Churn Reduction

Hitesh Juneja

2 December 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

Churn means loss of customer to competition. Churn is problem of most of the companies because it is more difficult to acquire a new customer than to keep you existing one for leaving. The objective of this project is to predict the customer behaviour .We would like to predict that whether the customer will keep using the companies service or he will move out.

1.2 Data

Our task is to build the classification model which will predict whether the customer will move out or not based on multiple predictors provided in the problem statement. Given below is the sample data which we will be using for predicting the customer behaviour.

Table 1.1:Churn Reduction Sample Data (Columns: 1-10)

State	Account Length	Area Code	Phone Number	International Plan	Voice Mail Plan	Number Vmail Messages	Total Day Minutes	Total Day Calls	Total Day Charge
KS	128	415	382-4657	no	yes	25	265.1	110	45.07
OH	107	415	371-7191	no	yes	26	161.6	123	27.47
NJ	137	415	358-1921	no	no	0	243.4	114	41.38
OH	84	408	375-9999	yes	no	0	299.4	71	50.9
OK	75	415	330-6626	yes	no	0	166.7	113	28.34

Table 1.2:Churn Reduction Sample Data (Columns: 11-20)

Total Eve Minutes	Tota l 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
197.4	99	16.78	244.7	91	11.01	10	3	2.7	1

195.5	103	16.62	254.4	103	11.45	13.7	3	3.7	1
121.2	110	10.3	162.6	104	7.32	12.2	5	3.29	0
61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2
148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3

Table 1.3:Churn Reduction Sample Data (Columns: 21)

Churn
False.
False.
False.
False.
False.

As we can see in the below table that we have 20 variables, using which we have to predict whether the customer will move or not:

Table 1.4 : Predictor Variables.

S.No.	**Predictor**	**S.No.**	**Predictor**
1	State	11	Total Eve Minutes
2	Account Length	12	Total Eve Calls
3	Area Code	13	Total Eve Charge
4	Phone Number	14	Total Night Minutes
5	International Plan	15	Total Night Calls
6	Voice Mail Plan	16	Total Night Charge
7	Number Vmail Messages	17	Total Intl Minutes
8	Total Day Minutes	18	Total Intl Calls
9	Total Day Calls	19	Total Intl Charge
10	Total Day Charge	20	Number Customer Service Calls

Chapter 2

Methodology

2.1 Pre Processing

We should always look at the data before preparing the model. However, in data mining terms looking at data refers to much more than just looking. Looking at the data means exploring the data, cleaning the data as well as visualizing the data through graphs and plot. This is often called as **Exploratory Data Analysis**.

According to Problem statement provided, we can see that the state, phone number and area code are not defined as the predictive variable hence we can remove these variables. Also, we can provide labels to the categorical variables.

2.1.1 Missing Value Analysis.

Before preceding, we must check our data set for the missing values. If there are missing values available in our data set then we must impute those values for getting better result. According to industry standards, if the data set have less than 30 % missing values than we must impute values. There are different techniques which can be used to impute missing values in data set.
Below you can see the missing values analysis of the provided data set.

```
#Missing values Analysis
missing_val=data.frame(apply(Data_Frame_train,2,function(x){sum(is.na(x))}))
```

	apply.Data_Frame_train..2..function.x...
account.length	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

As you can see above that there are no missing values present in the data set. Hence we can proceed to the next step of Exploratory Data Analysis.

2.1.2 Feature Engineering

Feature Engineering is one of the preprocessing technique. Feature Engineering is process of using domain knowledge of data to create features that make machine learning algorithm works. If feature Engineering is done correctly, then it can increase the predictive power of machine learning algorithms by creating new features from the raw data that help facilitate the machine learning process.

We can do feature engineering in the provided data set, as we can see that we have column like ‘total day minutes’, ‘total eve minutes’, ‘total night minutes’, ‘total intl minutes’ by using these features we can create a new feature, let’s call this feature **total_minutes**.

Also, we can create one more feature ‘**total_calls**’ by using feature like ‘total day calls’ , ‘ total eve calls’, ‘total night calls’, ‘total intl calls’.

Now, we have two new features in our data set which can help us in predictive analysis. These two feature can also be used to create one feature '**Avg_time**'. So, by using the raw data set we have created three new features which can be helpful in predictive analysis.

Details of new features:

total_minutes = This feature tell us about the total time spent by the customer on call in whole day. This feature store the sum of feature 'total day minutes', 'total eve minutes', 'total night minutes', 'total intl minutes'.

total_call = This feature tell us about the total number of calls made by the customer in a whole day. This feature stores the sum of feature 'total day calls' , ' total eve calls', 'total night calls', 'total intl calls'.

Avg_time = This feature tell us about the average time spent by the customer on each call.

Below is the line code used for the creation of new features.

