Predicting Churn

Jagdish Kumar A

13th August 2018

Contents

Chapter 1

Introduction

1.1 Problem Description

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.

1.2 Data Sets -

1) Test_data.csv
2) Train_data.csv

1.3 Problem statement -

The objective of this Case is to predict customer behaviour. We are providing you a public dataset that has customer usage pattern and if the customer has moved or not. We expect you to develop an algorithm to predict the churn score based on usage pattern. 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

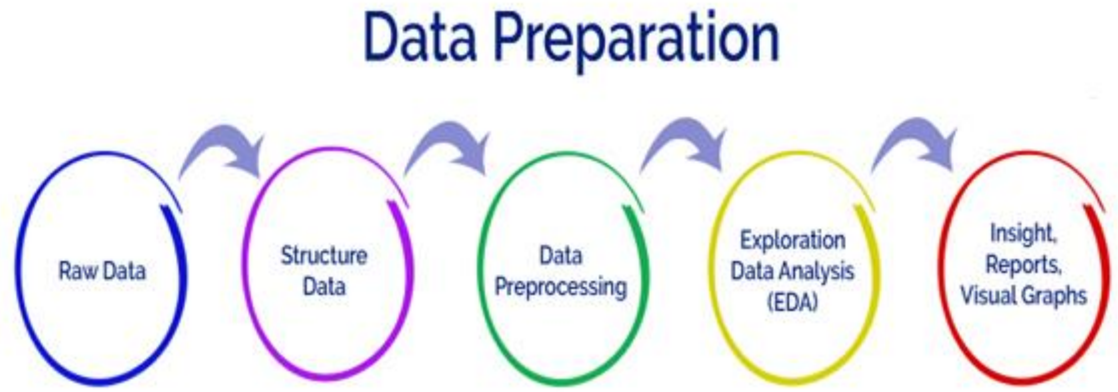
Target Variable :
move: if the customer has moved (1=yes; 0 = no)

Chapter 2

Methodology

2.1 Pre Processing

Data Preparation is an important part of Data Science. It includes two concepts such as **Data Cleaning** and **Feature Engineering**. These two are compulsory for achieving better accuracy and performance in the Machine Learning and Deep Learning projects.



Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of **Iterative Analysis**. The set of steps is known as Data Preprocessing. It includes -

- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

2.1.1 Missing Values

This dataset has no missing values which can be found with this line of code in R

missing_val = data.frame(apply(train_data,2,function(x){sum(is.na(x))}))

```
> missing_val
                              apply.train_data..2..function.x...
state                                                  0
account.length                                         0
area.code                                              0
phone.number                                           0
international.plan                                     0
voice.mail.plan                                        0
number.vmail.messages                                  0
total.day.minutes                                      0
total.day.calls                                        0
total.day.charge                                       0
total.eve.minutes                                      0
total.eve.calls                                        0
total.eve.charge                                       0
total.night.minutes                                    0
total.night.calls                                      0
total.night.charge                                     0
total.intl.minutes                                     0
total.intl.calls                                       0
total.intl.charge                                      0
number.customer.service.calls                          0
Churn                                                  0
```

2.1.2 Outlier Analysis

One of the other steps of **pre-processing** apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers, *Tukey's method*. We visualize the outliers using *boxplots*
.
In figure 1 and figure 2, we have plotted the boxplots for each predictor variable with respect to target variable Churn. A lot of useful inferences can be made from these plots. First as you can see, we have a lot of outliers and extreme values in each of the data set.

