Churn detection – loss of customers

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 analytic concepts.

Problem Statement:

The objective of this Case is to predict customer behavior. 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.

Understanding predictors in dataset:

1. state - A state is a variable which defines customer's state that they belongs to. Categorical variable containing 51 levels.

2. account.length - This is an integer value represents how long account has been active.

3. area.code - This can be consider as categorical variable which defines the area code of area that customer resides.

4. phone.number - This is an factor variable represents phone number of a customer. It doesn't serve any purpose while building model as it is an unique value across train and test data. It is like an id that refers a customer in a unique manner.

5. international.plan - It is a binary value describing whether customer activates international plan or not.

6. voice.mail.plan - It is a binary value describing whether customer activates voice mail plan or not.

7. number.vmail.messages - Voice mail messages count sent by customer. It is an integer variable.

8. total.day.minutes - Total day minutes used. It is an integer variable.

9. total.day.calls - Total day calls done. It is an integer variable.

10. total.day.charge - Total day charge charged for customers. It is an integer variable.

11. total.eve.minutes - Total eve minutes used. It is an integer variable.

12. total.eve.calls - Total eve calls called by customer. It is an integer variable.

13. total.eve.charge - Total eve charge charged for customers. It is an integer variable.

14. total.night.minutes - Total night minutes used. It is an integer variable.

15. total.night.calls - Total night calls called by customer. It is an integer variable.

16. total.night.charge - Total night charge charged for customers. It is an integer variable.

17. total.intl.minutes - Total international minutes used. It is an integer variable.

18. total.intl.calls - Total international calls called by customer. It is an integer variable.

19. total.intl.charge - Total international charge charged for customers. It is an integer variable.

20. number.customer.service.calls - Number of service calls made by customers.

Target Variable:

21. Churn : If the customers moved or not (True or False)

Basic exploration:

Let's perform some basic data analysis, which will help us to find the statistical properties of the training dataset. This kind of analysis is called exploratory data analysis and it will help us understand how our dataset represents the facts. After deriving some facts, we can use them in order to derive feature engineering.

Basic data analysis is a crucial step for understanding data. Now on wards, you will encounter code in R or Python with output explanation. All codes will be attached in attachments and some referred here too.
*Note: Accuracy and other metrics measure has been posted with respect to R code output. Python code accuracy may vary.

Steps of data analysis:

1. Listing statistical properties
2. Convert variables into required datatype
3. Missing value analysis
3. Outlier analysis
4. Feature selection
5. Feature scaling

These analysis on data will bring us tremendous value while modeling data. Understanding data on this manner gives way for us to better tune our parameters which in turn improve our accuracy. After processing data model selection and finalization process can be done.

1**. Listing statistical properties:**

In this section, we will get an idea about the statistical properties of the training dataset. Using summary function in R, we can list out the following properties:

For continuous variable:

1. Min: Minimum value of the given column
2. 1st Quartile: 25th percentile of the given column
3. Median: 50th percentile or center value after ordering dataset
4. Mean: Average value of the given dataset
5. 3rd Quartile: 75th percentile of the given column
6. Max: Maximum value of the given column

For categorical variable:

1. Count: Count of observations in the each category of a given column

```
> summary(train)
     state      account.length    area.code       phone.number  international.plan voice.mail.plan number.vmail.messages
 WV     : 106   Min.   :  1.0   Min.   :408.0    327-1058:   1    no :3010          no :2411       Min.   : 0.000
 MN     :  84   1st Qu.: 74.0   1st Qu.:408.0    327-1319:   1    yes: 323          yes: 922       1st Qu.: 0.000
 NY     :  83   Median :101.0   Median :415.0    327-3053:   1                                     Median : 0.000
 AL     :  80   Mean   :101.1   Mean   :437.2    327-3587:   1                                     Mean   : 8.099
 OH     :  78   3rd Qu.:127.0   3rd Qu.:510.0    327-3850:   1                                     3rd Qu.:20.000
 OR     :  78   Max.   :243.0   Max.   :510.0    327-3954:   1                                     Max.   :51.000
 (Other):2824                                    (Other)  :3327
 total.day.minutes total.day.calls total.day.charge total.eve.minutes total.eve.calls total.eve.charge total.night.minutes
 Min.   :  0.0     Min.   :  0.0   Min.   : 0.00    Min.   :  0.0     Min.   :  0.0   Min.   : 0.00    Min.   : 23.2
 1st Qu.:143.7     1st Qu.: 87.0   1st Qu.:24.43    1st Qu.:166.6     1st Qu.: 87.0   1st Qu.:14.16    1st Qu.:167.0
 Median :179.4     Median :101.0   Median :30.50    Median :201.4     Median :100.0   Median :17.12    Median :201.2
 Mean   :179.8     Mean   :100.4   Mean   :30.56    Mean   :201.0     Mean   :100.1   Mean   :17.08    Mean   :200.9
 3rd Qu.:216.4     3rd Qu.:114.0   3rd Qu.:36.79    3rd Qu.:235.3     3rd Qu.:114.0   3rd Qu.:20.00    3rd Qu.:235.3
 Max.   :350.8     Max.   :165.0   Max.   :59.64    Max.   :363.7     Max.   :170.0   Max.   :30.91    Max.   :395.0

 total.night.calls total.night.charge total.intl.minutes total.intl.calls total.intl.charge number.customer.service.calls
 Min.   : 33.0     Min.   : 1.040     Min.   : 0.00      Min.   : 0.000   Min.   :0.000     Min.   :0.000
 1st Qu.: 87.0     1st Qu.: 7.520     1st Qu.: 8.50      1st Qu.: 3.000   1st Qu.:2.300     1st Qu.:1.000
 Median :100.0     Median : 9.050     Median :10.30      Median : 4.000   Median :2.780     Median :1.000
 Mean   :100.1     Mean   : 9.039     Mean   :10.24      Mean   : 4.479   Mean   :2.765     Mean   :1.563
 3rd Qu.:113.0     3rd Qu.:10.590     3rd Qu.:12.10      3rd Qu.: 6.000   3rd Qu.:3.270     3rd Qu.:2.000
 Max.   :175.0     Max.   :17.770     Max.   :20.00      Max.   :20.000   Max.   :5.400     Max.   :9.000

    Churn
 False.:2850
 True. : 483
```

As explained, note categorical variable has only count attribute and continuous variable poses all the statistical properties.

Let us visualize target class proportion because in classification task, accuracy greatly depends on the number of data of respective classes.

