Project Report
On
Churn Reduction

Vismay Dhobe

10th September 2018

Contents

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. 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.

1.2 Data

The predictors provided are as follows:

1) State
2) Account length
3) Area code
4) Phone number
5) International plan
6) Voicemail plan
7) Number of voicemail messages
8) Total day minutes used
9) Day calls made
10) Total day charge
11) Total evening minutes
12) Total evening calls
13) Total evening charge
14) Total night minutes
15) Total night calls
16) Total night charge
17) Total international minutes used
18) Total international calls made
19) Total international charge
20) Number of customer service calls made

The target variable is

Churn : if the customer has moved (True or False)

2. Methodology

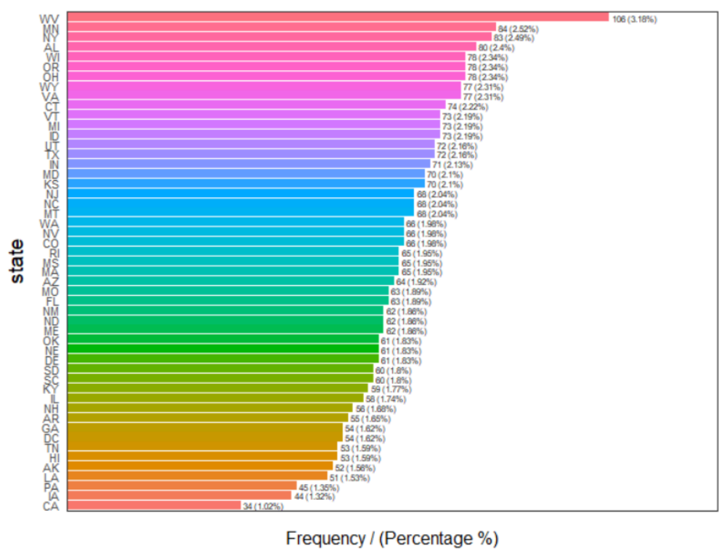
2.1 Pre-Processing

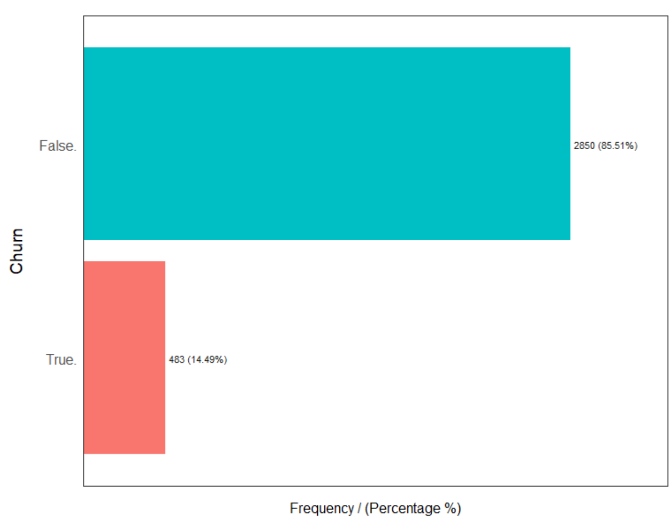
Data preprocessing is a data science technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. This is often called as exploratory data analysis.

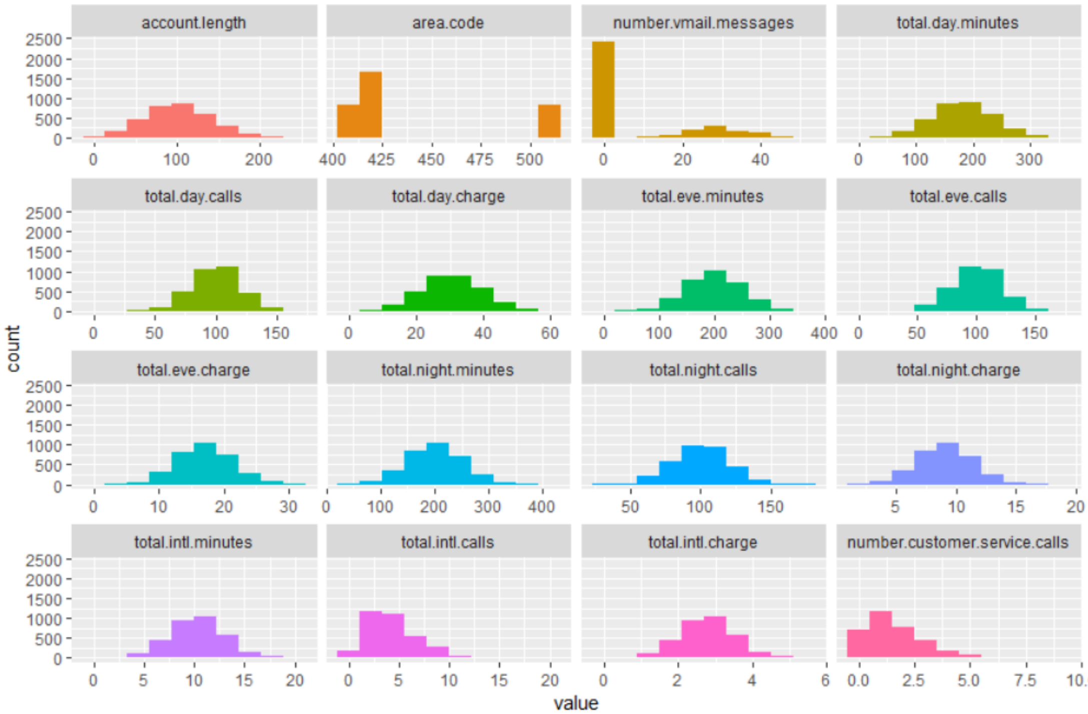
We have observed some key fetures of the data set using basic EDA as follows

