Customer Churn Rate Classification

# Anchal Sharma

# August 2018

**Contents**

**1 Introduction**

* 1.1 Problem Statement……………………………….….………. 4
* 1.2 Data………………………………………………..………… 6

**2 Methodology**

2.1  PreProcessing ………………………………………….……… 8

* 2.1.1 Histogram of Data ………………….……………………. 9
* 2.1.2 Box Plot Analysis………………………………………..…13
* 2.2 Missing Value Analysis……………………………………………….16
* 2.3 Feature Selection……………………………………………..………17
* 2.4 Feature Scaling……………………………………………………….19

**3 Modelling**

* 3.1  R……………………………………………..…………21
* 3.2 Python…………………………………………..……….23

**Appendix**

R Code for Boxplots…………………………………………..…………….25

R Code for Histograms……………………………………….…………… 26

An Excerpt of D-Tree………………………………………………………26

**References**………………………………………………………………..27

Customer Churn Rate

To call or not to call !!

Customer is the most valuable asset to most of the competing consumer based industries and will remain so in the future, since they are the ones who generate incomes and decide the future of the company. Granted, Customer are the pillars on which telecommunication sector stands its grounds. This makes treating customers right vital to the business.

With growing number of competitors in the communication industry, it has become ever more difficult and necessary to focus on churn rate and to convince the churning customer to reverse his/her decision. As for any industry, gaining a customer is always costly, if you take all the advertisement and campaigns cost combined. And when any industry loses a customer, the loss is inclusive of the gain, as they have to try different marketing strategy to retain the customers and that would further add to the costs. Hence, industry always wants to know in advance the customers who are likely to churn so that the industry could focus its resources on the churning customer and not on the ones who are there to stay.

We would develop machine learning algorithms to predict the churn rate in the report to follow.

**Chapter 1**

**Introduction**

**1.1 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 :**

Churn : if the customer has moved (1=False; 2 = True)

**1.2 Data**

Given below is the sample of the Train\_data.csv file. There are 16 continuous variables and 5 categorical variables for a total of 21 variables. The Train\_data.csv has 3333 observations whereas Test\_data.csv has 1667 observation, both have 21 variables.

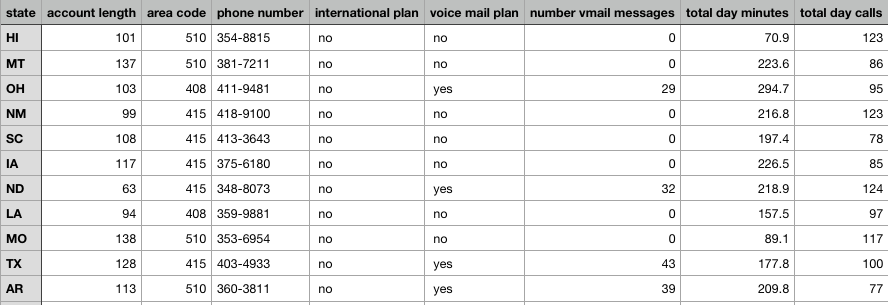
Table1.1 Columns(1-9)

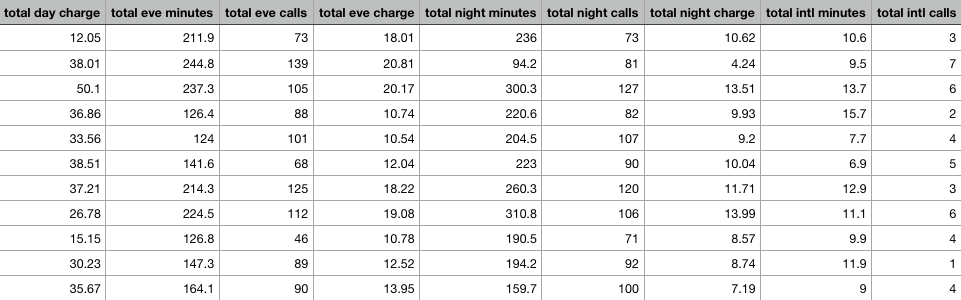
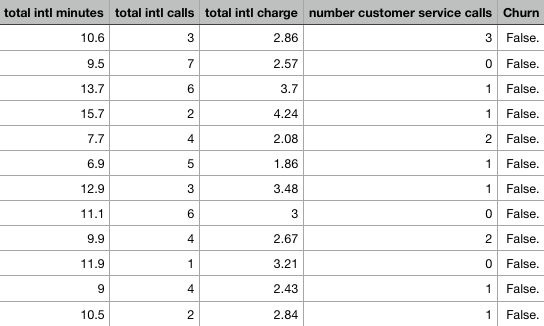
Table1.2 Columns(10-18)