Here,

0 represents False

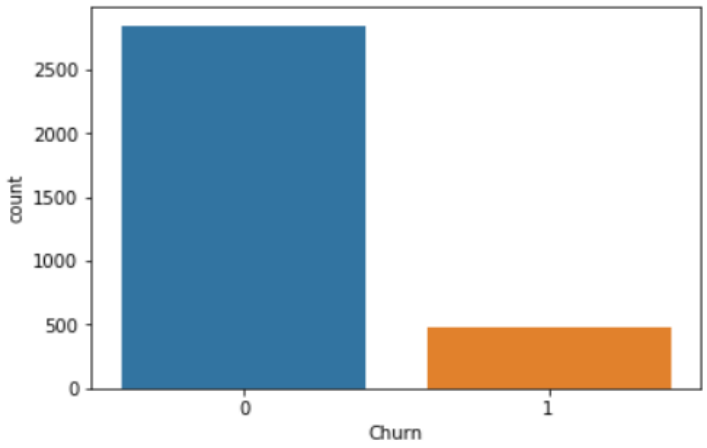
1 represents True

```
1 pd.crosstab(train['Churn'],train['Churn'])
```

Churn	0	1
Churn		
0	2850	0
1	0	483

```
1 sns.countplot(train['Churn'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x18894df5588>
```



As we see above,

False class has 2850 observations out of 3333 - 86%

True class has 483 observations out of 3333 - 14%

Hence we can say that data is heavily imbalanced towards false class.

We will generate synthetic data for deficient class later and see how model improves in terms of accuracy and some other important metrics.

2. Convert variables into required datatype

Some variables may be in inappropriate format. So we have to consider those variables and convert into intended type.

Here “area.code” variable has been as int data type. But it will be more suitable if it is in categorical variable.

This variable just contains 3 unique numbers across the dataset.

```
> unique(train$area.code)
[1] 415 408 510
> unique(test$area.code)
[1] 510 408 415
>
```

Hence convert them it into factor data type.

```
##Changing to appropriate type
train$area.code = as.factor(train$area.code)
test$area.code = as.factor(test$area.code)
```

Let's take a look at the structure of dataset.

```
> str(train)
'data.frame':   3333 obs. of  17 variables:
 $ state                        : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50 ...
 $ account.length               : int  128 107 137 84 75 118 121 147 117 141 ...
 $ area.code                    : Factor w/ 3 levels "408","415","510": 2 2 2 1 2 3 3 2 1 2 ...
 $ phone.number                 : Factor w/ 3333 levels " 327-1058"," 327-1319",..: 1927 1576 1118 1708 111 2254 1048 81 292 118 ...
 $ international.plan           : Factor w/ 2 levels " no"," yes": 1 1 1 2 2 2 1 2 1 2 ...
 $ voice.mail.plan              : Factor w/ 2 levels " no"," yes": 2 2 1 1 1 1 2 1 1 2 ...
 $ number.vmail.messages        : int  25 26 0 0 0 0 24 0 0 37 ...
 $ total.day.calls              : int  110 123 114 71 113 98 88 79 97 84 ...
 $ total.day.charge             : num  45.1 27.5 41.4 50.9 28.3 ...
 $ total.eve.calls              : int  99 103 110 88 122 101 108 94 80 111 ...
 $ total.eve.charge             : num  16.8 16.6 10.3 17.1 12.6 ...
 $ total.night.calls            : int  91 103 104 89 121 118 118 96 90 97 ...
 $ total.night.charge           : num  11.01 11.45 7.32 8.86 8.41 ...
 $ total.intl.calls             : int  3 3 5 7 3 6 7 6 4 5 ...
 $ total.intl.charge            : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
 $ number.customer.service.calls: int  1 1 0 2 3 0 3 0 1 0 ...
 $ Churn                        : Factor w/ 2 levels " False."," True.": 1 1 1 1 1 1 1 1 1 1 ...
> 
```

3. Missing value analysis

In order to find the missing values in the dataset, we need to check each and every column in the dataset. If we found any, we can replace with more meaningful values.

```
## Missing values - No missing values
missingValues=data.frame(apply(train, 2,function(x){sum(is.na(x))}))
```

	Missing value
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

As you see above, there is no missing values in the dataset. It was clean, filled with all values. So, let's move to next section.

3. Outlier analysis:

Outlier values are abnormal values that are different from normal distribution. It causes data skewed in some direction. This values should be replaced with some other values within the bound.

Outlier analysis involves following steps:

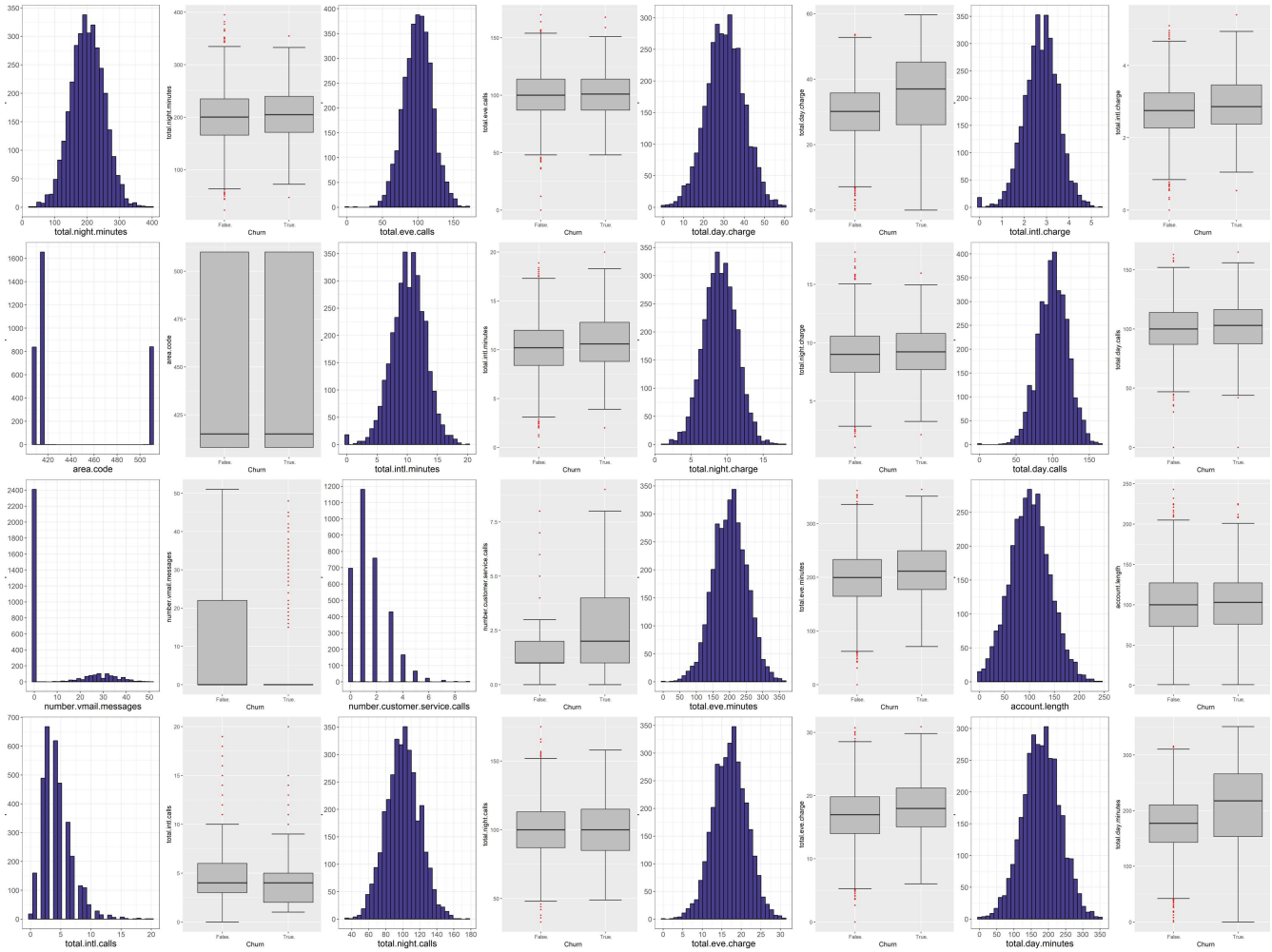
1. Detecting Outliers
2. Dealing Outliers

1. Detecting Outliers:

Outliers can be detected in many ways. Here we are going to use box plot method.