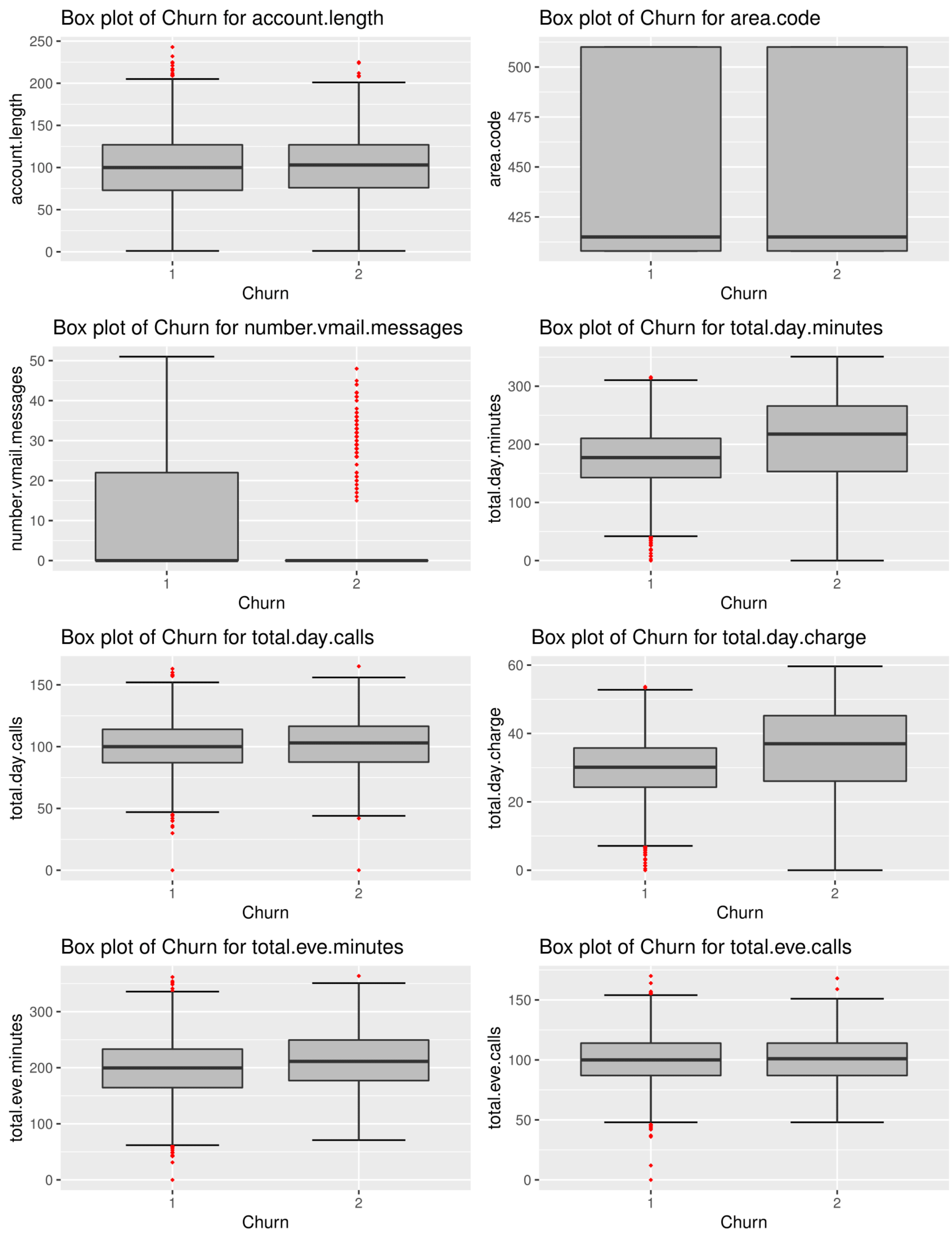


Figure 1 Boxplots for each predictor with respect to target variable Churn

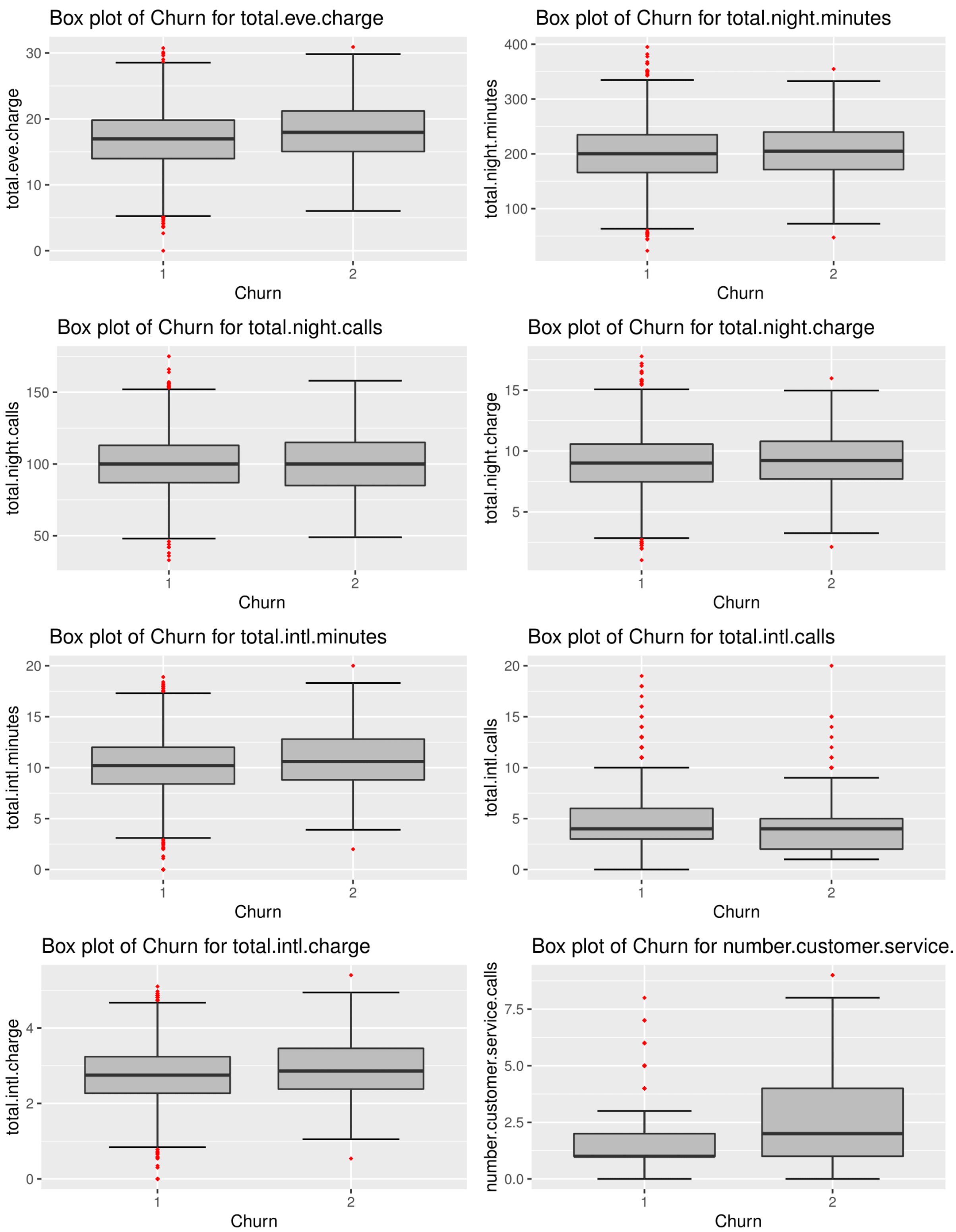


Figure 2 Boxplots for each predictor with respect to target variable Churn

After performing outlier analysis and removed outliers by replacing with NA. After trying with mean, median and knn imputation, I found median method is close to actual value hence imputed NA values using median method.

2.1.3 Feature Selection

Before performing any type of modelling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that.

For numeric(continuous) variables I used corrgram(correlation) plot to check for correlation analysis and found total.day.minutes and total.day.charge; total.eve.minutes and total.eve.charge; total.night.minutes and total.night.charge; total.intl.minutes and total.intl.charge are positively correlated with each other which can seen in figure 3. So I can drop one variable from these four sets.