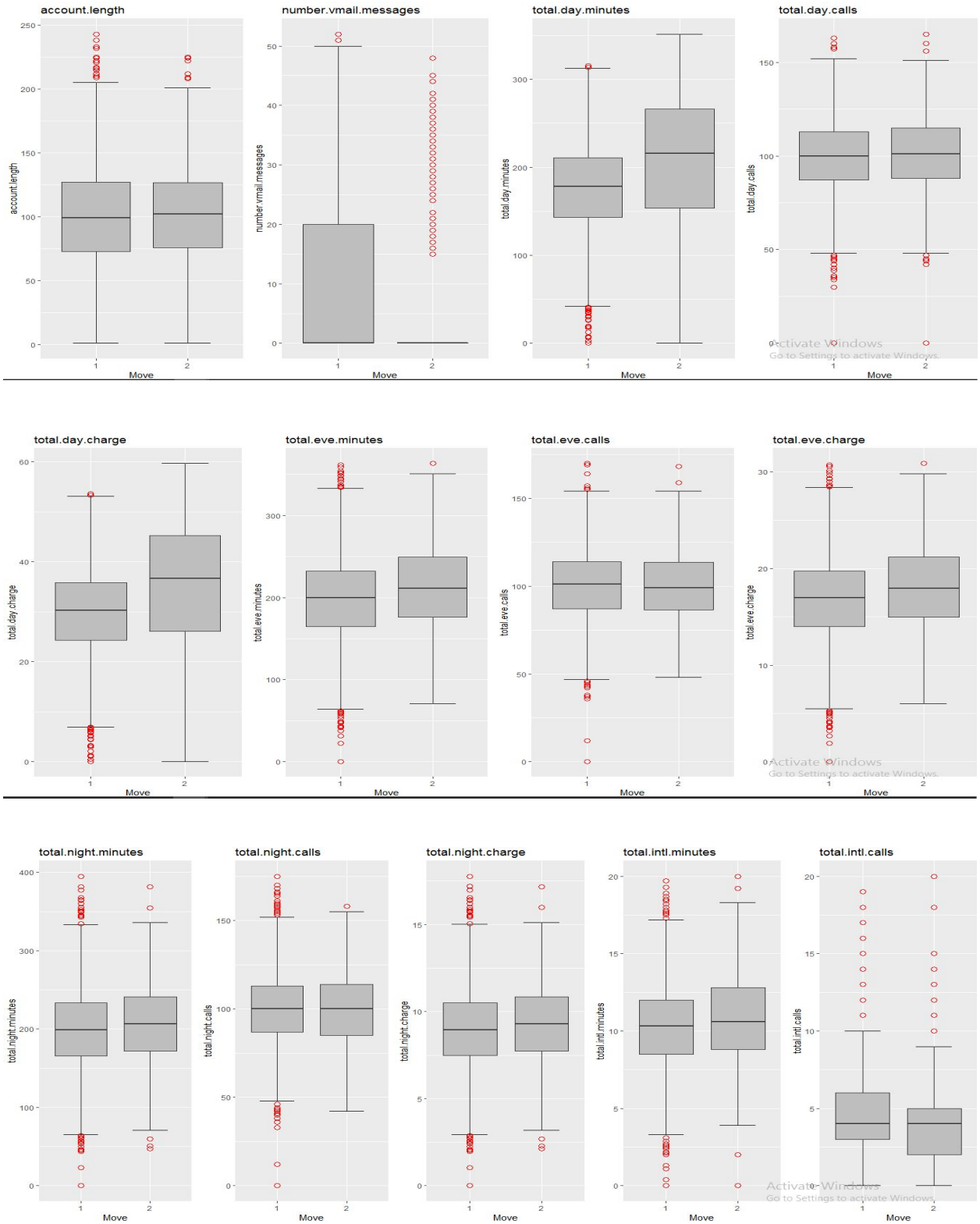
```
Data_Frame_train$total_minutes <- Data_Frame_train$total.day.minutes + Data_Frame_train$total.eve.minutes +
                                  Data_Frame_train$total.night.minutes+Data_Frame_train$total.intl.minutes
Data_Frame_train$total_calls <- Data_Frame_train$total.day.calls+Data_Frame_train$total.eve.calls+
                                  Data_Frame_train$total.night.calls+Data_Frame_train$total.intl.calls
Data_Frame_train$Avg_time <- Data_Frame_train$total_minutes/Data_Frame_train$total_calls
```