Box plot method declares outliers as the values below 25th percentile and values above the 75th percentile including some distance (iqr*1.5 where iqr is 75th -25th)

Let us visualize box plot along with distribution of data. Code in R and Python for visualization of both box plot and frequency distribution has been attached as appendix[1].



* Note: Separate image files for each column has been bagged with the report.

As you see the above image, Frequency distribution has been skewed either side if the outlier presents.

Red dots in box plot represents outliers and almost all the columns has outliers.

2. **Dealing with outliers**
   To deal with outliers, first of all replace all the outlier values with NA value. Take a look at the below code snippet.

```
##Outlier analysis
numeric_index = sapply(train, is.numeric)

numeric_data = train[, numeric_index]
cnames=colnames(numeric_data)


for (nnames in cnames){
  val = boxplot.stats(train[, nnames])$out
  train[train[,nnames] %in% val,nnames]= NA
}
## Missing values - No missing values
```

It replaces all the outlier values of numerical columns with NA value.

Now NA values can be replaced with below meaningful inbound values:

1. Mean
2. Median
3. Distance based method

After checking all the above methods and their corresponding accuracy, median yields most promising value compared to other for this dataset. So replace all the NA value with median value.

```
for (nnames in cnames){
  val = boxplot.stats(train[, nnames])$out
  train[train[,nnames] %in% val,nnames]= median(train[, nnames])
}
## Missing values - No missing values
```

All other replacement methods tried and corresponding codes has been attached in attachements. Accuracy images with respect to random forest has also been attached with attachment as images by varying outlier imputation methods.

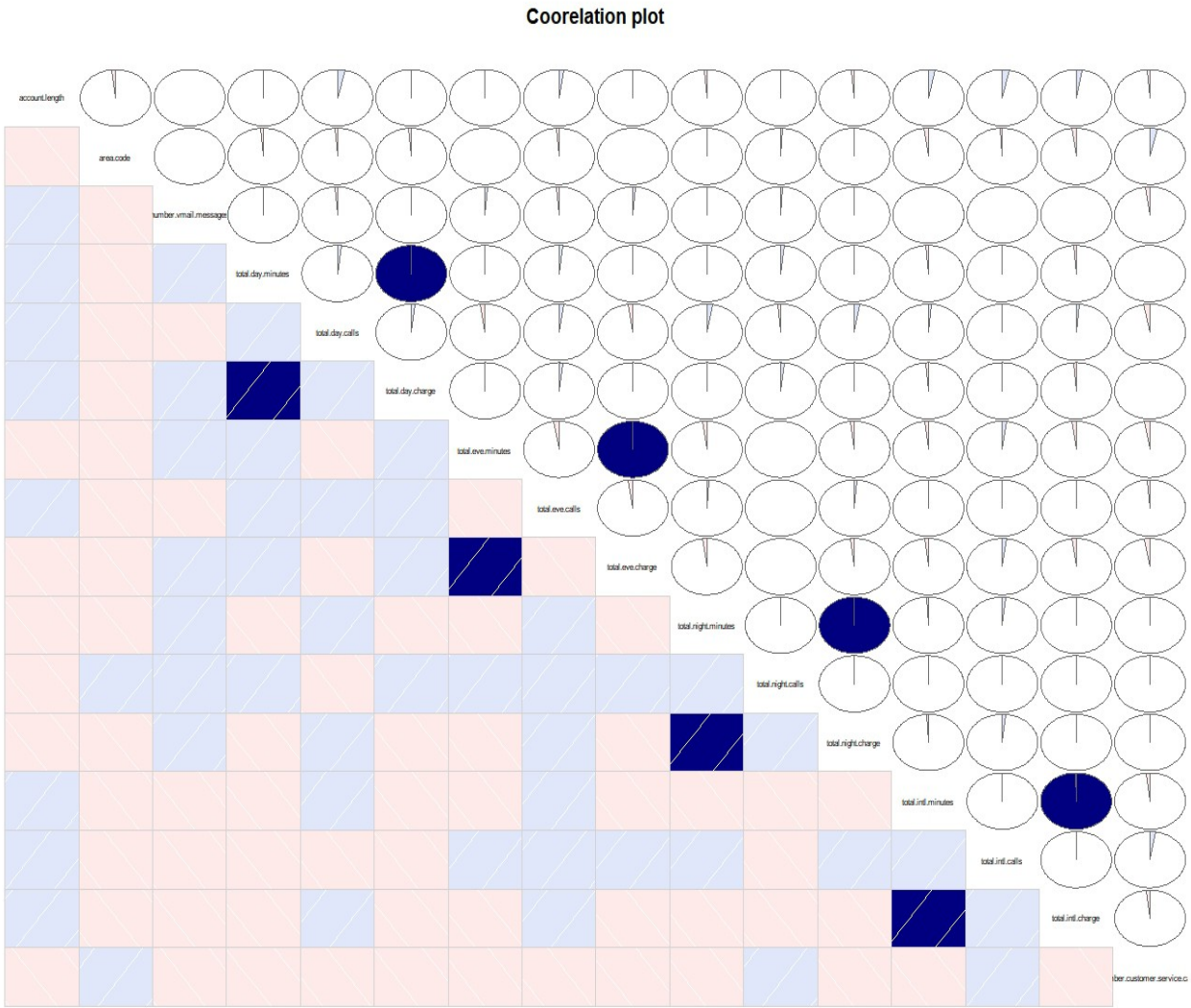
4. Feature selection:

Feature selection is a technique used for selecting most important or most contributing features towards prediction. Selecting appropriate features for modeling improves model accuracy and efficiency.

Correlation analysis is an important step in feature selection. Correlation analysis helps us to detect relationship between independent predictor variables. Correlation between predictor variables will be resulting to bad accuracy because it just repeating the same information. It may also leads to over-fitting.