Table1.3 Columns(17-21)

**Chapter 2**

**Methodology**

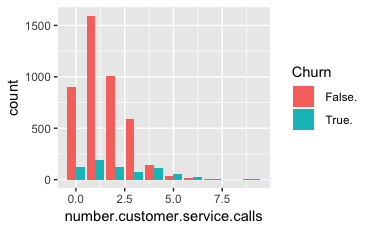
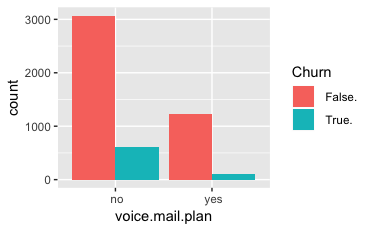
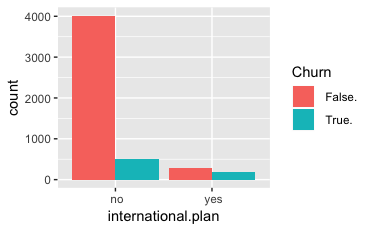
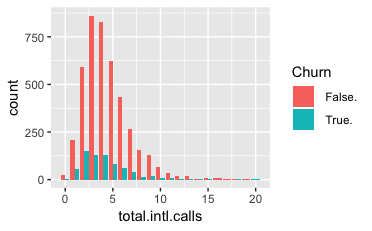
**2.1 Pre Processing**

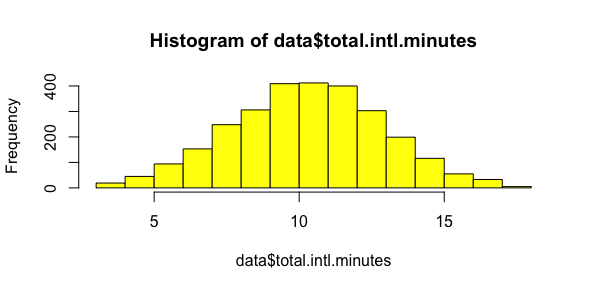
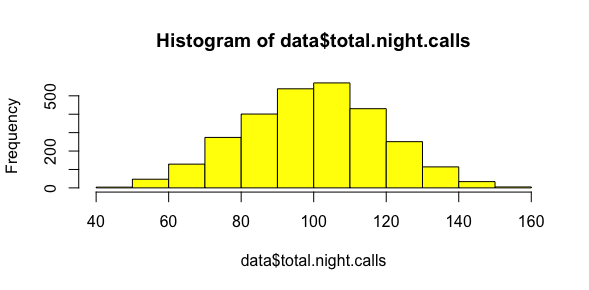
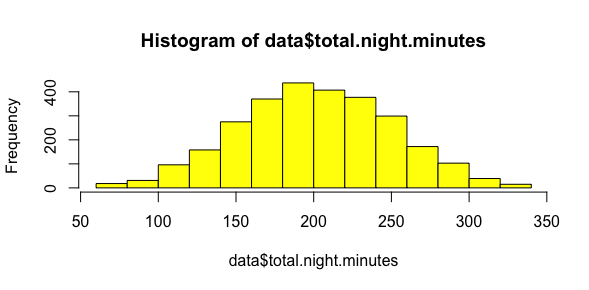
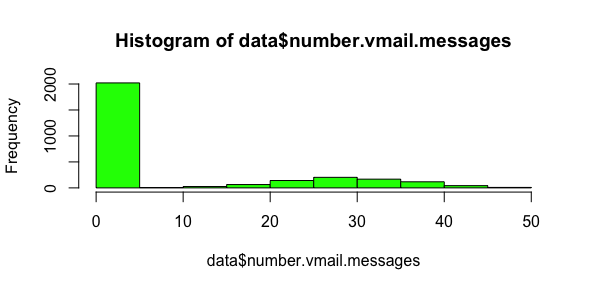
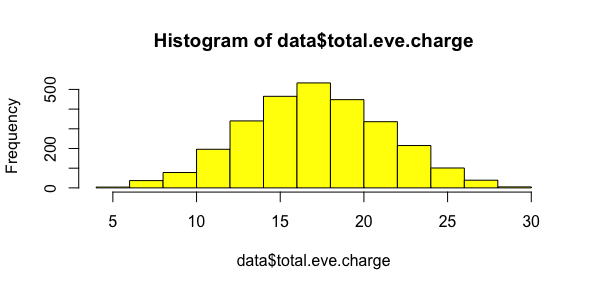
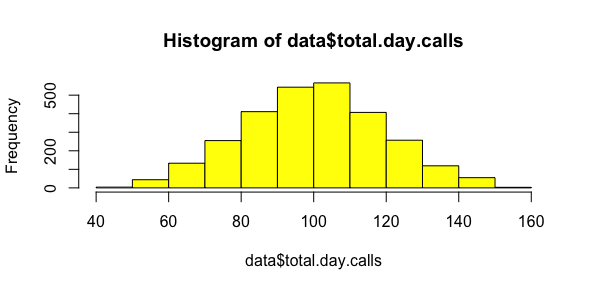
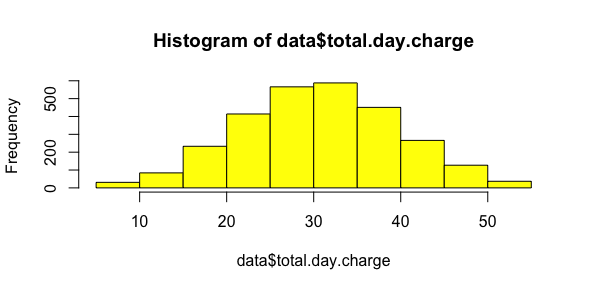
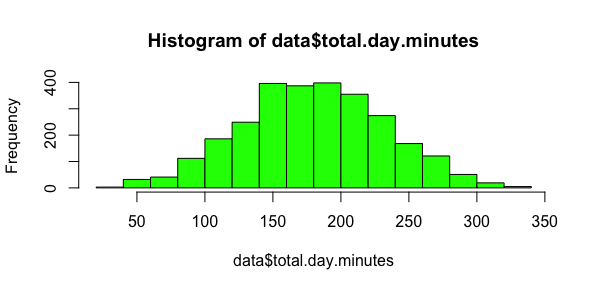
We require Pre-Processing as the data that we generally receive from the customer is noisy and not normalised. There could be a lot many missing values in the data, and accordingly we need to modify our dataset so that it could be fed into the required model , which in turn would increase the efficiency of the model in the long run, if it were to be deployed in the future.

