cases #raw
NB_Predictions = predict(NB_model, test[-13], type = 'class')
#Confusion matrix
ConfMatrix_NB = table(test$Churn,NB_Predictions)
confusionMatrix(ConfMatrix_NB,positive = '2')
error_metric(ConfMatrix_NB)
# accuracy = 0.8756, sensitivity =  0.5505,FNR =0.2575
roc.curve(test$Churn,NB_Predictions)
#Area under the curve for train_over:0.820

# Naive Bayes on both over and under sampled train-train_both
NB_model = naiveBayes(Churn ~ ., data = train_both)
#predict on test cases
NB_Predictions = predict(NB_model, test[-13], type = 'class')
#Confusion matrix
ConfMatrix_NB = table(test$Churn,NB_Predictions)
confusionMatrix(ConfMatrix_NB,positive = '2')
error_metric(ConfMatrix_NB)
# accuracy = 0.8680, sensitivity =  0.5284,FNR =0.2272
roc.curve(test$Churn,NB_Predictions)
#Area under the curve for train_both:0.828

# Naive Bayes on over sampled synthetically train-train_smote
NB_model = naiveBayes(Churn ~ ., data = train_smote)
#predict on test cases
NB_Predictions = predict(NB_model,test[,-c(1,2,13)], type = 'class')
#Confusion matrix
ConfMatrix_NB = table(test$Churn,NB_Predictions)
confusionMatrix(test$Churn,NB_Predictions,positive = '2')
error_metric(ConfMatrix_NB)
# accuracy = 0.8200, sensitivity =  0.4273,FNR =0.2651
roc.curve(test$Churn,NB_Predictions)
#Area under the curve for train_smote:0.785
```