Distribution of Categorical variables:





Distribution of numerical variables:



	variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
1	state	0	0.00	0	0	0	0	factor	51
2	account.length	0	0.00	0	0	0	0	integer	212
3	area.code	0	0.00	0	0	0	0	integer	3
4	phone.number	0	0.00	0	0	0	0	factor	3333
5	international.plan	0	0.00	0	0	0	0	factor	2
6	voice.mail.plan	0	0.00	0	0	0	0	factor	2
7	number.vmail.messages	2411	72.34	0	0	0	0	integer	46
8	total.day.minutes	2	0.06	0	0	0	0	numeric	1667
9	total.day.calls	2	0.06	0	0	0	0	integer	119
10	total.day.charge	2	0.06	0	0	0	0	numeric	1667
11	total.eve.minutes	1	0.03	0	0	0	0	numeric	1611
12	total.eve.calls	1	0.03	0	0	0	0	integer	123
13	total.eve.charge	1	0.03	0	0	0	0	numeric	1440
14	total.night.minutes	0	0.00	0	0	0	0	numeric	1591
15	total.night.calls	0	0.00	0	0	0	0	integer	120
16	total.night.charge	0	0.00	0	0	0	0	numeric	933
17	total.intl.minutes	18	0.54	0	0	0	0	numeric	162
18	total.intl.calls	18	0.54	0	0	0	0	integer	21
19	total.intl.charge	18	0.54	0	0	0	0	numeric	162
20	number.customer.service.calls	697	20.91	0	0	0	0	integer	10
21	Churn	0	0.00	0	0	0	0	factor	2

2.2.1 Missing Value Analysis

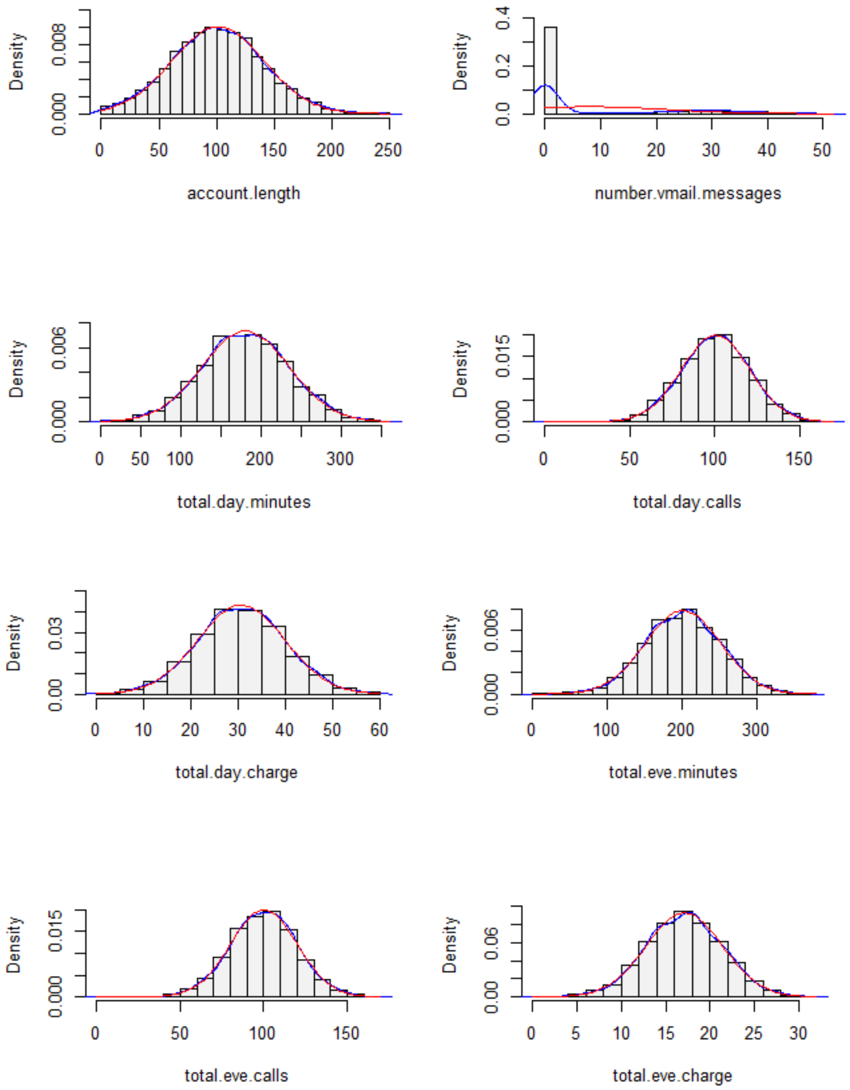
Missing value can arise due to many cases and Proper handling of missing values is important in all statistical analyses. Missing values are imputed using different methods such as Mean, median and KNN imputation. The criterion for imputation is that the variable should have missing values less than 30 percent, In this case no variable has missing values more than 30 percent. To choose the method for imputation we purposefully create a NA and try to impute it using different methods, whichever method gives the closest output, we freeze that method. Method may vary from variable to variable and it mainly depends upon the whether variable is categorical or continuous. Following is the percentage of missing values in each variable