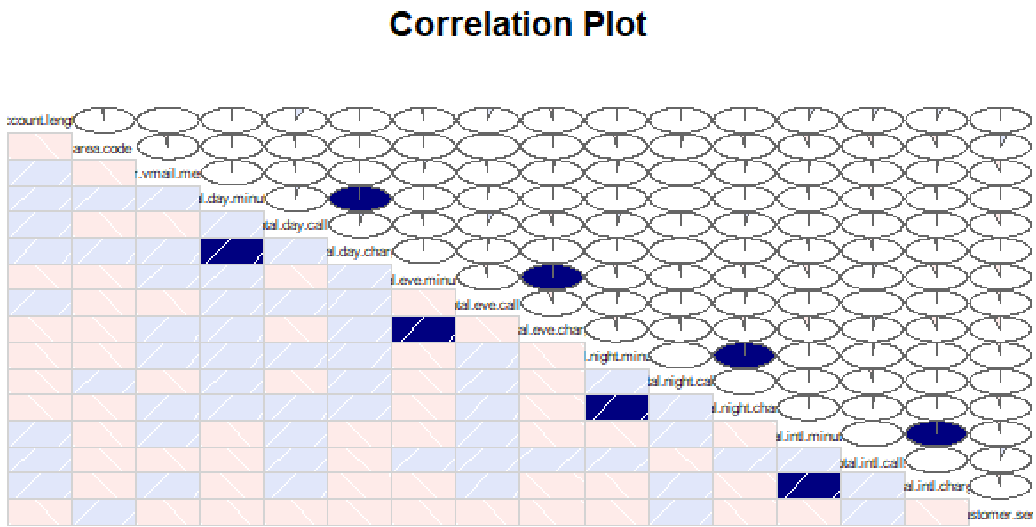


Figure 3 correlation plot of continuous variables.

For categorical variables, I used 'chi-square test of independence' and I selected variables p value less than 0.05 rejecting null hypothesis (these two variables are depend on each other) and rejected variables whose p value is greater than 0.05 as our null hypothesis is true ie these variables are dependent.

After correlation plot and chi-square test of independence I reduced dimensions by dropping variables like total.day.charge, total.eve.charge, total.intl.charge, total.night.charge, and phone.number.

2.1.4 Normalization

As data is not normally distributed I normalized the data for numerical variables using a loop to fit into the model and the data ranges from 0 and 1.

2.2 Modelling

2.2.1 Model Selection

As our problem statement states that we have to reduce churn and predict whether the customer is moving out or not. The dependent variable, in our case Churn, is yes or no, the only predictive analysis that we can perform is **Classification**.

You always start your model building from the most simplest to more complex. Therefore we used decision trees, random forest, logistic regression, KNN, and Naive Bayes,

2.2.2 Classification

```
> confusionMatrix(ConfMatrix_DT)
Confusion Matrix and Statistics

   DT_Predictions
       1    2
  1 1442    1
  2  107  117

               Accuracy : 0.9352
                 95% CI : (0.9223, 0.9466)
    No Information Rate : 0.9292
    P-Value [Acc > NIR] : 0.1827

                  Kappa : 0.6519
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9309
            Specificity : 0.9915
         Pos Pred Value : 0.9993
         Neg Pred Value : 0.5223
             Prevalence : 0.9292
         Detection Rate : 0.8650
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9612

       'Positive' Class : 1
```

With the help of confusion matrix for decision tree predictions you can find 'accuracy' which is **93.5%** and from the formula FNR = FN/FN+TP we can find 'False Negative Rate' giving us **47.7%**

2.2.3 Random Forest

```
> confusionMatrix(ConfMatrix_RF)
Confusion Matrix and Statistics

   RF_Predictions
      1    2
  1 1435    8
  2  114  110

             Accuracy : 0.9268
               95% CI : (0.9132, 0.9389)
  No Information Rate : 0.9292
  P-Value [Acc > NIR] : 0.6703

                Kappa : 0.6068
 Mcnemar's Test P-Value : <2e-16

          Sensitivity : 0.9264
          Specificity : 0.9322
       Pos Pred Value : 0.9945
       Neg Pred Value : 0.4911
           Prevalence : 0.9292
       Detection Rate : 0.8608
 Detection Prevalence : 0.8656
    Balanced Accuracy : 0.9293

      'Positive' Class : 1
```

We got accuracy of 92.6% and False Negative Rate of 50.8%

From logistic regression, we got accuracy 88.4 and FNR 72.7%

From KNN, I got accuracy 88.1% and FNR 35.2%

2.2.4 Naive Bayes,

```
> confusionMatrix(Conf_matrix_NB)
Confusion Matrix and Statistics

        predicted
observed    1    2
       1 1425   18
       2  171   53

             Accuracy : 0.8866
               95% CI : (0.8704, 0.9015)
  No Information Rate : 0.9574
  P-Value [Acc > NIR] : 1
```

```
                  Kappa : 0.315
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8929
            Specificity : 0.7465
         Pos Pred Value : 0.9875
         Neg Pred Value : 0.2366
             Prevalence : 0.9574
         Detection Rate : 0.8548
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8197

       'Positive' Class : 1
```

I got accuracy of 88.6% and FNR of 76.33%

Chapter 3
Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose.

There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:
1. Accuracy
2. False Negative Rate

3.1.1 Accuracy

Accuracy is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous section.