For exploring correlation between numerical predictors, take a look at the below plot:

```
corrgram(train[,cnames],order = F,upper.panel = panel.pie, text.panel = panel.txt,  main = "Coorelation plot")
```



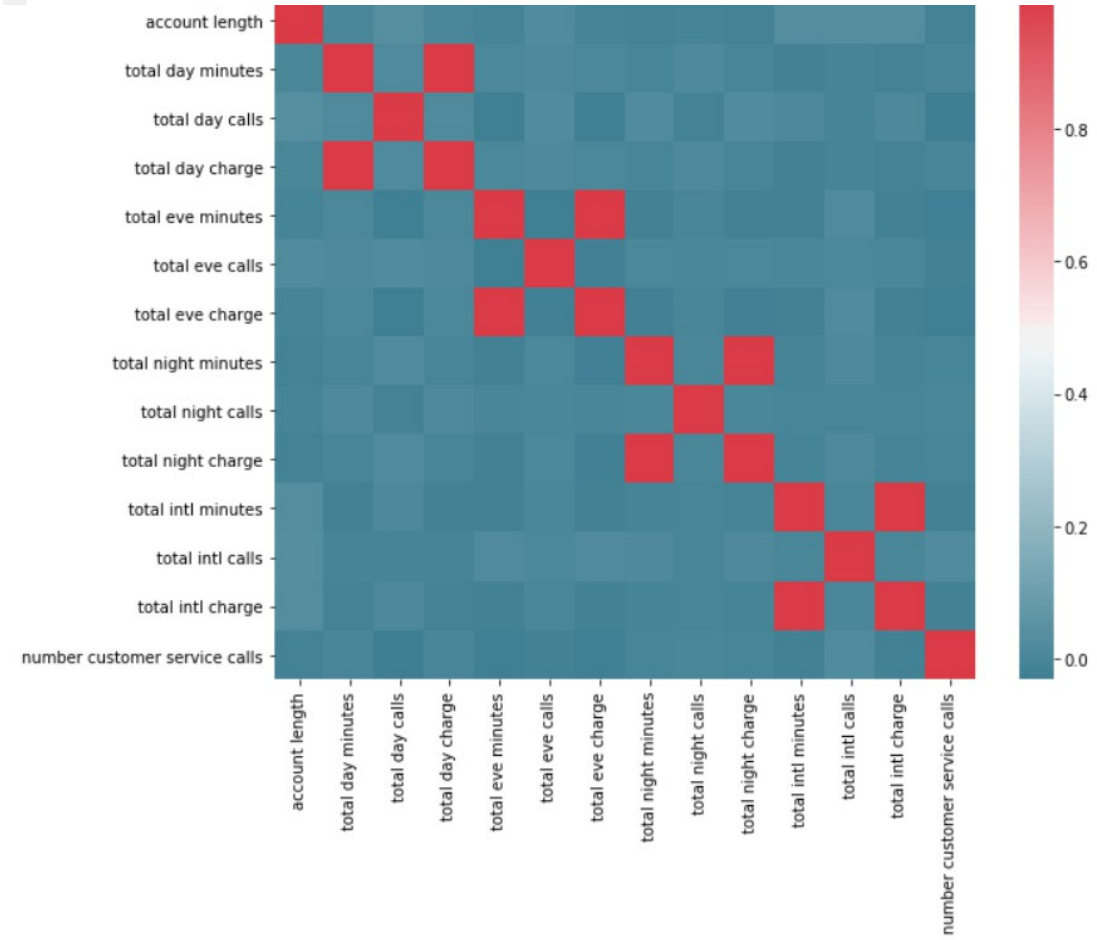
Here complete blue indicates high correlation between those variables.

Below numerical features are highly correlated in this data.

total.day.minutes ~ total.day.charge
total.eve.minutes ~ total.eve.charge
total.night.minutes ~ total.night.charge
total.intl.minutes ~ total.int.charge

Even if you see these variables semantically, we know, minutes usage directly results in increase of charge. Let's look a this relationship using heatmap.

```
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(220,10,as_cmap=True))
```



Red color in diagonal, refers self relationship. Other than that, same four variable sets has been highly correlated. It is just a redundant information. Remove one feature from correlated sets. Remove the features from both train and test data.

```
train = subset(train, select = -c(total.day.minutes, total.eve.minutes, total.night.minutes,total.intl.minutes))
test = subset(test, select = -c(total.day.minutes, total.eve.minutes, total.night.minutes,total.intl.minutes))
```

For categorical variables, Chi-square test will be done to find the contribution of independent feature towards prediction. Chi-square test has been done between independent categorical variable and dependent target variable and those two variables should be highly correlated. Chi-square test will output p value. Correlation can be determined using p value.

Let's define our hypothesis:

1. Null Hypothesis – Two variables are less correlated.
2. Alternative Hypothesis – Two variables are highly correlated.

If p value < 0.05 – Reject the null hypothesis. i.e) Two variables are Highly correlated.
P value > 0.05 – Reject alternative hypothesis. i.e) Two variables are Less correlated.

If any feature exhibits less than 0.05 p value, that feature will be selected for further processing.

```
[1] "state"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 83.044, df = 50, p-value = 0.002296

[1] "area.code"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 0.17754, df = 2, p-value = 0.9151

[1] "phone.number"

        Pearson's Chi-squared test

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 3333, df = 3332, p-value = 0.4919

[1] "international.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 222.57, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

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

data:  table(factor_data$Churn, factor_data[, i])
X-squared = 34.132, df = 1, p-value = 5.151e-09
```

Take a look at above Chi-square test result image. Test has been executed between target variable “Churn” and all independent categorical variables individually.