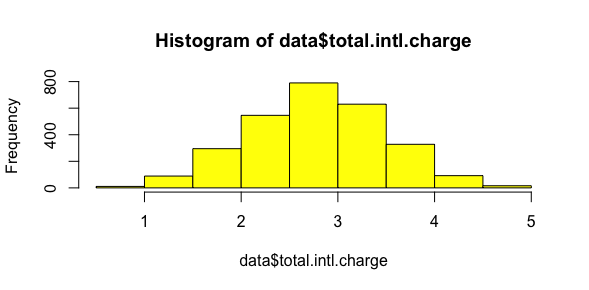
There are multiple steps that we need to follow while performing pre-processing such as:

* Data Cleaning
  + Filling in missing Value
  + Identifying outliers
  + Data transformation: Normalisation and standardisation
  + Data reduction

But before we dive in the above mentioned pre-processing techniques it is always better to visualise our data and get a rough idea of what we are dealing with. This process is also called Exploratory Data Analysis and is usually the first step in Data mining.



**Histogram of data**

****

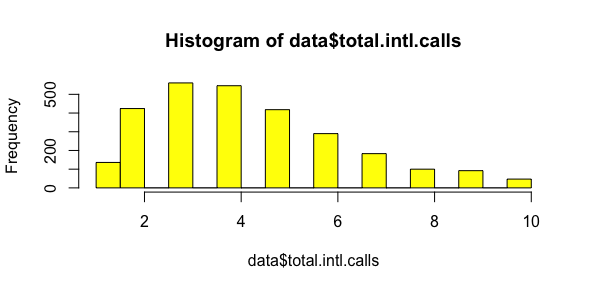
****

Figure 2.1 Histogram of Predictor Variables

As we can see from the above histogramsof continuous variables ,which also partially serve as predictor variables for out target variable, most of the data is normally distributed, or it looks like that. Most of the histograms are slightly left skewed (data$total.intl.calls) or slightly right skewed (data$total.day.charge) .

If we were to focus on the y-axis (frequency) we can see that it differs for most of the variables. for example the frequency of Total international calls fluctuates from 0-500 whereas that for the international charge fluctuates between 0-800 and so on. We now know that we need to normalise the data to bring all of them to the same scale so that input to our model would not lead to anomalies.