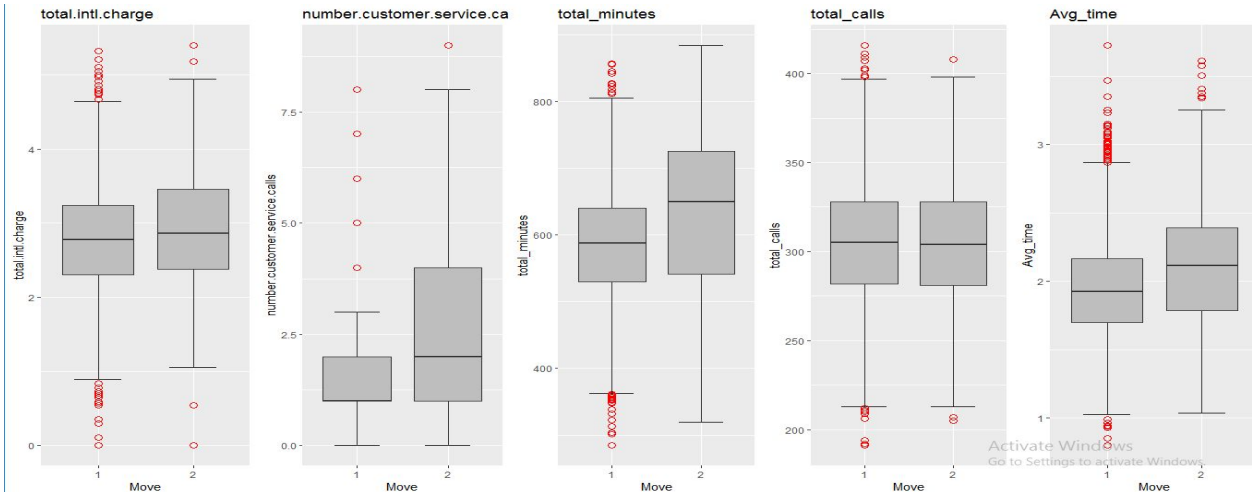
2.1.3 Outlier Analysis

One of the other step of pre-processing technique apart from missing value analysis and feature engineering is the presence of outliers. In this case we can use one of the classic approach of removing outliers, Tukey's method. We can visualize the outliers using boxplots. We can only plot the box plot of numeric features with the target variable. As in our data set we have 18 features of type numeric. Hence we will only plot the box plot of only those feature.

In Figure we have plotted the box plots of 18 predictor variables with respect to each Churn value 1 (False) and 2 (True). 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.