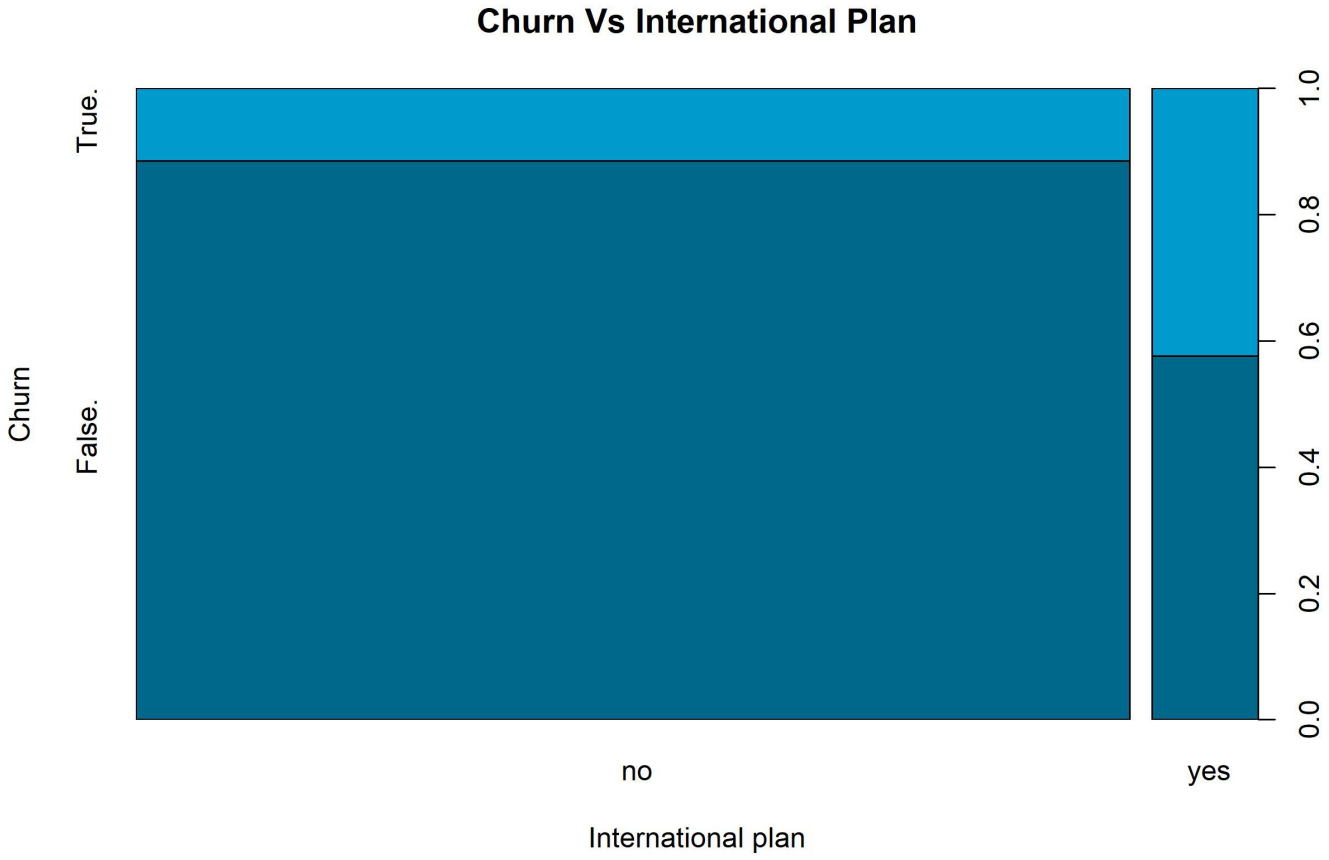
Feature name	P- value
state	0.002296
area.code	0.9151
phone.number	0.4919
international.plan	< 2.2e-16
voice.mail.plan	5.151e-09

area.code and phone.number variable has p-value > 0.05. These variable doesn't contribute in predicting target variable. Remove these variables from dataset.

```
train=subset(train, select = -c(area.code, phone.number))
test=subset(test, select = -c(area.code, phone.number))
```

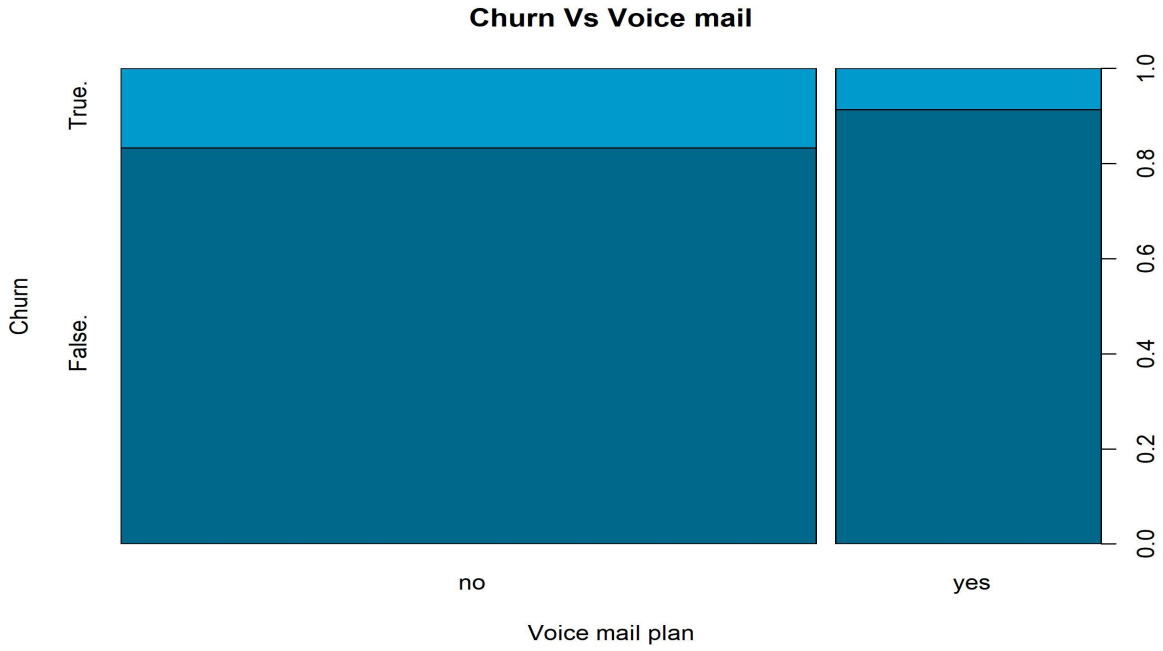
Business Insights:

Now, let's realize some hidden facts about dataset with the help of below plots.
We can regulate the business rules with these insights.



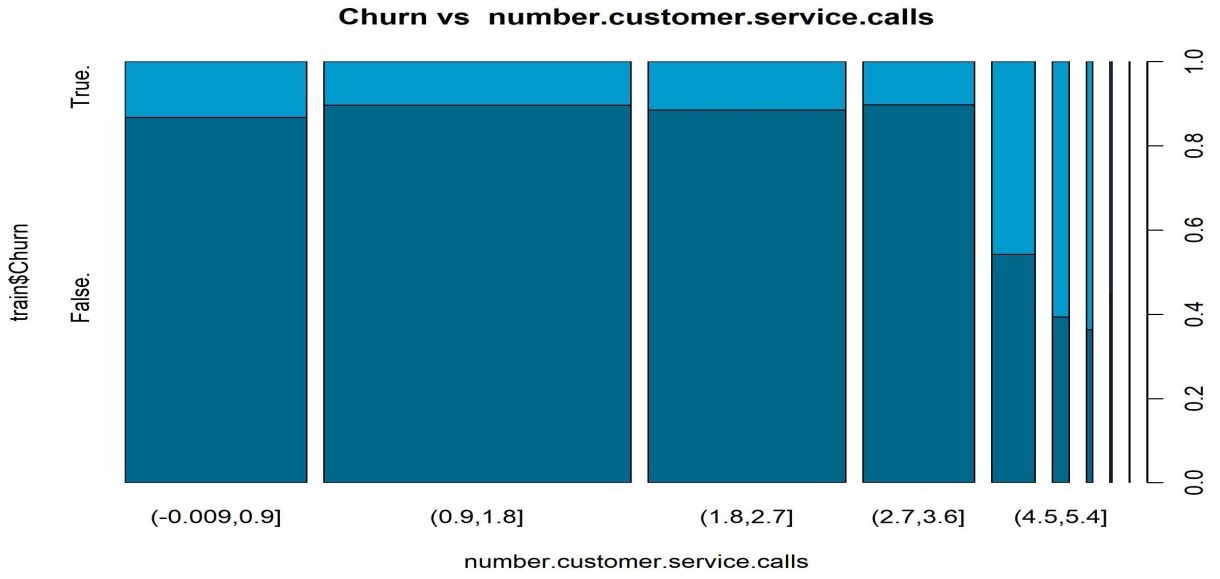
This plot reveals the hidden truth,

“Customers with international plan will churn frequently”



This plot reveals,

"Customer with voice mail plan activated will have less churn rate"



This plot reveals,

"If a customer called more than four times to a customer service centers, then that customer will more prone to churn "

*Note: Code and image of these graphs has been attached in attachment

Finally, list out the importance feature with the help of random forest technique.