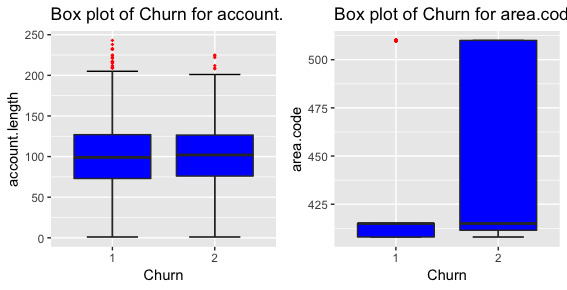
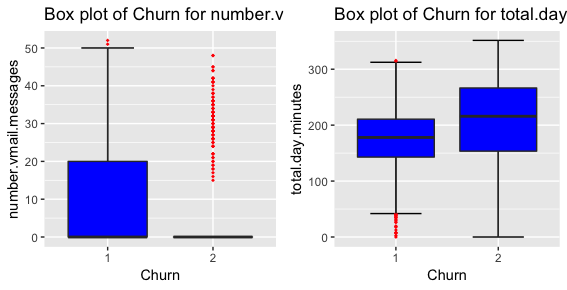
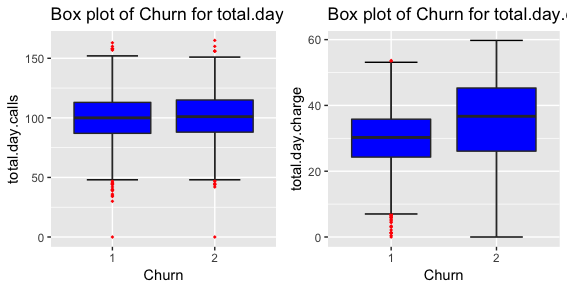
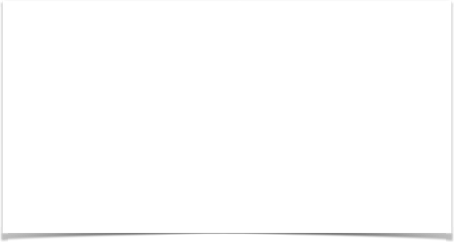
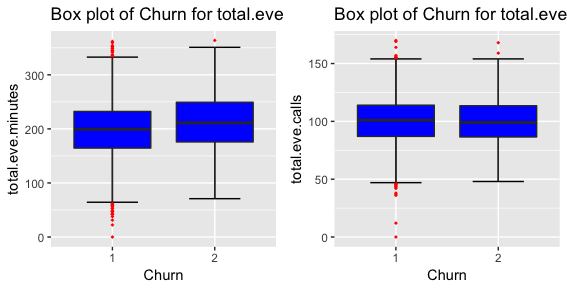
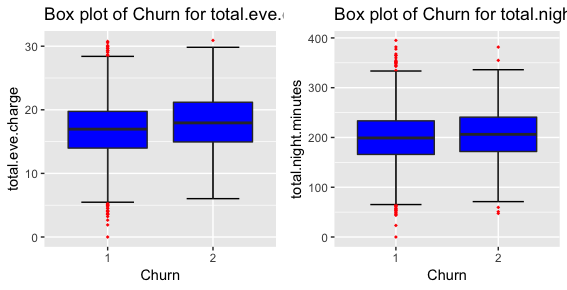
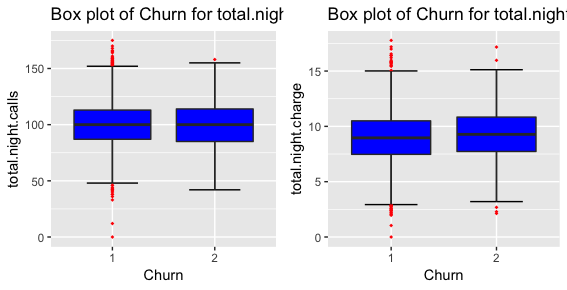
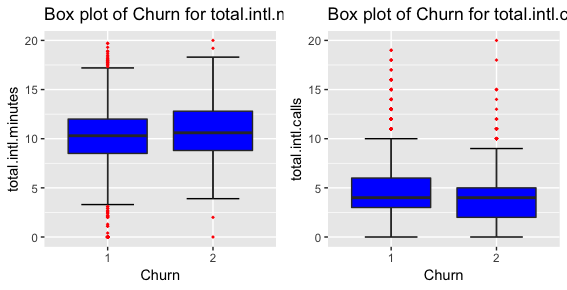
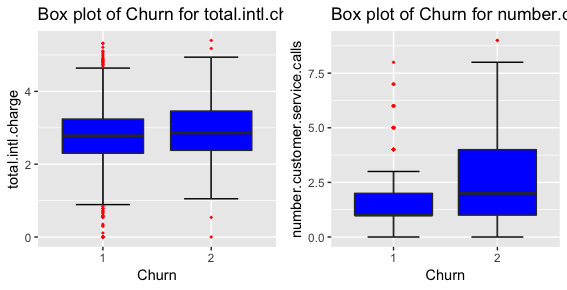
If we were to look at the categorical plots of the data, we can easily visualise that the Customers are less likely to churn if they do not have an international plans, as compared to those who have international plans. This sort of makes sense as well, Since international calls tend to be costly, customers are always on the lookout for cheaper alternatives and are more likely to Churn.