Fig 2.1 : BoxPlot of Numeric features with Outliers

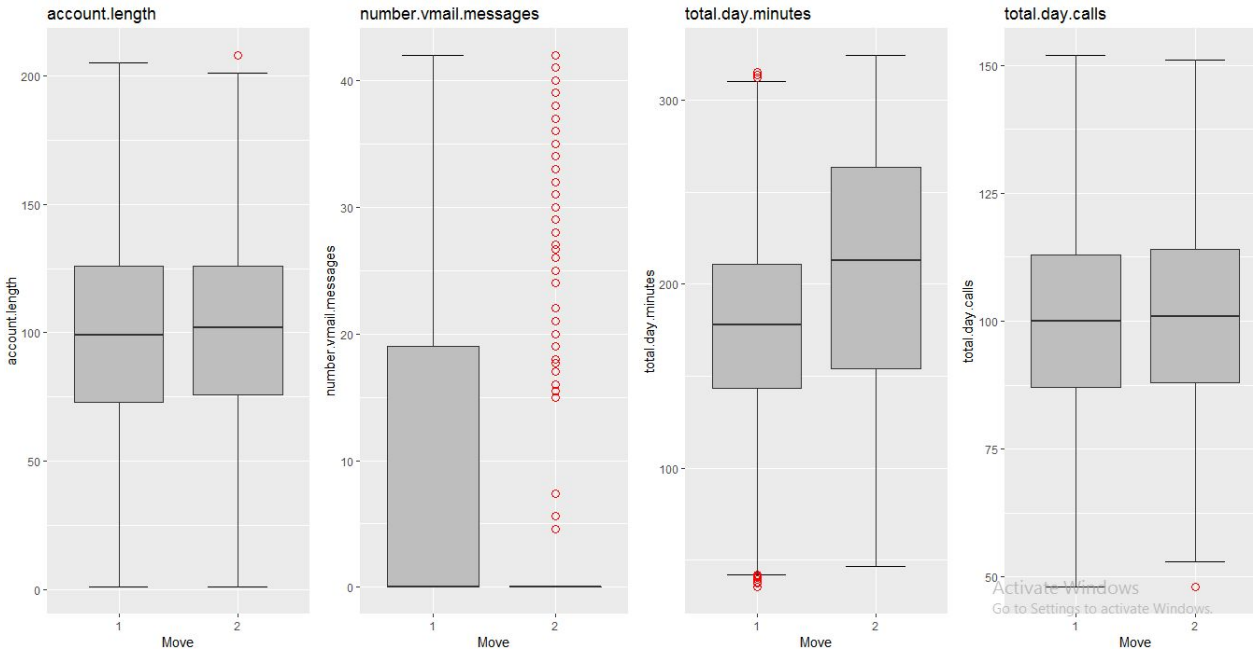


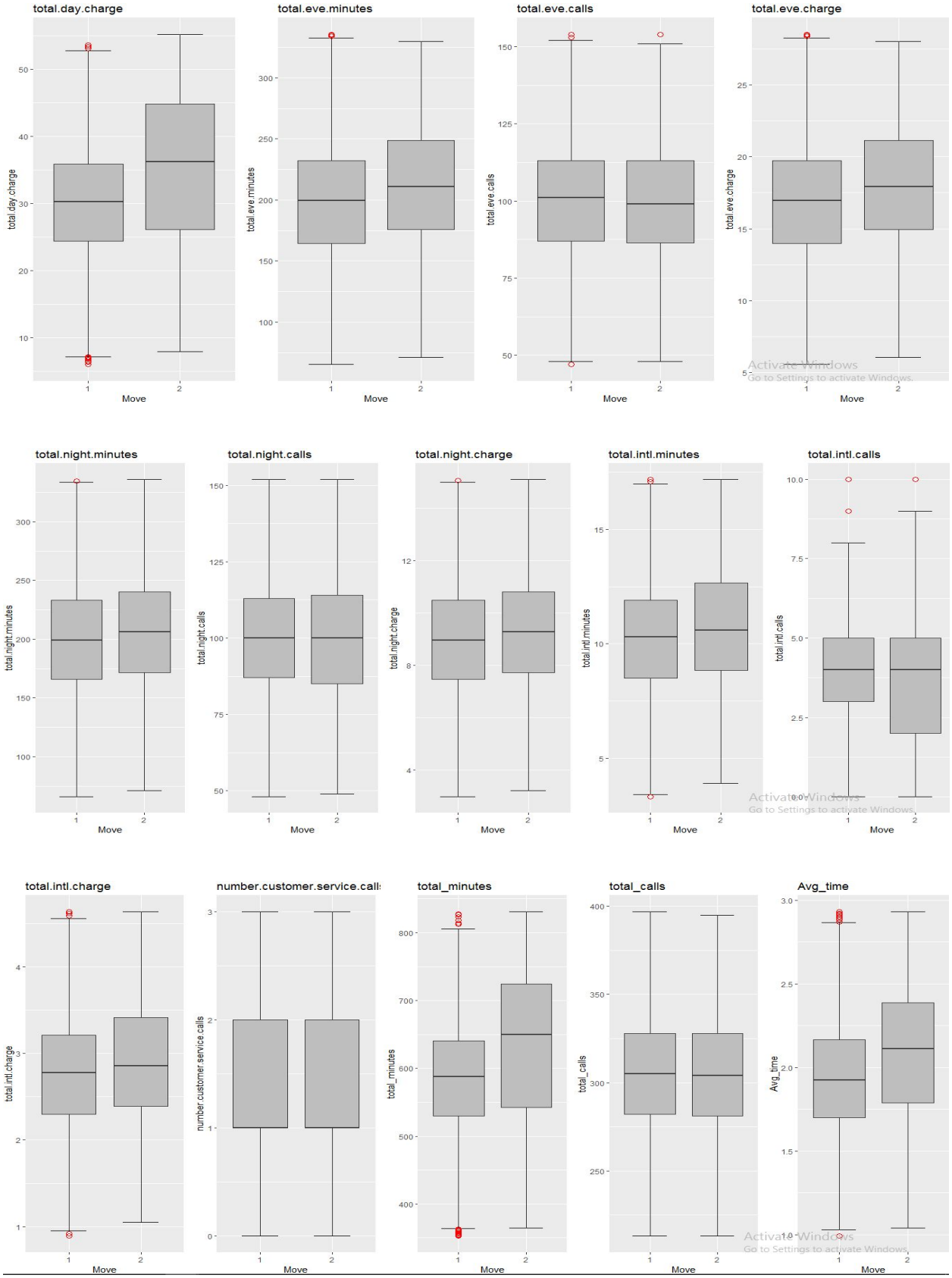


As we can see in the above image that there are lot of outliers present in the numeric data. We have to remove these outliers to bring our data into proper shape.

After performing the outlier analysis and removing outliers using Tukey's Boxplot method. We plot Predictor Vs Quality Box Plot Again

Fig 2.2 : Box Plot of numeric feature after removing Outliers.