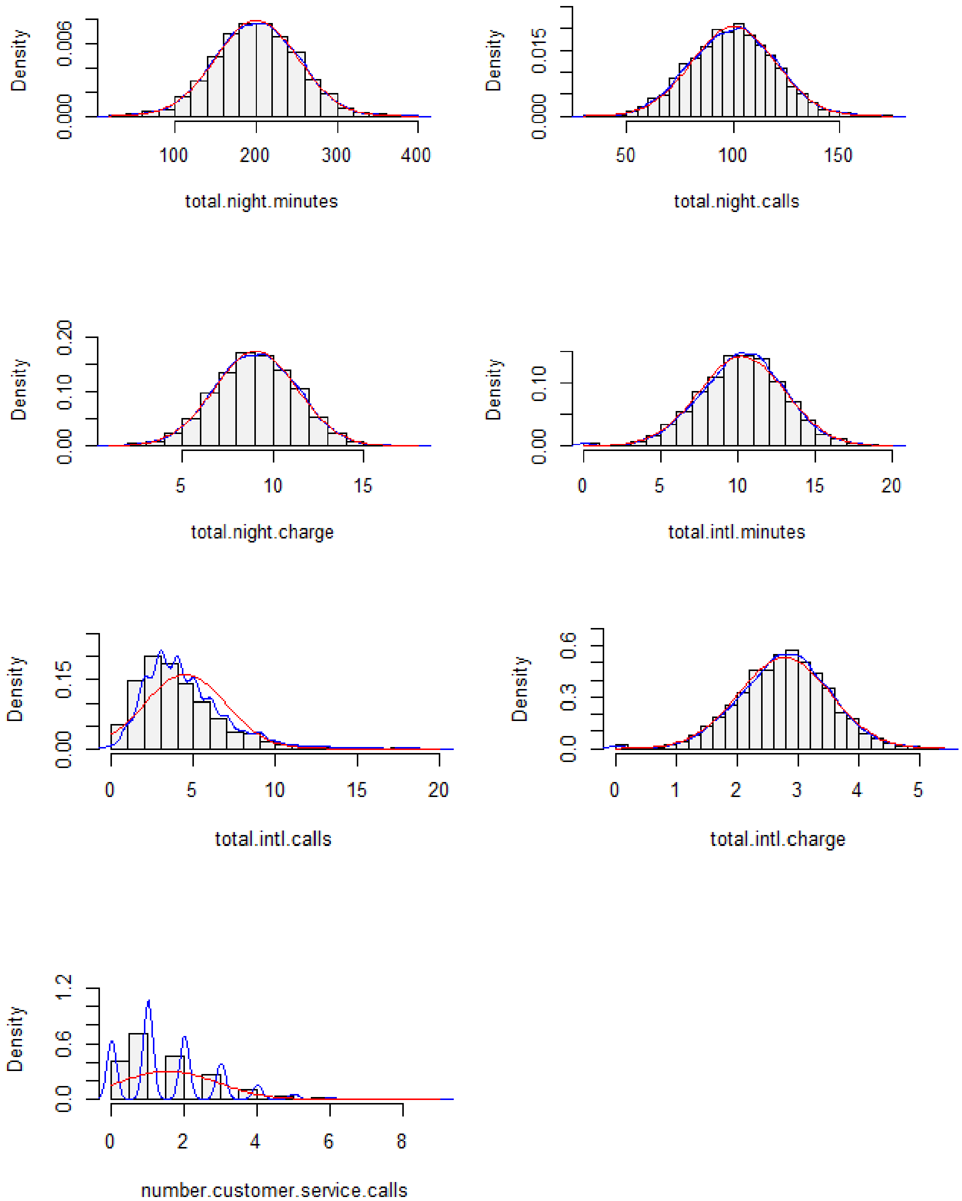
	variable	no_of_missing_values	Missing_percentage
1	state	0	0
2	account.length	0	0
3	area.code	0	0
4	phone.number	0	0
5	international.plan	0	0
6	voice.mail.plan	0	0
7	number.vmail.messages	0	0
8	total.day.minutes	0	0
9	total.day.calls	0	0
10	total.day.charge	0	0
11	total.eve.minutes	0	0
12	total.eve.calls	0	0
13	total.eve.charge	0	0
14	total.night.minutes	0	0
15	total.night.calls	0	0
16	total.night.charge	0	0
17	total.intl.minutes	0	0
18	total.intl.calls	0	0
19	total.intl.charge	0	0
20	number.customer.service.calls	0	0
21	Churn	0	0

As we can notice, there are no missing values in any of the predictor as well as in target variable Hence we do not require Missing value analysis here.