The proportion of customers who are likely to churn to those who won’t churn is greater for customers with a voice mails plan in comparison to the same proportion for those customers without a voice mail plan. Although the resemblance is uncanny, it would be pronounced if our dataset might have been scaled up.

Another insight that we get from the histogram plot is that those customers who have called customer service 4 or more times are more likely to churn than their counterparts. This again seems like a valid datapoint as a customer who is unhappy or not satisfied with the service of the telecom , would call customer care more often.

Now, there are a few variables in our dataset that need to be dropped beforehand as they do not add much value to our analysis, for example, information such as phone number won’t add much value, as most of the phone numbers are distinct and they just add noise to our clean data. Other variables such as area code, account length, and state codes would not add much value to our data and hence it is better to drop them out of before model phase begins.

For our ease, I have merged Train and Test data together in a dataset and then I have performed outlier analysis, normalisation and standardisation on the same. Post which, before feeding the dataset to the model, I did take care to further split our dataset into train and test, so that our ultimate goal remains unaltered.



Box Plot for variables of dataset.(See R Code in appendix)

As we can analyse by looking at the box plots of various variables in our dataset, There are many outliers present. Specifically the histogram plot and book plot for variables number.vmail.messages, total.international.calls, and number.customer.service.calls show major variations in the data. We could fix them by removing the respective outliers and thereby fixing their normality. But as we are having limited data for now (in total 5000 observations) it is better that we keep them to train our Machine Learning Algorithm. If instead, our dataset would have been much larger(~10K) we would have preferred to remove the outliers. But with the smaller dataset in hand , it would be better that we continue with Normalisation on the given data instead of removing the outliers.

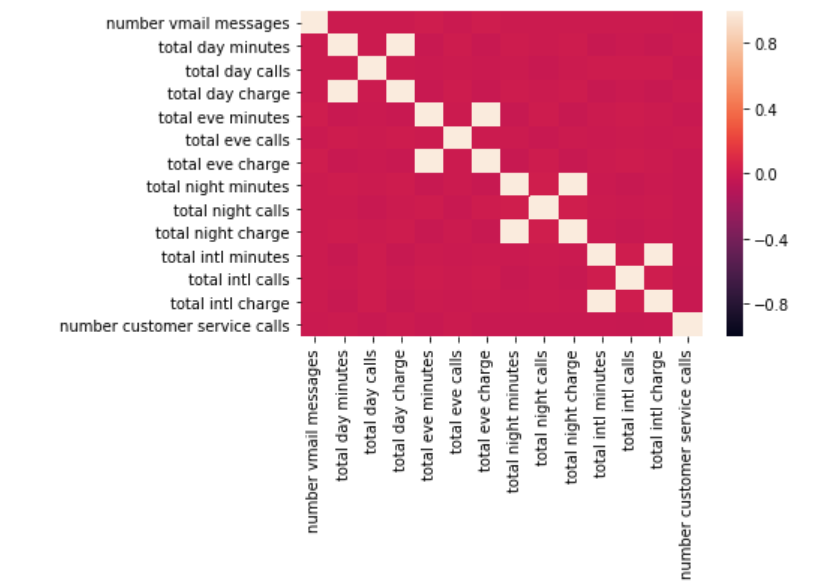
2.2 Missing Value Analysis

**tot=data.frame(sapply(data,function(x){sum(is.na(x))}))**

This line of code in R will return the missing values(if any) in the dataset. In here we have used the apply function which encapsulates data frame and number 2 as we are performing column level operation, and a function body, which returns the sum of the ‘na’ , if any, in the dataset. The default method for is.na applied to an atomic vector returns a logical vector of the same length as its argument x, containing TRUE for those elements marked NA or, for numeric or complex vectors, [NaN](https://www.rdocumentation.org/link/NaN?package=base&version=3.5.1), and FALSE otherwise. After analysing the data, it is certain that the dataset doesn’t contain any missing values. And hence we don’t need to do missing value imputation further.