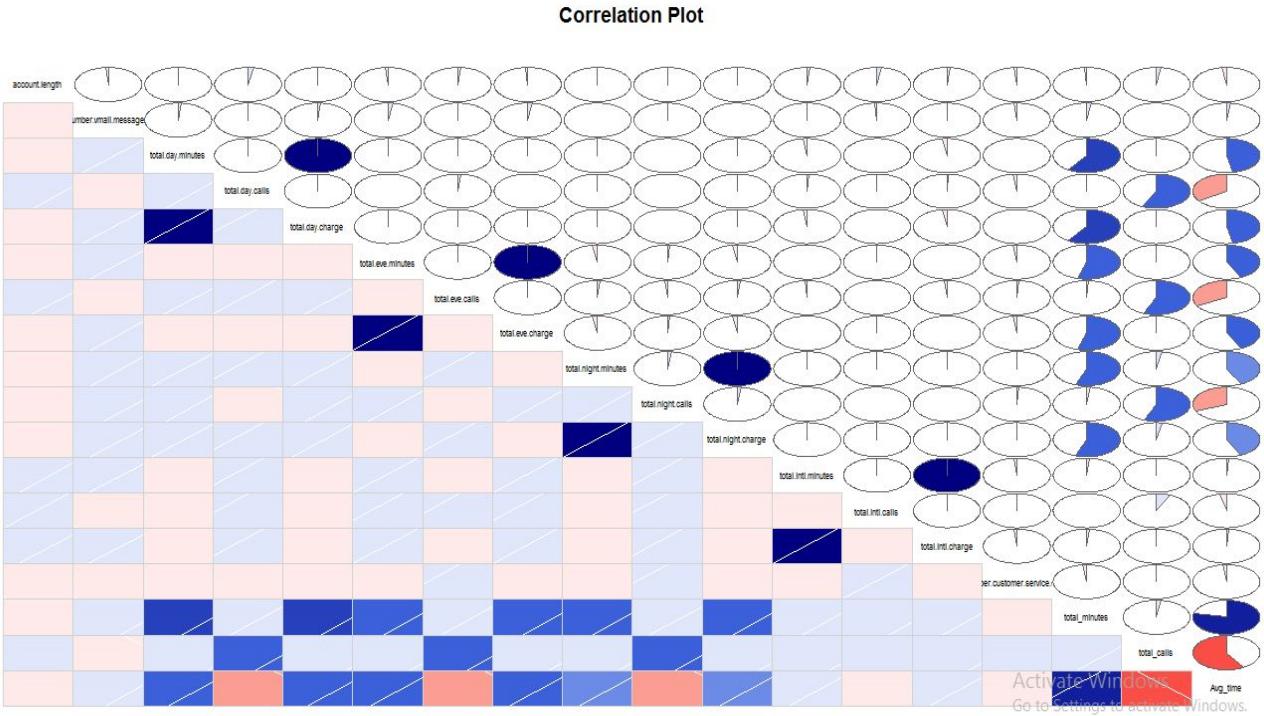



total.day.charge
total.day.charge
Move
total.eve.minutes
total.eve.minutes
Move
total.eve.calls
total.eve.calls
Move
total.eve.charge
total.eve.charge
Move
total.night.minutes
total.night.minutes
Move
total.night.calls
total.night.calls
Move
total.night.charge
total.night.charge
Move
total.intl.minutes
total.intl.minutes
Move
total.intl.calls
total.intl.calls
Move
total.intl.charge
total.intl.charge
Move
number.customer.service.calls
number.customer.service.calls
Move
total_minutes
total_minutes
Move
total_calls
total_calls
Move
Avg_time
Avg_time
Move

2.1.4 Feature Selection

Before performing any type of modeling we need to check the importance of each predictor variable present in our analysis. There is a possibility that many variables are not at all important to the problem of class prediction. The process of selecting only important features from our data set is called Feature Selection. There are several method for selecting feature from the data set. Below we have use Correlation method for Numeric Feature Selection and Chi-Square test for Categorical variable for Feature Selection.

Fig 2.3 : Correlation Plot for Feature Selection



As we can see in above correlation plot that ‘total day minutes’, ‘total eve minutes’, ‘total night minutes’ and ‘total intl minutes’ predictors are highly correlated with the ‘total day charge’, ‘total eve charge’, ‘total night charge’ and ‘total intl charge’ respectively. So, we can either remove all the minutes predictors or we can remove all the charge feature from the data set.

Chi-Square Test for Feature Selection

Code :

```
## ---------------- Chi -square test for factor variable reduction
factor_index=sapply(Data_Frame_train,is.factor)
factor_data=Data_Frame_train[,factor_index]
colnames(factor_data)
factor_data_length = length(colnames(factor_data))

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

Output :

```
[1] "international.plan"

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

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

After, performing Chi-square test we can see that all the categorical variables have p-value less than 0.05, that the variables are completely independent.

After performing Correlation Analysis and Chi-square test we can remove ‘total day charge’, ‘total eve charge’, ‘total night charge’, ‘total intl charge’ because these variables are dependent variables.