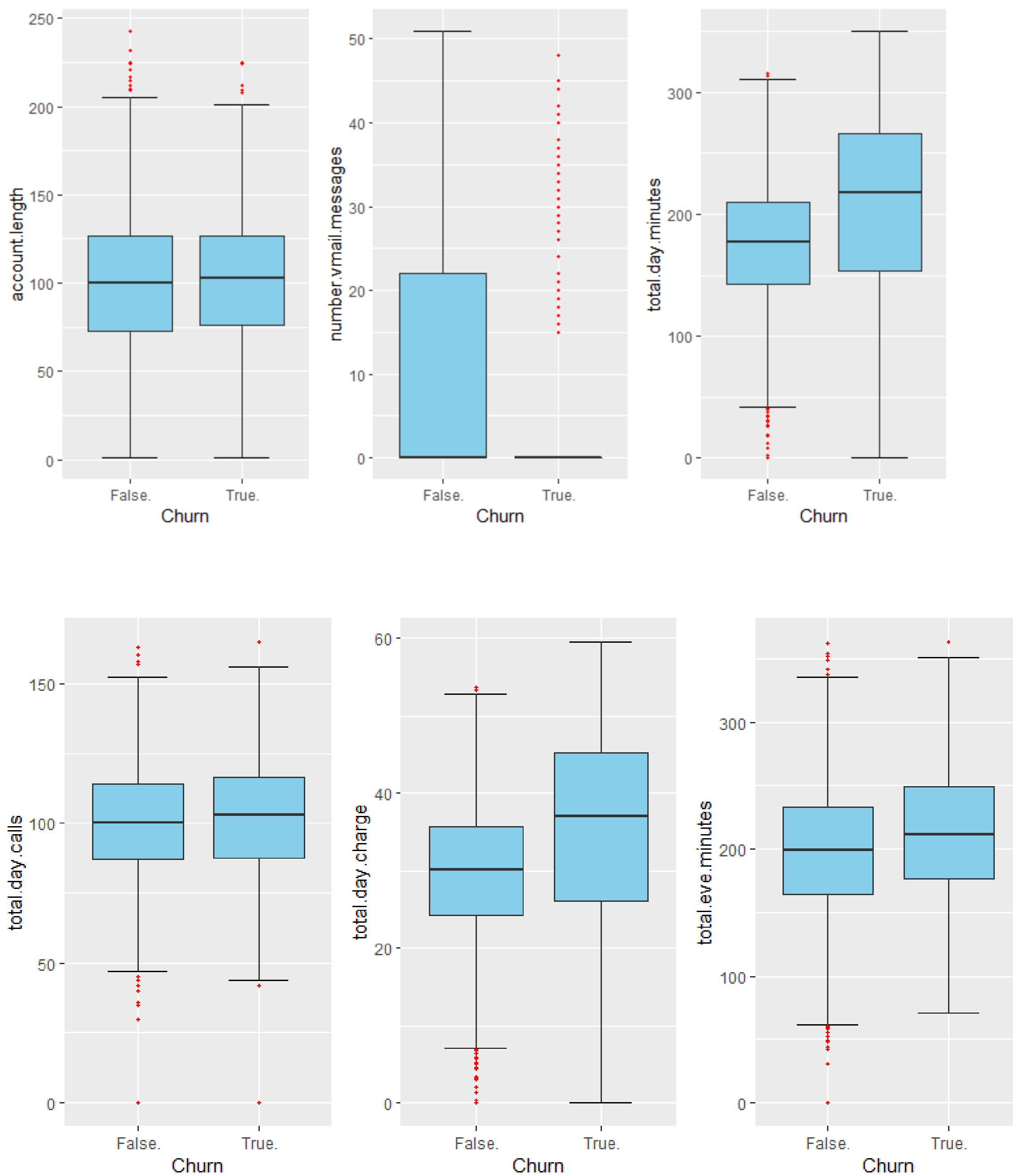
2.2.2 Outlier Analysis

Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors. This data contains outliers in variables such as 'total day minutes', 'total day charge' etc. Below are histograms of numerical variables. we need to perform outlier analysis which will try to impute the outliers with medians or means of the data or we can cap them to a particular percentile value. Although distribution plots does not help us to find the outliers, we use boxplot method to identify and remove or impute the outliers. We visualize the data using boxplots.