3.1.2 False Negative Rate

FNR can be obtained as follows from confusion matrix

FNR = False Negative/(False Negative + True Positive)

3.2 Model Selection

We can see that Decision Trees and KNN methods give us high accuracy and low FNR but KNN is a lazy method I fixed decision tree model for this dataset.

Appendix

Complete R file

```
setwd("D:/edwisor/Projects/Churn Reduction")
train_data <- read.csv("D:/edwisor/Projects/Churn
Reduction/train_data.csv", header = TRUE, sep = ",")
test_data <- read.csv("D:/edwisor/Projects/Churn
Reduction/test_data.csv", header = T, sep = ",")
#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest",
"unbalanced", "C50", "dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees',
'class')

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

#checking for missing values
missing_val =
data.frame(apply(train_data,2,function(x){sum(is.na(x))}))

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

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

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

  }
}

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

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

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

  }
}
```

```
# ## BoxPlots - Distribution and Outlier Check
numeric_index = sapply(train_data,is.numeric) #selecting only numeric

numeric_data = train_data[,numeric_index]

cnames = colnames(numeric_data)

 for (i in 1:length(cnames))
 {
   assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"),
data = subset(train_data))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey"
,outlier.shape=18,
                     outlier.size=1, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(y=cnames[i],x="Churn")+
          ggtitle(paste("Box plot of Churn for",cnames[i])))
 }

# ## Plotting plots together
 gridExtra::grid.arrange(gn1, gn2, gn3, gn4, gn5, gn6, gn7, gn8,
nrow=4, ncol=2)
 gridExtra::grid.arrange(gn9, gn10, gn11, gn12, gn13, gn14, gn15, gn16,
nrow =4, ncol=2)

 # # #Remove outliers using boxplot method
  df = train_data
 # train_data = df

 # # #loop to remove outliers from all variables
  for(i in cnames){
    print(i)
    val = train_data[,i][train_data[,i] %in%
boxplot.stats(train_data[,i])$out]
    print(length(val))
    train_data = train_data[which(!train_data[,i] %in% val),]
  }

# #Replace all outliers with NA and impute
 for(i in cnames){
   val = train_data[,i][train_data[,i] %in%
boxplot.stats(train_data[,i])$out]
   print(length(val))
   train_data[,i][train_data[,i] %in% val] = NA
 }
```

Imputing with with mean, median and kNN
#created a missing value at location train_data[1,9] with Actual 110 got #mean 100.57, median 101 and knn 98.49 hence I fixed median method to impute

```
 #Mean Method
# train_data$total.day.calls[is.na(train_data$total.day.calls)] =
mean(train_data$total.day.calls, na.rm = T)

 #Median Method
train_data$number.vmail.messages[is.na(train_data$number.vmail.mess
ages)] = median(train_data$number.vmail.messages, na.rm = T)
 train_data$total.eve.calls[is.na(train_data$total.eve.calls)] =
median(train_data$total.eve.calls, na.rm = T)

 # kNN Imputation
# train_data = knnImputation(train_data, k = 3)
```

#correlation plot

```
 corrgram(train_data[,numeric_index], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation
Plot")
```

Chi-squared Test of Independence

```
 factor_index = sapply(train_data,is.factor)
 factor_data = train_data[,factor_index]

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

Dimension Reduction

```
 train_data2 = subset(train_data,
                        select = -c(total.day.charge, total.eve.charge,
total.night.charge, total.intl.charge, phone.number))
```

#Normalisation

```
  cnames2 = c("account.length", "area.code", "number.vmail.messages",
"total.day.minutes", "total.day.calls", "total.eve.minutes",
"total.eve.calls", "total.night.minutes", "total.night.calls",
"total.intl.minutes", "total.intl.calls", "number.customer.service.calls")

 for(i in cnames2){
   print(i)
```

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

#model development

##Decision tree for classification

```
 DT_model = C5.0(Churn ~., train_data2, trials = 100, rules = TRUE)

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

test_data2 = subset(test_data, select = -c(4, 10, 13, 16, 19, 21))

#Let us predict for test cases
DT_Predictions = predict(DT_model, test_data2, type = "class")
```

##Evaluate the performance of classification model

```
ConfMatrix_DT = table(test_data$Churn, DT_Predictions)
confusionMatrix(ConfMatrix_DT)