First of all, train the model with current features.

```
RandomF_Model = randomForest(Churn ~., com[1:3333,], importance = TRUE,ntree = 500)
```

```
> varImp(RandomF_Model)
                                False.      True.
state                        1.4500241  1.4500241
account.length              -1.7304681 -1.7304681
international.plan          48.1241883 48.1241883
voice.mail.plan             13.4505373 13.4505373
number.vmail.messages       12.3889895 12.3889895
total.day.calls             -0.3759991 -0.3759991
total.day.charge            56.7691640 56.7691640
total.eve.calls             -2.3559873 -2.3559873
total.eve.charge            18.0917824 18.0917824
total.night.calls            0.6065154  0.6065154
total.night.charge           4.2773558  4.2773558
total.intl.calls            20.6308928 20.6308928
total.intl.charge           16.4819357 16.4819357
number.customer.service.calls  3.1933742  3.1933742
> |
```

This list defines how much a variable is contributing towards prediction variable.
If we look at it, “account.length”,”state”, “total.eve.calls”, “total.day.calls” “total.night.calls” has poor contribution towards variable prediction. Remove those variables from dataset.

```
train = removeFeatures(train, c('state','account.length','total.day.calls', 'total.eve.calls', 'total.night.calls' ))
test = removeFeatures(test, c('state','account.length','total.day.calls', 'total.eve.calls', 'total.night.calls' ))
```

Final features used for modeling data can be find in below image

```
> str(train)
'data.frame':   3333 obs. of  10 variables:
 $ international.plan          : Factor w/ 2 levels "1","2": 1 1 1 2 2 2 1 2 1 2 ...
 $ voice.mail.plan             : Factor w/ 2 levels "1","2": 2 2 1 1 1 1 2 1 1 2 ...
 $ number.vmail.messages       : num  1.238 1.311 -0.591 -0.591 -0.591 ...
 $ total.day.charge            : num  1.617 -0.349 1.205 2.268 -0.251 ...
 $ total.eve.charge            : num  -0.07686 -0.11524 -1.63162 0.00472 -1.07738 ...
 $ total.night.charge          : num  0.9054 1.1074 -0.7883 -0.0814 -0.2879 ...
 $ total.intl.calls            : num  -0.615 -0.615 0.352 1.318 -0.615 ...
 $ total.intl.charge           : num  -0.1121 1.325 0.7358 -1.4341 -0.0689 ...
 $ number.customer.service.calls: num  -0.295 -0.295 -1.364 0.774 1.842 ...
 $ Churn                       : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
> |
```

5. Feature scaling:

Feature scaling enables model to converge quickly. It is a step of Data processing which is applied to independent variables or features of data. It basically helps to normalize the data within a particular range. It also helps in speeding up the calculations in an algorithm.

There are two ways of feature scaling:

1. Standardization
2. Normalization

Standardization – Standardization deals with min and range variables.

```
standardizeData = function(data){
  numericNames = sapply(data, is.numeric)
  numeric_data = data[, numericNames]
  for (colname in colnames(numeric_data)) {
    data[,colname] = (data[,colname] - min(data[,colname]))/(max(data[,colname])-min(data[,colname]))
  }
  return(data)
}
```

Normalization – Normalization deals with standard deviation and mean. Normalization works better for normally distributed data.

```
normalizeData = function(data){
  numericNames = sapply(data, is.numeric)
  numeric_data = data[, numericNames]
  for (i in colnames(numeric_data)) {
    data[,i]=(data[,i]-mean(data[,i]))/(sd(data[,i]))
  }
  return(data)
}
```

Hence, we are going to apt normalization for scaling.

Summary of scaled data:

```
> summary(train)
 international.plan voice.mail.plan number.vmail.messages total.day.charge   total.eve.charge    total.night.charge
 1:3010             1:2411          Min.   :-0.5914       Min.   :-2.74986   Min.   :-2.790497   Min.   :-2.789487
 2: 323             2: 922          1st Qu.:-0.5914       1st Qu.:-0.67691   1st Qu.:-0.693483   1st Qu.:-0.687275
                                    Median :-0.5914       Median :-0.01013   Median : 0.004722   Median : 0.005813
                                    Mean   : 0.0000       Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.000000
                                    3rd Qu.: 0.8718       3rd Qu.: 0.68457   3rd Qu.: 0.688531   3rd Qu.: 0.698900
                                    Max.   : 3.0665       Max.   : 2.74858   Max.   : 2.771148   Max.   : 2.764393
 total.intl.calls  total.intl.charge   number.customer.service.calls Churn
 Min.   :-2.0651   Min.   :-2.713051   Min.   :-1.3635               1:2850
 1st Qu.:-0.6151   1st Qu.:-0.658121   1st Qu.:-0.2950               2: 483
 Median :-0.1318   Median : 0.002906   Median :-0.2950
 Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000
 3rd Qu.: 0.3515   3rd Qu.: 0.663933   3rd Qu.: 0.7736
 Max.   : 2.7682   Max.   : 2.718863   Max.   : 1.8422
```

Selecting machine learning algorithms:

This section is the most important one, Here, we will try a couple of different ML algorithms in order to get an idea about which ML algorithm performs better. Also, we will finalize algorithms by comparing training accuracy and false negative ratio.

Here False Negative ratio is also an important metrics compared to others because predicting churning customer as not will lead to greater business loss.

By the time, we will definitely know that this particular problem is considered a classification problem. The algorithms that we are going to choose are as follows:

1. Decision tree
2. Random forest algorithm
3. Logistic regression

1. **Decision tree**

   Decision tree belongs to supervisor learning algorithms.
   By inferring rules from past data, it performs future prediction. To perform decision tree entropy is used as measure of splitting data with features.

   Let's take a look at the below code:

```
#Model selection
c50_Model = C5.0(Churn ~., train, trails = 100, rules = TRUE)
```

   Decision tree model will yields information about attribute usage towards prediction.

```
Attribute usage:

 96.22% total.day.charge
 87.58% international.plan
 78.79% total.eve.charge
 65.92% total.intl.calls
 65.71% total.intl.charge
 10.08% voice.mail.plan
  4.05% total.night.charge
```

As we see, almost all variables contribute significant level towards predicting target variable. So we have selected right features at the feature selection.