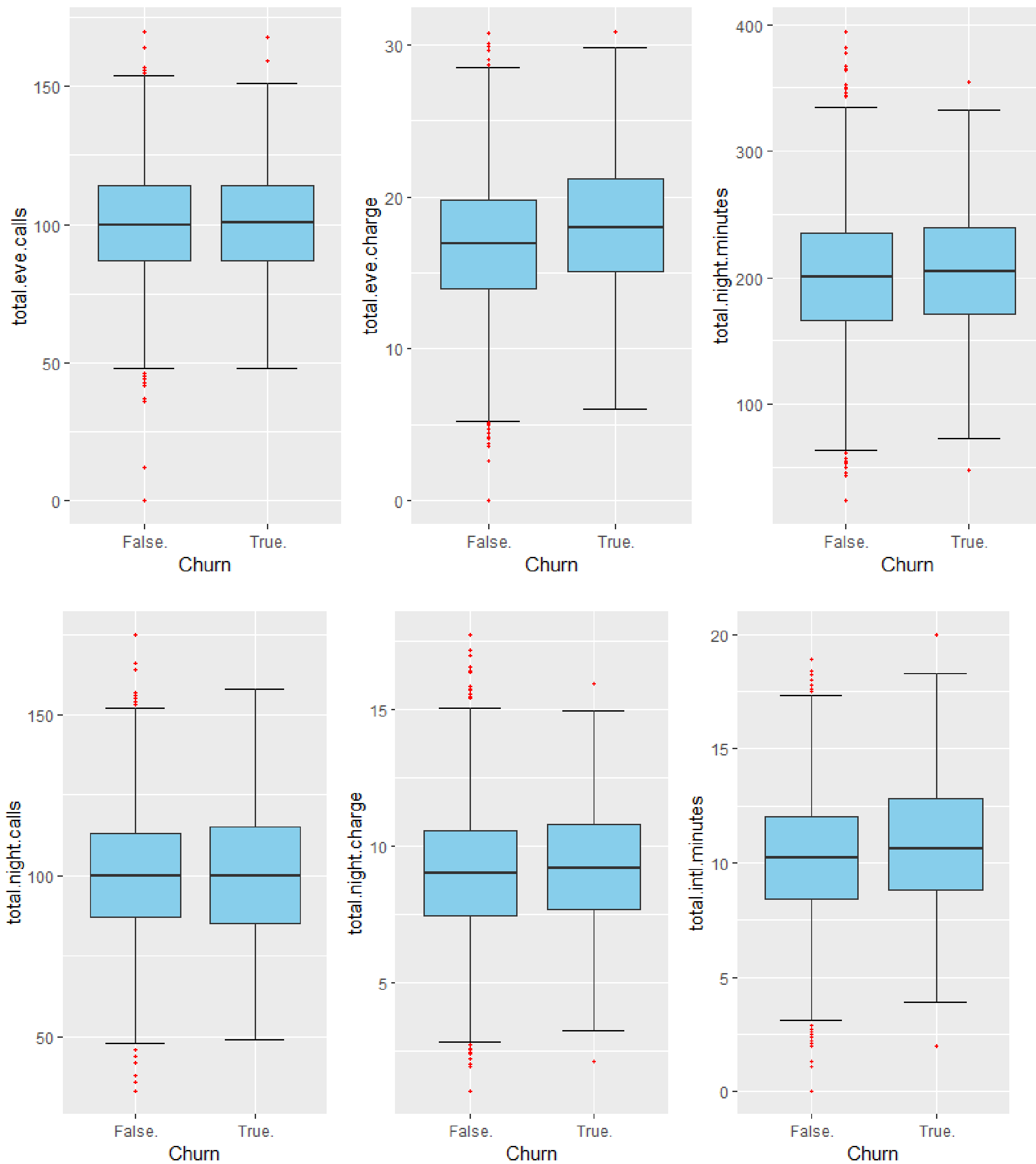




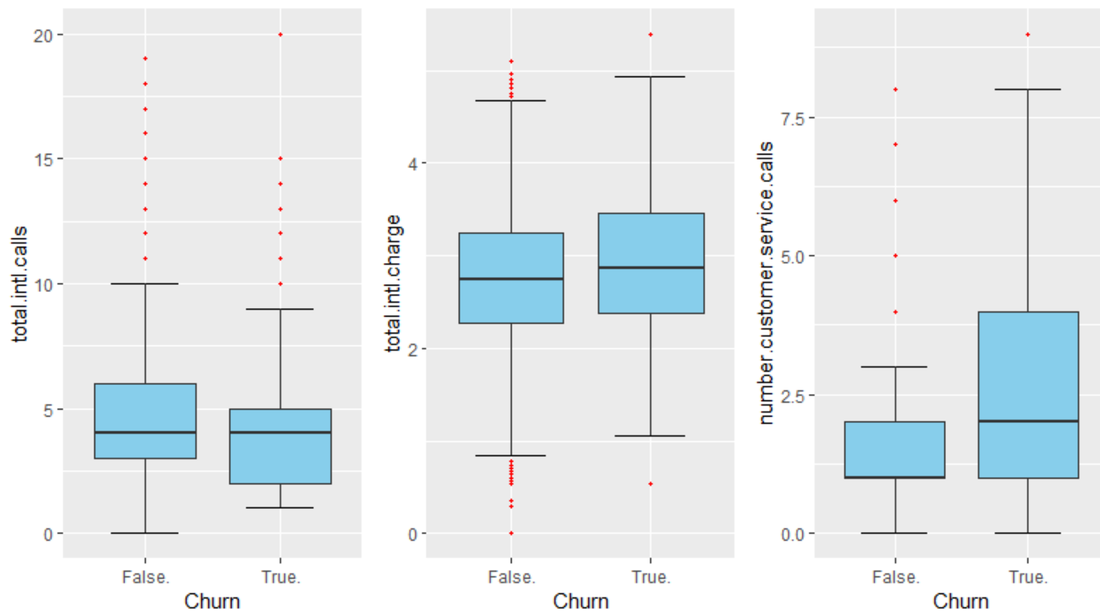
We can clearly see the distribution is not normal for some of the variables hence below are the boxplots for the same. In these boxplots the observations which are above or below 1.5 times the interquartile range are marked as outliers. Interquartile range is shaded as blue while outliers are marked as Red dots.