2.3 Feature Selection

Once we receive the dataset from the client and after defining the problem category and forming the problem statement, we perform the above mentioned steps to get some sense of the data, but all that sense is intuitional and exploratory. We need to use Feature Selection in order to explore further trends and insights into out continuous and categorical variables so as to see the hidden dependencies between different such variables. As such in our dataset, we have performed two feature selection techniques: One, for the categorical variables and another one for the continuous variables.

For **Continuous variables**, we have employed Correlation Plot method. This graphical method will plot a correlation plot on the dataset in which bright colours represent high dependencies between two variables.

As we could see from the correlation plot of our continuous variables, most of the data is independent of each other and the only ones which are showing dependencies are the ones who are kind of similar in a manner. For example we have high collinearity between total day minutes and total day charge. It makes sense as well since the amount of time you talk on the phone will be proportional to the amount you will be charged by the telecom operator. and so on. It is better , due to limited number of variables in our dataset, that we consider all the mentioned columns , as they add further information to the dataset which would be required for algorithm to make decisions.

For Categorical variable, we need to follow a different approach: **Chi-Square Test of Independence**

In Chi-square test of independence, we need to make two hypothesis: Null and alternated.

Out Null Hypothesis holds true if the p-value of the data is less than 0.05, and alternate hypothesis is true is p-value is greater than 0.05.

Here our null hypothesis says that the there is high dependency between Independent variable and dependent variable and low dependency between two independent variable. Vice-Versa is true for alternate hypothesis

Once we have analysed and predicted the p-value for the respective categorical variables, we have found that the p-value for **phone number** column is greater than 0.05 and hence we have to follow the alternate hypothesis and remove the phone number variable. It also makes sense that phone number of a customer would not hold much value for us to decide whether the customer is going to Churn or not.

2.4 Feature Scaling

As we can see from the histogram plots of all the predictor variables, the frequency as well as scale of different variables vary. For example if we take the total international calls variable, the unit of which is not defined in the SI unit scale, whereas if we were to take total international minutes, which has unit of minutes in the end. As we have to deal with different variables , almost all of which have separate units, its better that we normalise the data such that the range of all the variables will be the same (0-1). For such scale would lead to better interpretation of the data by out machine learning algorithm and would result in less anomalies. This process comes under Feature Scaling and is preferred most of the time if the dataset is not normalised. However, if you have a dataset that is already normalised then its better to go for Standardisation(**Z-Score**), a process which, instead of normalising the range from 0-1, focuses on the standard deviation of the particular value from mean of its comprising variable. Mostly the data that we receive from client is not normalised and needs lot many pre-processing steps , hence in those cases, standardisation is a no-no.

After reducing our variables to the defined range, our data is now ready to be fed to the machine learning algorithm. Lets get started.

**Chapter 3**

**Modelling**

Once we are done with above mentioned steps, we need a model to feed out data and get the test result ,all with better accuracy. A good model is the one which can predict from the test data accurate predictions. In other words we need a model with higher accuracy and another factor that client mostly takes into consideration is False positive rate. Now, We have separate models where target variable is categorical , separate ones for those with target variable as continuous. The model ,which predicts continuous variables, generally comes under regression models, whereas the models which predict classes be it ordinal, nominal or binomial, are known as classification models. The simplest of most models is Linear regression ( for continuous variables) and Logistic Regression (for categorical variables). Though the data set that we have in hand doesn’t comply with one of the assumptions of Logistic regression, i.e , the target variable need to be balanced. By balanced it meant to say that for example, if we have 100 observations, then 50 have to be Yes and ~50 have to be false. Hence we are not employing Logistic Regression right now in this dataset.

The following models have been taken into consideration while coding for the dataset in R and Python, and their calculated accuracies are mentioned as well. Please refer to the Code Files for R and Python for reference.

**R:**

**Decision Trees:**

**#Accuracy** 96.4% **#False Negative Rate**: 23.66%

C50\_model = C5.0(Churn~., train, trials = 100, rules = TRUE)

Here we have employed C5.0 model, which takes Information Gain as the factor to split up the nodes and categorise the data. Trials are put to 100 and rules=TRUE, which provides us with all the rules that are employed by the model.