2.2 Modeling

2.2.1 Model Selection

The dependent variable can fall in either of the categories:

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

If the dependent variable, in our case Move, is Interval or Ratio then the normal method is to do a Regression Analysis, or Classification after binning. If the dependent variable is Ordinal, then both regression and classification can be done. But in our case dependent variable is Nominal the only predictive analysis that we can perform is Classification.

There are lot of machine learning algorithm available which can be used to solve the classification problem. Few of them are Decision Tree, Random Forest, Logistic Regression, Naive Bayes etc. We can use any machine learning algorithm to predict the values of target variable. We will use multiple algorithm to find the values of target variable. And, then after calculating the values of target variable, we will select only that model which have the highest accuracy,

2.2.2 Decision Tree

It is predictive model based on the branching series of boolean test. Decision Tree is basically used to create the predictive model which can be used to find the class or label value by using a decision rules which are collected from the past. Output of the decision tree is simple business rules. There are different algorithm present in the Decision Tree like C5.0, CART etc.

The output of the decision tree is alway in form of rules, these rules are then applied on the test data to predict the values of target variables.

```
> C50_model=C5.0(Move~.,train,trials=100,rules=TRUE)
> C50_predications=predict(C50_model,test[,-17],type="class")
> #Evaluate the performance of classfication model
> confMat_C50=table(test$Move,C50_predications)
> confusionMatrix(confMat_C50)
Confusion Matrix and Statistics

   C50_predications
       1    2
  1 1432   11
  2   73  151

               Accuracy : 0.9496
                 95% CI : (0.938, 0.9596)
    No Information Rate : 0.9028
    P-Value [Acc > NIR] : 1.449e-12

                  Kappa : 0.7547
 Mcnemar's Test P-Value : 2.821e-11

            Sensitivity : 0.9515
            Specificity : 0.9321
         Pos Pred Value : 0.9924
         Neg Pred Value : 0.6741
             Prevalence : 0.9028
         Detection Rate : 0.8590
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9418

       'Positive' Class : 1
```

As we can see above that C5.0 method of Decision Tree algorithm is giving us the accuracy of 94.96%. The accuracy we get after applying decision tree model on the test data is really impressive, but we cannot rely only on this model. We must use some other models to predict values of target variable of our data set and then come to conclusion.

2.2.3 Random Forest

Random Forest is an ensemble that consists of many decision trees. Random forest is basically a combination of weak learn to produce the strong learning model.

```
> RF_model= randomForest(Move~.,train, importance = TRUE, ntree=700)
> RF_Predictions = predict(RF_model, test[,-17])
> confMatrix_RF = table(test$Move, RF_Predictions)
> confusionMatrix(confMatrix_RF)
Confusion Matrix and Statistics

   RF_Predictions
      1    2
  1 1440    3
  2   87  137

               Accuracy : 0.946
                 95% CI : (0.9341, 0.9564)
    No Information Rate : 0.916
    P-Value [Acc > NIR] : 1.774e-06

                  Kappa : 0.7242
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9430
            Specificity : 0.9786
         Pos Pred Value : 0.9979
         Neg Pred Value : 0.6116
             Prevalence : 0.9160
         Detection Rate : 0.8638
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9608

       'Positive' Class : 1
```

As we can see above that the Random Forest model is giving us the accuracy of 94.6%, which is very close to the accuracy given by the decision tree model.

2.2.4 Logistic Regression

Logistic Regression is somewhat similar to the Linear regression model. The main difference is Logistic regression is basically used when the target variable is categorical variable. Now lets use logistic regression model to predict the vales of the target variable.

```
> ###-----------Logistic Regression
> logit_model = glm(Move~., data=train, family = "binomial")
> logit_Predit = predict(logit_model,newdata = test , type = "response")
> logit_Predit = ifelse(logit_Predit>0.5,2,1)
> confMarix_LR = table(test$Move,logit_Predit)
> confusionMatrix(confMarix_LR)
Confusion Matrix and Statistics

   logit_Predit
      1    2
  1 1426   17
  2  187   37

               Accuracy : 0.8776
                 95% CI : (0.8609, 0.893)
    No Information Rate : 0.9676
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2258
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8841
            Specificity : 0.6852
         Pos Pred Value : 0.9882
         Neg Pred Value : 0.1652
             Prevalence : 0.9676
         Detection Rate : 0.8554
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.7846

       'Positive' Class : 1
```

As we can see above that the logistic regression model is giving us the accuracy of 87.76%, which is quite less as compared to the Decision Tree Model and Random Forest Model.