account.length
number.vmail.messages
total.day.minutes
total.day.calls
total.day.charge
total.eve.minutes
Churn
False.
True.



total.eve.calls
0
50
100
150
total.eve.charge
0
10
20
30
total.night.minutes
100
200
300
400
total.night.calls
50
100
150
total.night.charge
5
10
15
total.intl.minutes
0
5
10
15
20
False.
True.
Churn



We used the boxplot method to identify and cap the outliers from the variables to value of 95 percentile and 5 percentile.

```
library(scales)
for (i in cnames)
{
  print(i)
  pr <- .95
  q<- quantile( d1[,i], c(1-pr, pr))
  d1[,i] <- squish( d1[,i], q)
}
```

2.2.3 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. We are here using correlation plot to identify the correlation between numerical variables and we will neglect the variable which are highly correlated to each other so that they does not carry the same information to the model development. In same way we will use Chi-Square test for categorical variable.

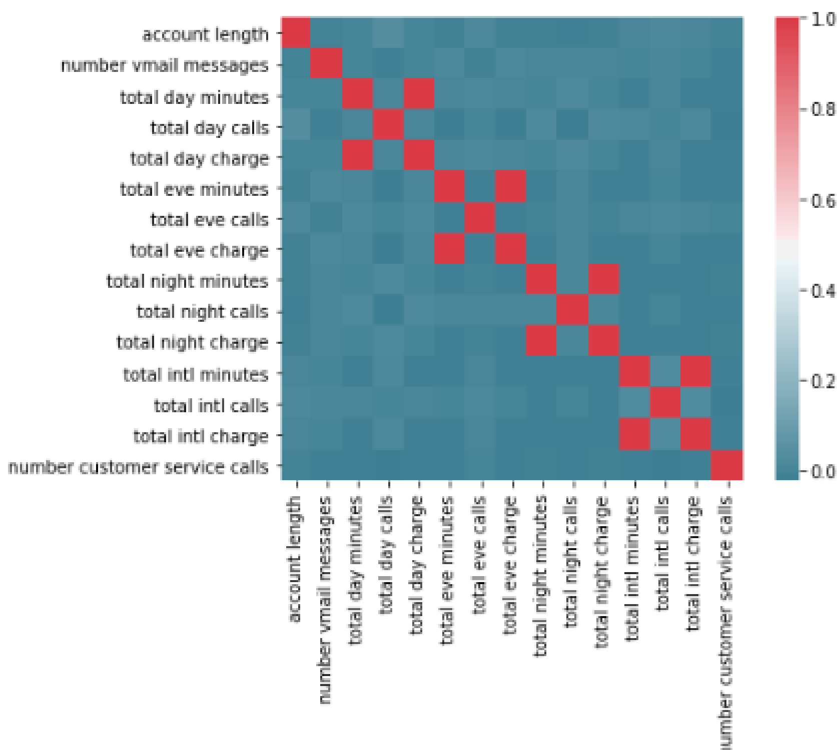
The chi-square test is a statistical test of independence to determine the dependency of two variables. It shares similarities with coefficient of determination, R^2. However, chi-square test is only applicable to categorical or nominal data while R^2 is only applicable to numeric data. Below, we can see the output of Chi-square test of independence for categorical variables from given data

```
state
0.002296221552011188
area code
0.9150556960243712
international plan
2.4931077033159556e-50
voice mail plan
5.15063965903898e-09
```

This test resides on P value of the variable. The null hypothesis of this test is that the two variables are independent of each other hence Alternate hypothesis would be that they are dependent on each other hence if the P value is greater than 0.05 which allows null hypothesis to be correct and states that the variables are not dependent on each other.

Here, ‘area code’ is the categorical variables which follows null hypothesis and hence the target variable is independent of this variable. Hence we will drop this variable and proceed with other for model development

Correlation plot for the numerical variables from data is given below



Here, Dark Red indicates that the two variables are highly positively correlated to each other and Dark Blue indicates that the two variables are highly negatively correlated to each other. As we can see ‘total day minutes’ and ‘total day charge’ are highly positively correlated to each other so we should drop any one variable from them to avoid multicollinearity.

Using above two techniques to eliminate variables, we have dropped 'total day charge', 'total eve charge', 'total night charge', 'total intl charge' from the data set.

2.2.4 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data.
There are two methods of feature scaling viz. Normalization and standardization.
Standardization is applied when data is normally distributed and normalization is applied in other cases. Here we have used normalization as there is skewness in some variables.
This method of EDA is only applicable for some modelling techniques such as KNN.