The accuracy and False Negative rate of this model is most optimum of that of every other model employed for this dataset, However that could be because the tree is overfitted, and is very sensitive to even a minute change in data values in dataset. This is one disadvantage of using a D-Tree. One needs to find a perfect balance in overfitting and under-fitting, and that is time-consuming.

**KNN Implementation:**

**#Accuracy** 90.16 % **#False Negative Rate**: 15.11%

KNN\_Predictions = knn(train[, 1:17], test[, 1:17], train$Churn, k = 7)

K-Nearest-Neigbour, also known as lazy learning method, imputes the predicted value using distance based algorithms (Eucledian, Manhattan, etc.) In case of classification, the algorithm calculates the majority-minority rule and then predict the value of the target variable. Whereas in case of continuous variable, it calculates the mean and median of the nearest neighbours and then predict the target variable. Here, We have calculated accuracy and FNR(False Negative Rate) for 1,3,5 and 7 nearest neighbours, but the accuracy remained constant once k=7, hence if we were to increase the neighbours it would not lead to high increase in accuracy , hence 7 is the optimal number for our dataset.

**Random Forest:**

**#Accuracy** 96.16% **#False Negative Rate**: 26.33%

RF\_model = randomForest(Churn ~ ., train, importance = TRUE, ntree = 500)

Random Forests are an extension to D-Trees in that the random forests comprises of multiple D-trees. The problem of overfitting is taken care of in Random forest technique. The method randomForest( ) grows trees until the error rate for two or more trees are same. If the error rate doesn’t increase or decrease the method will stop growing more trees. In our code, given above, we have provided the number of trees to be grown in the forest. Random Forest is the ensemble method of machine learning algorithm.

Hence the best model in R for our dataset would be KNN Implementation as although the accuracy using this algorithm is not the best, The False Negative rate is lowest and that is one metric that clients mostly prefer.

**Python:**

**Decision Tree:**

**#Accuracy** 92.00% **#False Negative Rate**: 29.00%

C50\_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X\_train, y\_train)

Looking at the accuracy and FNR of the decision tree model overall it seems like the model is a good fit, although we need to test the fit of our dataset for other models as well.

**Random Forest:**

**#Accuracy** 96.00% **#False Negative Rate**: 26.00%

RF\_model = RandomForestClassifier(n\_estimators = 500).fit(X\_train, y\_train)

The accuracy of the model has increased by 4% and consecutive decrease in the FNR of about 3% also qualifies this model as the perfect fir for our dataset.

**KNN Implementation:**

**#Accuracy** 91.00 % **#False Negative Rate**: 56.00%

KNN\_model = KNeighborsClassifier(n\_neighbors = 7).fit(X\_train, y\_train)

The Accuracy of the algorithm is less than both random forests and Decision tree , plus it leads to consecutive increase in the False Negative rate as well. Hence, until now we could see that Random forest has been the best fit overall.

**Naive Bayes:**

**#Accuracy** 87.00% **#False Negative Rate**: 47.00%

NB\_model = GaussianNB( ).fit(X\_train, y\_train)

Naive Bayes model calculates the posterior conditional probability of of a categorical variable with respect to the certain class of target variable. It also includes the Prior Probability of the class and computes and relies solely on the probability function of the same.

The accuracy of Naive Bayes is the least of all the above models although False Negative Rate of the model is less than that of KNN Implementation. FNR is one of the most important metrics which is taken into consideration by clients. Hence, we need to find the model where FNR is low and accuracy is high.

Hence, we could come to the conclusion that of the above 4 models employed by us, the best one would be to use the Random Forest Algorithm as the accuracy is high: 96% as well as the False negative rate is also very low (26%), which is the added advantage.