#False Negative rate
FNR = FN/FN+TP

#Accuracy 93.5%
#FNR 47.7%
```

###Random Forest

```
RF_model = randomForest(Churn ~ ., train_data2, importance = TRUE,
ntree = 100)

#Predict test data using random forest model
RF_Predictions = predict(RF_model, test_data2)

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

#Accuracy 92.8%
#FNR 50.8%
```

#Logistic Regression

```
logit_model = glm(Churn ~ ., data = train_data2, family = "binomial")

#summary of the model
summary(logit_model)

#predict using logistic regression
```

```
logit_Predictions = predict(logit_model, newdata = test_data2, type =
"response")

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)


##Evaluate the performance of classification model
ConfMatrix_LR = table(test_data$Churn, logit_Predictions)

#Accuracy
sum(diag(ConfMatrix_LR))/nrow(test_data2)

#Accuracy 88.4%
#FNR = FN/FN+TP = 72.7%
```

##KNN Implementation

```
KNN_Predictions = knn(train_data2[, 1:15], test_data2[, 1:15],
train_data2$Churn, k = 3)

#Confusion matrix
Conf_matrix_knn = table(KNN_Predictions, test_data$Churn)

#Accuracy
sum(diag(Conf_matrix_knn))/nrow(test_data2)

#False Negative rate
FNR = FN/FN+TP

#Accuracy = 88.1
#FNR = 35.2
```

#naive Bayes model

```
NB_model = naiveBayes(Churn ~ ., data = train_data2)

#predict on test cases #raw
NB_Predictions = predict(NB_model, test_data2[,1:15], type = 'class')

#Look at confusion matrix
Conf_matrix_NB = table(observed = test_data[,21], predicted =
NB_Predictions)
confusionMatrix(Conf_matrix_NB)

#Accuracy: 88.66
#FNR: 76.33
```

#Fixed decision tree algorithm for this dataset due to their low FNR and high accuracy

```
 test_data = test_data[, 1:20]
df = data.frame(DT_Predictions)
output_R = cbind(test_data, df)
write.csv(output_R, "output_R.csv", row.names = F)
```

Python code

```
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency

os.chdir("D:/edwisor/Projects/Churn Reduction")

train = pd.read_csv("D:/edwisor/Projects/Churn
Reduction/train_data.csv")

test = pd.read_csv("D:/edwisor/Projects/Churn Reduction/test_data.csv")

# Assigning levels to data sets

#assigning levels to categorical varibales of train dataset

for i in range(0, train.shape[1]):

    if(train.iloc[:,i].dtypes == 'object'):

        train.iloc[:,i] = pd.Categorical(train.iloc[:,i])

        train.iloc[:,i] = train.iloc[:,i].cat.codes



#assigning levels to categorical varibales of test dataset

for i in range(0, test.shape[1]):

    if(test.iloc[:,i].dtypes == 'object'):

        test.iloc[:,i] = pd.Categorical(test.iloc[:,i])
```

```
test.iloc[:,i] = test.iloc[:,i].cat.codes

#storing target variable

train_target = train.Churn

test_target = test.Churn

combined = train.append(test)

# Checking data types of variables and converting

combined.dtypes
```


```
[{"metadata":{"trusted":true,"scrolled":true},"cell_type":"code","source":"
combined.dtypes","execution_count":9,"outputs":[{"output_type":"execute_res
ult","execution_count":9,"data":{"text/plain":"state
int8\naccount length                     int64\narea code
int64\nphone number                       int16\ninternational plan
int8\nvoice mail plan                     int8\nnumber vmail messages
int64\ntotal day minutes                float64\ntotal day calls
int64\ntotal day charge                 float64\ntotal eve minutes
float64\ntotal eve calls                    int64\ntotal eve charge
float64\ntotal night minutes              float64\ntotal night calls
int64\ntotal night charge               float64\ntotal intl minutes
float64\ntotal intl calls                   int64\ntotal intl charge
float64\nnumber customer service calls      int64\nChurn
int8\ndtype: object"},"metadata":{}}]}]
```


```
combined['area code'] = combined['area code'].astype('object')

# Checking relationship of target variable 'Churn' with 'State'

y = combined["Churn"].value_counts()

sns.barplot(y.index, y.values)

combined.groupby(["state", "Churn"]).size().unstack().plot(kind='bar',
stacked=True, figsize=(30,10))