3. Modelling

Model selection depends on the Target variable. In case of given problem statement and dataset the target variable is categorical, hence the model will be classification model.
For classification, there are many models with which we can train our data and test on the same. We will consider some models in here and then depending on error rate we will decide on the same.

Error Metric:

Confusion Matrix	Predicted false	Predicted true
Actual false	TN	FP
Actual true	FN	TP

We have customized the confusion matrix according to problem need. Based on above confusion metric, we have some parameters with which we can judge our model and chose accordingly

Accuracy	(TP+TN)/(TP+TN+FP+FN)
Precision	TP/(TP+FN)
Recall	TP/(FP+TP)
F1 Score	2*Precision*Recall/(Precision+Recall)

Here we have performed different EDA for each model depending on the results of the model such as we have compared the accuracy using outlier analysis and without using outlier analysis

3.1 Decision Tree Algorithm for Classification

```
#Develop Model on training data
C50_model = C5.0(Churn ~., d1, trials = 100, rules = TRUE)
summary(C50_model)
#Test the data with testing data
C50_Predictions = predict(C50_model, test[,-15], type = "class")
ConfMatrix_C50 = table(test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1438	5
Actual true	65	159

Accuracy	0.958008398
Precision	0.709821429
Recall	0.969512195
F1 Score	0.819587629

Above result was taken without performing outlier analysis on the variables.
After performing outlier analysis, results came out as follows

Confusion Matrix	Predicted false	Predicted true
Actual false	1438	5
Actual true	70	154

Accuracy	0.955008998
Precision	0.6875
Recall	0.968553459
F1 Score	0.804177546

3.2 Random Forest

```
#Random Forest
RF_model = randomForest(Churn ~ ., d1, importance = TRUE,ntree=300)
#Presdict test data
RF_Predictions = predict(RF_model, test[,-15])
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1424	19
Actual true	65	159

Accuracy	0.949610078
Precision	0.709821429
Recall	0.893258427
F1 Score	0.791044776

3.3 Logistic Regression using cut off probability as .025

```
#Logistic Regression
logit_model = glm(Churn ~ ., data = d1, family = "binomial")
summary(logit_model)
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test[,-15], type = "response")
logit_Predictions = ifelse(logit_Predictions > 0.25, 1, 0)
ConfMatrix_LR = table(test$Churn, logit_Predictions)
ConfMatrix_LR
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1242	201
Actual true	93	131

Accuracy	0.823635273
Precision	0.584821429
Recall	0.394578313
F1 Score	0.471223022

We performed VIF test for multicollinearity here and came to know that 'number vmail messages' posseses VIF >10 hence after removing this variable we retrained the model and then output came as follows

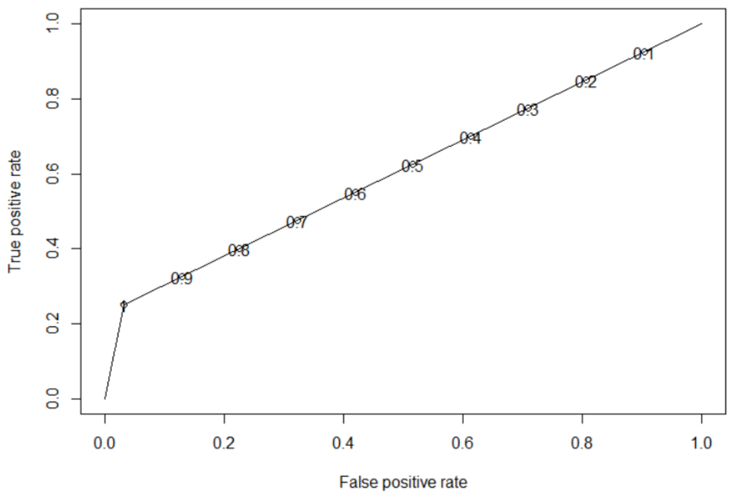
```
#Logistic Regression
logit_model = glm(Churn ~ ., data = d1, family = "binomial")
summary(logit_model)
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test[,-14], type = "response")
logit_Predictions = ifelse(logit_Predictions > 0.25, 1, 0)
ConfMatrix_LR = table(test$Churn, logit_Predictions)
ConfMatrix_LR
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1240	203
Actual true	89	135

Accuracy	0.824835033
Precision	0.602678571
Recall	0.399408284
F1 Score	0.480427046

We have used ROC curve for deciding cut off frequency for logistic regression. The ROC curve is given as follow

```
library(ROCR)
pred=prediction(logit_Predictions,test$Churn)
pref=performance(pred,"tpr","fpr")
plot(pref,colorsize=TRUE,print.cutoffs.at=seq(0.1,by=0.1))
```



3.4 KNN Classification

```
#KNN classification
library(class)
#Predict test data
KNN_Predictions = knn(d1[, 1:14], test[, 1:14], d1$Churn, k = 3)
Conf_matrix = table(KNN_Predictions, test$Churn)
sum(diag(Conf_matrix))/nrow(test)
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1283	160
Actual true	137	87

Accuracy	0.821835633
Precision	0.388392857
Recall	0.352226721
F1 Score	0.369426752

3.5 Naïve Bayes Model