**Appendix**

**R code for box plot:**

**numeric\_index = sapply(data,is.numeric)**

**numeric\_data = 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(data))+**

**stat\_boxplot(geom = "errorbar", width = 0.5) +**

**geom\_boxplot(outlier.colour="red", fill = "blue" ,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,ncol=2)**

**gridExtra::grid.arrange(gn3,gn4,ncol=2)**

**gridExtra::grid.arrange(gn5,gn6,ncol=2)**

**gridExtra::grid.arrange(gn7,gn8,ncol=2)**

**gridExtra::grid.arrange(gn9,gn10,ncol=2)**

**gridExtra::grid.arrange(gn11,gn12,ncol=2)**

**gridExtra::grid.arrange(gn13,gn14,ncol=2) gridExtra::grid.arrange(gn15,gn16,ncol=2)**

**R Code for plotting the histograms with ‘Churn’ as the filling element:**

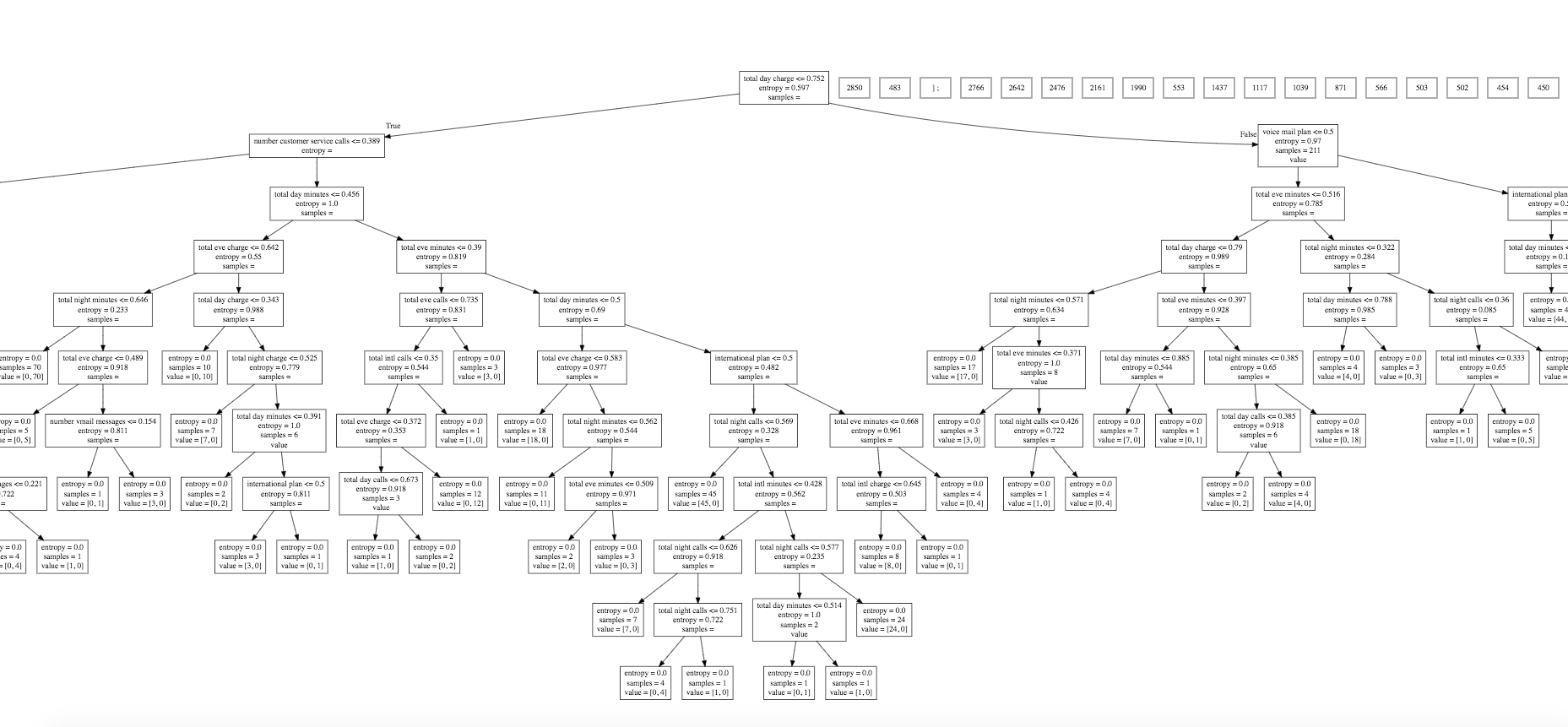
**library(ggplot2)**

**ggplot(data, aes(x = ‘Variable Name’, fill =Churn)) +**

**geom\_bar(position = “dodge")**

**R code to plot histograms with Continuous Variables:**

**hist(data$column name, col=‘yellow')**

**Excerpt of D-Tree:**

Type to enter text

**References**

* **[www.edwisor.com](http://www.edwisor.com)**
* **[www.datacamp.com](http://www.datacamp.com)**
* **[www.kaggle.com](http://www.kaggle.com)**