It also prints all the rules because we set “rules” argument to true in the model argument. Example rule:

```
Rule 1: (160/4, lift 1.1)
        international.plan = 1
        voice.mail.plan = 2
        total.day.charge > 0.8208318
        ->  class 1  [0.969]
```

Totally model infers 12 rules with training data. These data will be used for predicting test dataset. Now we have build a model. Let's predict the test data

```
Dt_prediction = predict(c50_Model, test[,-10],type = "class")
```

10th variable is target variable. So other than that, every thing will be fed into predict method. Type refers to output type whether it is an class or probability.

Build confusion matrix.

```
Dt_table = table(test[,10],Dt_prediction)
confusionMatrix(Dt_table)
```

```
Confusion Matrix and Statistics

   Dt_prediction
      1    2
  1 1436    7
  2  122  102

               Accuracy : 0.9226
                 95% CI : (0.9087, 0.935)
    No Information Rate : 0.9346
    P-Value [Acc > NIR] : 0.9767

                  Kappa : 0.5752
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9217
            Specificity : 0.9358
         Pos Pred Value : 0.9951
         Neg Pred Value : 0.4554
             Prevalence : 0.9346
         Detection Rate : 0.8614
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9287

       'Positive' Class : 1
```

Here key metrics to consider is below:

Accuracy: 92%
False Negative Ratio: 122/(122+102) = 0.544642857
False Positive Ratio: 7/(1436+7) = 0.00485100485

We have achieved some considerable amount of accuracy. But FNR is quite higher. Let's put these metrics of Decision tree aside to compare with other models metrics.

2. Logistic Regression

Logistic regression is an statistical model which can be used classification problems.
Target variable for logistic regression

- binomial
- ordinal
- multinomial

Here our target variable is binomial.
We already meet all our assumptions for using logistic regression.
Let's take a look at the below code:

```
logit_model = glm(Churn ~., data = train, family = "binomial",control = list(maxit = 1000))
```

Summary of the model:

```
Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -1.73083    0.15162 -11.416  < 2e-16 ***
international.plan2          1.82221    0.13648  13.351  < 2e-16 ***
voice.mail.plan2            -1.97492    0.53601  -3.684 0.000229 ***
number.vmail.messages        0.48882    0.23068   2.119 0.034084 *
total.day.charge             0.56940    0.05509  10.335  < 2e-16 ***
total.eve.charge             0.26704    0.05396   4.949 7.46e-07 ***
total.night.charge           0.17380    0.05310   3.273 0.001065 **
total.intl.calls            -0.23522    0.05540  -4.246 2.18e-05 ***
total.intl.charge            0.16992    0.05384   3.156 0.001600 **
number.customer.service.calls -0.10850    0.05310  -2.043 0.041010 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2758.3  on 3332  degrees of freedom
Residual deviance: 2363.8  on 3323  degrees of freedom
AIC: 2383.8
```

It outputs p value for all the predictors. If we interpret above image, then all the predictors are below 0.05 which means it is highly correlated with target variable.

Now, predict the test data with model we build.
Build confusion matrix with predicted and actual values

```
logit_prediction = predict(logit_model, newdata = test[, c(-10)], type = "response")

logit_out = ifelse(logit_prediction > 0.5, 2, 1)
logit_matrix = table(test[,10],logit_out)
confusionMatrix(logit_matrix)
```

```
Confusion Matrix and Statistics

          logit_out
              1    2
        1 1422   21
        2  195   29

                     Accuracy : 0.8704
                       95% CI : (0.8534, 0.8862)
          No Information Rate : 0.97
          P-Value [Acc > NIR] : 1

                        Kappa : 0.171
       Mcnemar's Test P-Value : <2e-16

                  Sensitivity : 0.8794
                  Specificity : 0.5800
               Pos Pred Value : 0.9854
               Neg Pred Value : 0.1295
                   Prevalence : 0.9700
               Detection Rate : 0.8530
         Detection Prevalence : 0.8656
            Balanced Accuracy : 0.7297

             'Positive' Class : 1
```

Here metrics to compare:

Accuracy: 87%
False Negative Ratio: 195/(195+29) = 0.87053
False Positive Ratio: 7/(1422+21) = 0.01455

3. Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes. It is suitable for classification problems.

Let's take a look at the below code:

```
RandomF_Model = randomForest(Churn ~., com[1:3333,], importance = TRUE,ntree = 500)
```

importance - importance parameter should be set to true, for listing variable importance.
ntree – Number of trees to be constructed.

Rules will be generated by inferring training data. Those rules will be used for predicting future data. Create confusion matrix with predicted and actual values

```
Confusion Matrix and Statistics

   RF_Predictions
       1    2
  1 1442    1
  2  108  116

                Accuracy : 0.9346
                  95% CI : (0.9217, 0.946)
    No Information Rate : 0.9298
    P-Value [Acc > NIR] : 0.2381

                   Kappa : 0.6479
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9303
            Specificity : 0.9915
         Pos Pred Value : 0.9993
         Neg Pred Value : 0.5179
             Prevalence : 0.9298
         Detection Rate : 0.8650
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.9609

       'Positive' Class : 1
```

Here metrics to compare:

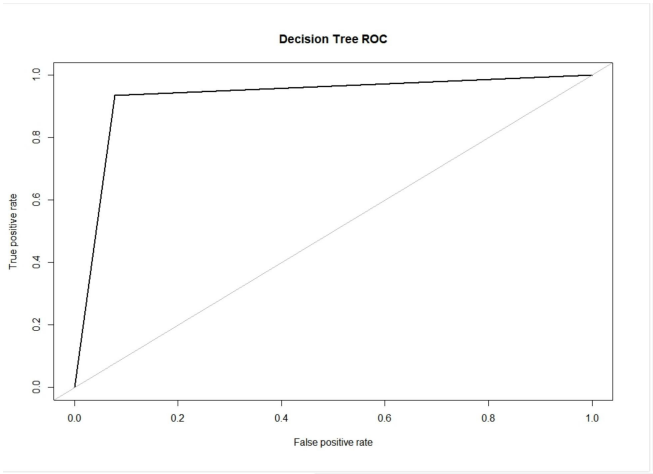
Accuracy: 93%
False Negative Ratio: 108/(108+116) = 0.48214
False Positive Ratio: 1/(1422+1) ~ 0

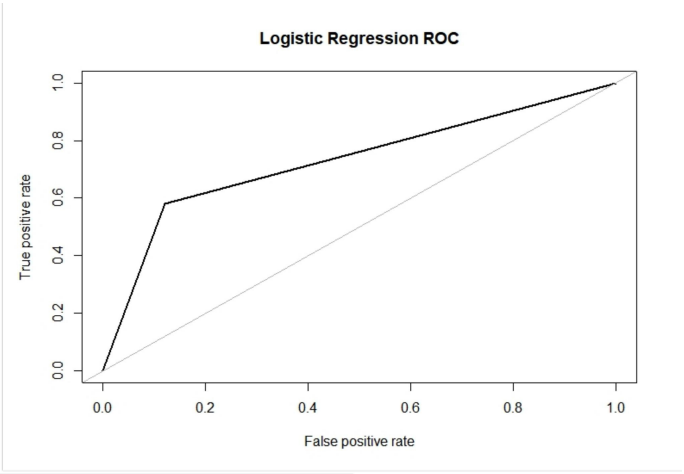
Let's build a table of all algorithm's metrics:

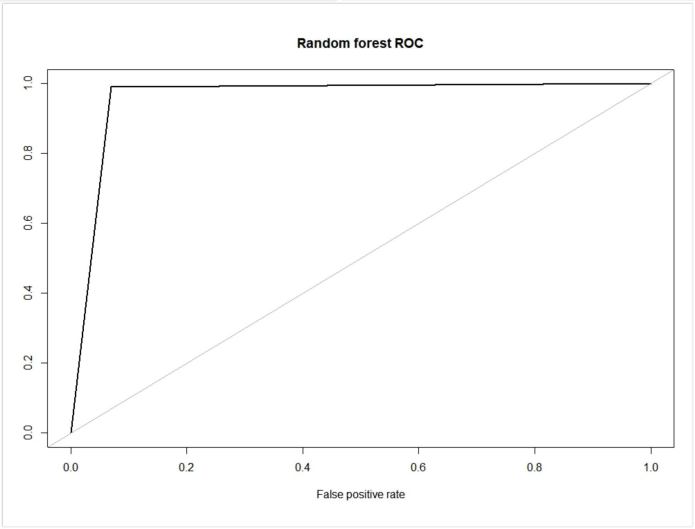
Algorithm	Accuracy	False Negative Ratio	False Positive Ratio
Decision Tree	92.00%	0.54	0.0049
Logistic Regression	87.00%	0.87	0.01
Random Forest	93.00%	0.48	~0

This table clearly explains that random forest is performing better than other algorithms in all metrics. So we can finalize random forest as our machine learning algorithm.

ROC curve has been shown in below. It clearly explains that Random forest was performing better than other algorithms while considering FPR.







Problem with current results:

Even though random forest yields greater accuracy, FNR is 0.48%. It is quite higher than standard. This is because "no" class in target variable has much more less observations than "yes" class. As calculated previously,

False class has 2850 observations out of 3333 - 86%

True class has 483 observations out of 3333 - 14%

Imbalanced training set:

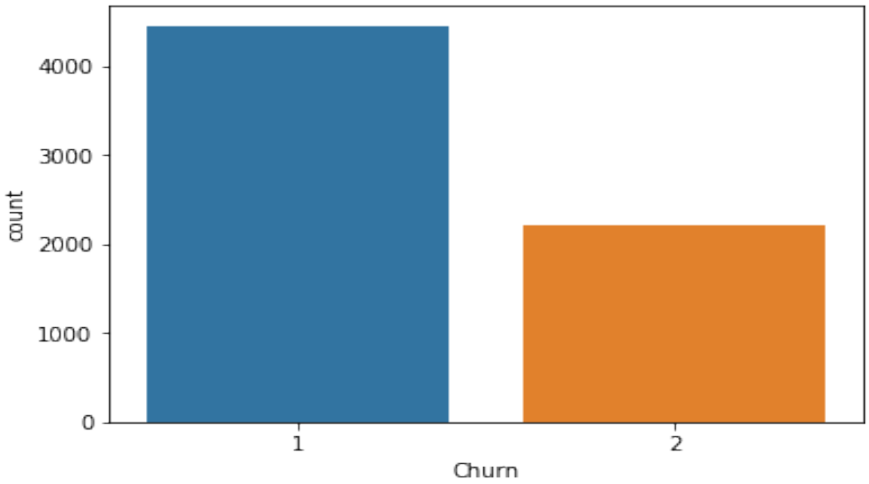
Imbalanced training set has been handled using many methods like over-sampling, under-sampling etc. Here synthetic data generation for deficient class has been done using random over sampling method.

```
#Synthetic data generation
library(ROSE)
generated = ROSE(Churn ~., data=train,seed = 3)$data
train = rbind(train, generated)
```

After generation of data, target class proportion will be below:

```
> table(train$Churn)

   1    2
4455 2211
```



As we see above,

False class has 4455 observations out of 6666 - 67%
True class has 2211 observations out of 6666 - 33%

Now we considerably increased data proportion of deficient classes.

Data generation should be done only after removing all the correlated variables. Outlier detection and feature scaling should be executed after data generation. Then, Important variable has been fetched using Random forest and removed less considerable features.

As we already selected our machine learning algorithm as Random forest, same will be executed for this synthetic data too.

Results:

Accuracy: 88 % - 93 %
FNR: 0.13 - 0.18
FPR: 0.12 – 0.04

As data has been generated in random, accuracy varies from 88 to 93%. FNR varies from 0.13 to 0.20 and FPR varies from 0.12 to 0.04

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

   RF_Predictions
      1    2
  1 1273  170
  2   30  194

               Accuracy : 0.88
                 95% CI : (0.8635, 0.8952)
    No Information Rate : 0.7816
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.592
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9770
            Specificity : 0.5330
         Pos Pred Value : 0.8822
         Neg Pred Value : 0.8661
             Prevalence : 0.7816
         Detection Rate : 0.7636
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.7550
```

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

   RF_Predictions
      1    2
  1 1380   63
  2   45  179

               Accuracy : 0.9352
                 95% CI : (0.9223, 0.9466)
    No Information Rate : 0.8548
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7306
 Mcnemar's Test P-Value : 0.1019

            Sensitivity : 0.9684
            Specificity : 0.7397
         Pos Pred Value : 0.9563
         Neg Pred Value : 0.7991
             Prevalence : 0.8548
         Detection Rate : 0.8278
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.8540

       'Positive' Class : 1
```

Since FNR would produce major business loss, As far as this dataset has been concerned, Our random forest model feeded with synthetic data can be finalized as best fit model.

*Note: Code for Synthetically generated model has been attached as separate file

Attachments:

1. **Churn prod . R** - Contains all the steps described above except visualization and synthetic data generated model.
2. **Synthetic churn prod . R** - Contains synthetic data generation with all corresponding steps for modeling
3. **Direct load Synthetic . R –** Gets generated "Synthetic data.csv" as input, code has been changed accordingly
4. **Churn Visualization . R** - Contains all the visualization codes in R. Some codes saves files into hard disk.
5. **Utility Functions . R** - Contains helper functions used in all above files.
6. **Churn.ipynb** - Contains all the equivalent codes for R in python. Contains codes for all the explanations in the report
7. **train.csv and test.csv** – Train and test data given
8. **Synthetic data.csv** – Synthetic data generated using training data
9. report images.zip –
    1. Contains all the images used in the report.
    2. Contains individual distribution image with and without outliers
    3. Contains individual ROC images
    4. Contains all the images(in "facts" folder) that has been used for deriving business insights in feature selection section.