2.2.5 Naive Bayes

Naive Bayes is basically used for classification algorithm. It works on Bayes theorem of probability to predict the class of unknown dataset.

```
> NB_Predict = naiveBayes(Move~., train)
> NB_Predictions = predict(NB_Predict, test[,-17], type ='class')
> confMatrix_NB = table(test$Move, NB_Predictions)
> confusionMatrix(confMatrix_NB)
Confusion Matrix and Statistics

   NB_Predictions
      1    2
  1 1399   44
  2  126   98

               Accuracy : 0.898
                 95% CI : (0.8825, 0.9121)
    No Information Rate : 0.9148
    P-Value [Acc > NIR] : 0.9927

                  Kappa : 0.4815
 Mcnemar's Test P-Value : 5.218e-10

            Sensitivity : 0.9174
            Specificity : 0.6901
         Pos Pred Value : 0.9695
         Neg Pred Value : 0.4375
             Prevalence : 0.9148
         Detection Rate : 0.8392
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8038

       'Positive' Class : 1
```

As we can see that Naive Bayes is giving us the accuracy of 89.8%, which is less as compared to the accuracy of Decision tree and Random forest model.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have 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 the models. We can compare the model using any of criteria. There are different classification metrics available to evaluate the performance of the model.

- Confusion Matrix
- Accuracy
- Misclassification
- Specificity
- Recall
- False Negative Rate

We will use Accuracy and False Negative Rate for evaluating our model.

3.1.1 Accuracy

Accuracy is one of the measure which we can use to evaluate the performance of the model.

Accuracy of Decision Tree Model : 94.96%
Accuracy of Random Forest Model : 94.6%
Accuracy of Logistic Regression Model : 87.76%
Accuracy of Naive Bayes Model : 89.8%

As we can see that Decision tree and Random Forest are giving us the highest accuracy, but we cannot only evaluate our model based on accuracy. We should consider other metrics also.

3.1.2 False Negative Rate

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

The lesser the value of FNR the better will be the performance of our model.

FNR of Decision Tree: 32.58%
FNR of Random Forest: 38.83%
FNR of Logistic Regression: 83.48%
FNR of Naive Bayes: 56.25%

As we can see that FNR of the Decision Tree model is very less as compared to other models. Hence we can say that Decision Tree will give better performance in predicting the values of target variable of test data set as compared to other models.

3.2 Model Selection

We can use Decision Tree model, as we can see above that the accuracy of decision tree is high than the other models and also the False Negative Rate of Decision Tree is lower than other models.

Appendix A - R Code

Churn Reduction BoxPlot (Fig 2.1)

```
for ( i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Move"), data =
subset(Data_Frame_train))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=1,
                       outlier.size=3, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(x="Move")+
          ggtitle(paste(cnames[i])))+scale_x_discrete(breaks = NULL)
}
```

Outlier Analysis (Fig 2.2)

```
##-------------Removing all the outliers----------------##
for(i in cnames){
  val = Data_Frame_train[,i][Data_Frame_train[,i] %in%
boxplot.stats(Data_Frame_train[,i])$out]
  Data_Frame_train[,i][Data_Frame_train[,i] %in% val] = NA
}


##------------Calculating Missing values in data frame
missing_val_removed=data.frame(apply(Data_Frame_train,2,function(x){sum(is.na(x))}))

##---------------KNN Imputation for imputing missing values-----------##
Data_Frame_train = knnImputation(Data_Frame_train, k = 5)
sum(is.na(Data_Frame_train))
```

Correlation Plot (Fig 2.3)