```
#Naive Bayes
library(e1071)
NB_model = naiveBayes(Churn ~ ., data = d1)
NB_Predictions = predict(NB_model, test[,1:14], type = 'class')
Conf_matrix = table(observed = test[,15], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)
```

Confusion Matrix	Predicted false	Predicted true
Actual false	1370	73
Actual true	125	99

Accuracy	0.881223755
Precision	0.441964286
Recall	0.575581395
F1 Score	0.5

4. Conclusion

4.1 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 models using any of the following criteria:
1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In this case, if we consider Predictive performance of all models then after comparing their F1 Scores we can pick Decision Tree model as it is giving F1 score of 0.819.
We can also prefer Random Forest model as it is giving F1 score of 0.80 with number of trees as 300

4.2 Model Validation

Data was divided in train and test with 3333 train observation and 1667 as test observation Hence all the models were validated with whole 1667 observations each time and hence model validation leads to Decision Tree model and this model gave un-deviated output

Sample output is given with Pred_output.csv attached with this Report. We can validate the results using this file

5. Appendix

5.1 R Code

```
rm(list=ls())
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees',"usdm","class","tidyverse","funModeling","Hmisc")
lapply(x, require, character.only = TRUE)
rm(x)

d1=read.csv("Train_data.csv")
test=read.csv("Test_data.csv")
d1=d1_tmp

basic_eda <- function(d1)
{
  glimpse(d1)
  df_status(d1)
  freq(d1)
  profiling_num(d1)
  plot_num(d1)
  describe(d1)
}

basic_eda(d1)

sum(is.na(d1))
#outlier analysis
numeric_index = sapply(d1,is.numeric) #selecting only numeric
numeric_data = d1[,numeric_index]
cnames = colnames(numeric_data)

library(psych)
for (i in cnames){
  multi.hist(numeric_data[,i], main = i, dcol = c("blue", "red"),
           dlty = c("solid", "solid"), bcol = "grey95")
}

for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "Churn"), data = d1)+
           stat_boxplot(geom = "errorbar", width = 0.5) +
```

```
        geom_boxplot(outlier.colour="red", fill = "skyblue" ,outlier.shape=20,
                outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i],x="Churn")
        )
}

#we can use squish function to capp the outliers to particular values
library(scales)
for (i in cnames)
{
  print(i)
  pr <- .95
  q<- quantile( d1[,i], c(1-pr, pr))
  d1[,i] <- squish( d1[,i], q)
}

#Feature selection
#Numerical variable
numeric_index = sapply(d1,is.numeric)
numeric_data = d1[,numeric_index]
cnames = colnames(numeric_data)

corrgram(d1[,numeric_index], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#categorical variable
factor_index =  sapply(d1,is.factor)
factor_data = d1[,factor_index]
cat_names = colnames(factor_data)

for (i in 1:length(cat_names))
{
  print(names(factor_data)[i])
  print(chisq.test(table(d1$Churn,factor_data[,i],simulate.p.value = TRUE)))
}

d1= subset(d1,select =-
c(total.day.charge,total.eve.charge,total.night.charge,total.intl.charge,area.code))


##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(Churn ~., d1, trials = 100, rules = TRUE)
```

```
summary(C50_model)
#Test the data with testing data
C50_Predictions = predict(C50_model, test[,-15], type = "class")
ConfMatrix_C50 = table(test$Churn, C50_Predictions)
confusionMatrix(ConfMatrix_C50)
write.csv(C50_Predictions,"pred_output.csv")

#Random Forest
RF_model = randomForest(Churn ~ ., d1, importance = TRUE,ntree=350)
#Presdict test data
RF_Predictions = predict(RF_model, test[,-15])
ConfMatrix_RF = table(test$Churn, RF_Predictions)
confusionMatrix(ConfMatrix_RF)


#Logistic Regression
logit_model = glm(Churn ~ ., data = d1, family = "binomial")
summary(logit_model)
#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test[,-14], type = "response")
logit_Predictions = ifelse(logit_Predictions > 0.25, 1, 0)
ConfMatrix_LR = table(test$Churn, logit_Predictions)
ConfMatrix_LR

library(ROCR)
pred=prediction(logit_Predictions,test$Churn)
pref=performance(pred,"tpr","fpr")
plot(pref,colorsize=TRUE,print.cutoffs.at=seq(0.1,by=0.1))

#calculate VIF for logistic regression
library(usdm)
vif(d1[,-15])
vifcor(d1[,-15], th = 0.9)
d1_temp=d1
test_temp=test
d1=subset(d1,select=-c(number.vmail.messages))
test=subset(test,select=-c(number.vmail.messages))

#KNN classification
library(class)
#Predict test data
KNN_Predictions = knn(d1[, 1:14], test[, 1:14], d1$Churn, k = 3)
Conf_matrix = table(KNN_Predictions, test$Churn)
sum(diag(Conf_matrix))/nrow(test)
```

```
#Naive Bayes
library(e1071)
NB_model = naiveBayes(Churn ~ ., data = d1)
NB_Predictions = predict(NB_model, test[,1:14], type = 'class')
Conf_matrix = table(observed = test[,15], predicted = NB_Predictions)
confusionMatrix(Conf_matrix)
```