## Missing Value Analysis

missing_val = pd.DataFrame(combined.isnull().sum())

missing_val
```

Future selection

```
#save numeric names
```

```
cnames =  ["account length", "number vmail messages", "total day minutes", "total day calls", "total day charge", "total eve minutes", "total eve calls", "total eve charge", "total night minutes", "total night calls", "total night charge",
        "total intl minutes", "total intl calls", "total intl charge", "number customer service calls"]

##Correlation analysis
#Correlation plot
df_corr = combined.loc[:,cnames]

#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(9, 7))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
        square=True, ax=ax)

#Chisquare test of independence
#Save categorical variables
cat_names = ["area code", "state", "phone number", "international plan", "voice mail plan"]

#loop for chi square values
for i in cat_names:
   print(i)
   chi2, p, dof, ex = chi2_contingency(pd.crosstab(combined['Churn'], combined[i]))
   print(p)
```

```
area code 0.7546581385329686 state 7.850836224371827e-05 phone number 0.7892627381002844 international plan 1.9443947474998577e-74 voice mail plan 7.164501780988496e-15
```

```
combined = combined.drop(['area code','total day charge', 'total eve charge', 'total night charge', 'total intl charge', 'phone number', 'Churn'], axis=1)
```

Future scaling

```
#Normality check
%matplotlib inline
plt.hist(train['total intl calls'], bins='auto')
```

```
#save numeric names
cnames_1 =  ["account length", "number vmail messages", "total day
minutes", "total day calls", "total eve minutes", "total eve calls", "total
night minutes", "total night calls",
         "total intl minutes", "total intl calls", "number customer service
calls"]

#Nomalisation
for i in cnames_1:
   print(i)
   combined[i] = (combined[i] - min(combined[i]))/(max(combined[i]) -
min(combined[i]))
```

Model development

```
#Import Libraries for decision tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
from sklearn import tree

train = combined[:3333]

test = combined[3333:]

#Decision Tree
C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(train,
train_target)

#predict new test cases
C50_Predictions = C50_model.predict(test)

#testing accuracy and to build confusion matrix
from sklearn.metrics import confusion_matrix
CM = confusion_matrix(test_target, C50_Predictions)

#check accuracy of model
accuracy_score(test_target, C50_Predictions)*100
#91.96

#False Negative rate
#(FN*100)/(FN+TP)

#Results
#Accuracy: 92.44#
#FNR: 30.35#
```

```
CM
array([[1385, 58], [ 68, 156]], dtype=int64)

#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF_model = RandomForestClassifier(n_estimators = 20).fit(train,
train_target)

RF_Predictions = RF_model.predict(test)
CM_RF = confusion_matrix(test_target, RF_Predictions)
accuracy_score(test_target, RF_Predictions)*100

#Accuracy 94.54%
#FNR 37.94%

CM_RF
array([[1437, 6], [ 85, 139]], dtype=int64)


#KNN implementation
from sklearn.neighbors import KNeighborsClassifier

KNN_model = KNeighborsClassifier(n_neighbors = 9).fit(train,
train_target)

#predict test cases
KNN_Predictions = KNN_model.predict(test)

#build confusion matrix
CM_kNN = confusion_matrix(test_target, KNN_Predictions)
accuracy_score(test_target, KNN_Predictions)*100
#CM = pd.crosstab(y_test, KNN_Predictions)

#False Negative rate
#(FN*100)/(FN+TP)

#Accuracy: 86.80%
#FNR: 95.98%

CM_kNN
array([[1438, 5], [ 215, 9]], dtype=int64)

#Naive Bayes
from sklearn.naive_bayes import GaussianNB

#Naive Bayes implementation
NB_model = GaussianNB().fit(train, train_target)
```

```
#predict test cases
NB_Predictions = NB_model.predict(test)

#build confusion matrix
CM_NB = confusion_matrix(test_target, NB_Predictions)
accuracy_score(test_target, NB_Predictions)*100

#False Negative rate
#(FN*100)/(FN+TP)

#Accuracy: 85.84
#FNR: 60.26

CM_NB
array([[1342, 101], [ 135, 89]], dtype=int64)

#I will fix Decision Tree model for this dataset because it is giving us low FNR and high accuracy.
Predict = pd.DataFrame(C50_Predictions)
Predict = Predict.rename(columns = {0:'Predictions'})
test = test.join(Predict['Predictions'])

test.to_csv("output.csv", index = False)
```