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

Complete R file

```
#setting The Working Directory
rm(list=ls())
setwd("D:/Data Science/Projects/Project 1")
getwd()

## installing packages

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "C50",
"e1071","Rcpp","gridExtra","MASS", "gbm")
lapply(x, require, character.only = TRUE)
rm(x)



###------------------Reading Train and Test Data
train_data=read.csv("Train_data.csv",header = T,na.strings = c(" ", "", "NA"))
train_data_copy=read.csv("Train_data.csv",header = T,na.strings = c(" ", "", "NA"))
test_Data = read.csv("Test_data.csv",header=T,na.strings = c(" ","","NA"))

##--------------------Combining test_data and train_data for Exploratory Data Analysis
train_data=rbind(train_data,test_Data)
colnames(train_data)[21]='Move'

Data_Frame_train=data.frame(train_data)

###---------------------Removing the unwanted features
Data_Frame_train = subset(Data_Frame_train,select = -c(phone.number,area.code,state))

###---------------------Assigning labels to categorical variables
Data_Frame_train$Move=factor(Data_Frame_train$Move,labels =
```

```
(1:length(levels(factor(Data_Frame_train$Move)))))
Data_Frame_train$voice.mail.plan=factor(Data_Frame_train$voice.mail.plan , labels =
(1:length(levels(factor(Data_Frame_train$voice.mail.plan)))))
Data_Frame_train$international.plan=factor(Data_Frame_train$international.plan,labels =
(1:length(levels(factor(Data_Frame_train$international.plan)))))



str(Data_Frame_train)

###----------------------Missing values Analysis
missing_val=data.frame(apply(Data_Frame_train,2,function(x){sum(is.na(x))}))

###----------------------Feature Engineering
Data_Frame_train$total_minutes <- Data_Frame_train$total.day.minutes +
Data_Frame_train$total.eve.minutes +
Data_Frame_train$total.night.minutes+Data_Frame_train$total.intl.minutes
Data_Frame_train$total_calls <-
Data_Frame_train$total.day.calls+Data_Frame_train$total.eve.calls+Data_Frame_train$total.n
ight.calls+Data_Frame_train$total.intl.calls
Data_Frame_train$Avg_time <-
Data_Frame_train$total_minutes/Data_Frame_train$total_calls
Data_Frame_train = Data_Frame_train[,c(1:17,19:21,18)]



### ----------------------Getting only numeric data from the Data_Frame_train set
numeric_index=sapply(Data_Frame_train, is.numeric)
numeric_data=Data_Frame_train[,numeric_index]
cnames=colnames(numeric_data)
cnames

###--------------Creating Box plot of all the numeric  data for outlier analysis

for ( i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Move"), data =
subset(Data_Frame_train))+
          stat_boxplot(geom = "errorbar", width = 0.5) +
          geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=1,
                       outlier.size=3, notch=FALSE) +
          theme(legend.position="bottom")+
          labs(x="Move")+
          ggtitle(paste(cnames[i])))+scale_x_discrete(breaks = NULL)
```

```
}

###-----------------------displaying box plot using grid Extra function
gridExtra::grid.arrange(gn1,gn2,gn3,gn4, ncol=4)
gridExtra::grid.arrange(gn5,gn6,gn7,gn8,ncol=4)
gridExtra::grid.arrange(gn9,gn10,gn11,gn12,gn13,ncol=5)
gridExtra::grid.arrange(gn14,gn15,gn16,gn17,gn18,ncol=5)



##-----Outlier Analysis----------##

##-------------Removing all the outliers---------------##
for(i in cnames){
  val = Data_Frame_train[,i][Data_Frame_train[,i] %in%
boxplot.stats(Data_Frame_train[,i])$out]
  Data_Frame_train[,i][Data_Frame_train[,i] %in% val] = NA
}


##------------Calculating Missing values in data frame
missing_val_removed=data.frame(apply(Data_Frame_train,2,function(x){sum(is.na(x))}))

##---------------KNN Imputation for imputing missing values-----------##
Data_Frame_train = knnImputation(Data_Frame_train, k = 5)
sum(is.na(Data_Frame_train))



##Feature Selection##

####----Plot Correlation to check the dependency amoung numeric variables------####

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



## ---------------- Chi -square test for factor variable reduction
factor_index=sapply(Data_Frame_train,is.factor)
factor_data=Data_Frame_train[,factor_index]
colnames(factor_data)
factor_data_length = length(colnames(factor_data))
```

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




## ---------------- Dimension Reduction

Data_Frame_train = subset(Data_Frame_train,select =
-c(total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))


###--------------------Model Development----------------##

###----------------Separating test and train after doing EDA

train=Data_Frame_train[1:3333,]
test = Data_Frame_train[3334:5000,]
rownames(test) <- NULL


## ------------C5.0
C50_model=C5.0(Move~.,train,trials=100,rules=TRUE)
summary(C50_model)
C50_predications=predict(C50_model,test[,-17],type="class")

#Evaluate the performance of classification model
confMat_C50=table(test$Move,C50_predications)
confusionMatrix(confMat_C50)


#Accuracy 94.96
##FNR 32.58

## ----------------- Random Forest

RF_model= randomForest(Move~.,train, importance = TRUE, ntree=700)
RF_Predictions = predict(RF_model, test[,-17])
confMatrix_RF = table(test$Move, RF_Predictions)
confusionMatrix(confMatrix_RF)

#Accuracy 94.6
```

```
#FNR 38.83%



###-----------Logistic Regression
logit_model = glm(Move~., data=train, family = "binomial")
logit_Predit = predict(logit_model,newdata = test , type = "response")
logit_Predit = ifelse(logit_Predit>0.5,2,1)
confMarix_LR = table(test$Move,logit_Predit)
confusionMatrix(confMarix_LR)



#Accuracy 87.76%
#FNR 83.48


# ---------- Naive Bayes

NB_Predict = naiveBayes(Move~., train)
NB_Predictions = predict(NB_Predict, test[,-17], type ='class')
confMatrix_NB = table(test$Move, NB_Predictions)
confusionMatrix(confMatrix_NB)

#Accuracy 89.8
#FNR 56.25
```

References

- Edwisor.com - Online Learning Material
- https://www.analyticsvidhya.com/