Project Report on Customer Churn Reduction

Made By: Sakshi Rajput

16 March, 2019

Chapter 1

Introduction

Customer churn refers to when a customer ceases his or her relationship with a company. Businesses typically treat a customer as churned once a particular amount of time has elapsed since the customer's last interaction with the site or service. The full cost of customer churn includes both lost revenue and the marketing costs involved with replacing those customers with new ones. Reduction customer churn is important because cost of acquiring a new customer is higher than retaining an existing one. Reducing customer churn is a key business goal of every business. This case is related to telecom industry where particular organizations want to know that for given certain parameters whether a person will churn or not.

1.1 Problem Statement

In the given problem, data from a telecom company is provided that contains 20 variables like services, charges and calls made by customer. The data set provided also consist of area code and phone numbers of the customers. The target variable is "Churn" which tells whether the customer has churned out or not.

1.2 Data

The dataset provided contains two data sets (Test.csv and Train.csv) that contains 20 predective variables and 1 target variables.

. 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

Size of Dataset Provided :-

Train.csv = 3333 rows, 21 Columns

Test.csv = 1667 rows, 21 Columns

Following are the top 5 rows of the dataset:

State	Account Length	Area code	Phone no	International plans	Voice mail plans	Number vmail messages
KS	128	415	382-4657	no	yes	25
OH	107	415	371-7191	no	yes	26
NJ	137	415	358-1921	no	no	0
OH	84	408	375-9999	yes	no	0
OK	75	415	330-6626	yes	no	0

Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes
265.1	110	45.07	197.4	99	16.78	244.7
161.6	123	27.47	195.5	103	16.62	254.4
243.4	114	41.38	121.2	110	10.3	162.6
299.4	71	50.9	61.9	88	5.26	196.9
166.7	113	28.34	148.3	122	12.61	186.9

total night calls	total night charge	total night minutes	total intl calls	total intl charge	number customer service calls	Churn
91	11.01	10	3	2.7	1	False
103	11.45	13.7	3	3.7	1	False
104	7.32	12.2	5	3.29	0	False
89	8.86	6.6	7	1.78	2	False
121	8.41	10.1	3	2.73	3	False

Churn is the Target Variable.

Chapter 2

Methodology

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1 Exploratory Data Analysis (EDA)

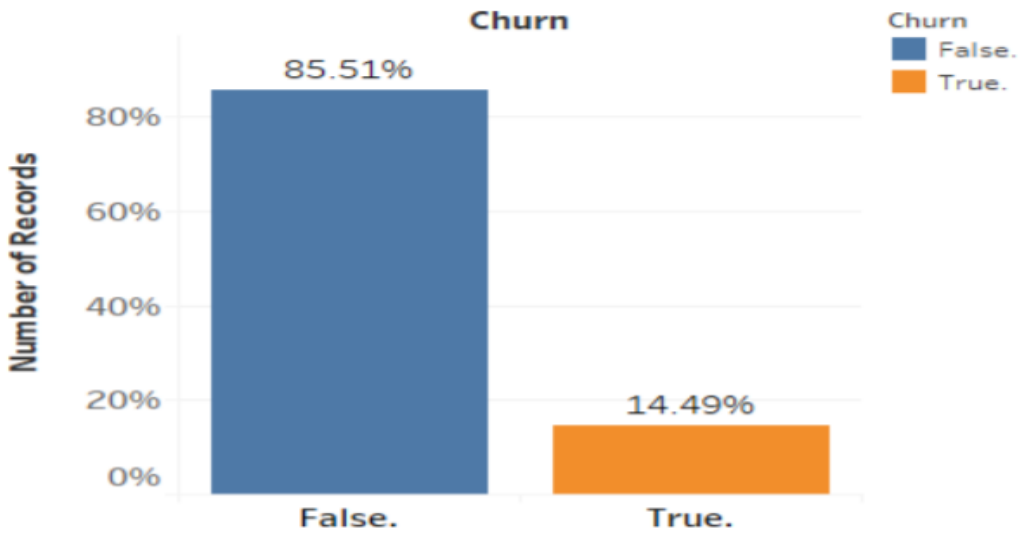
Exploratory Data Analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

2.1.1 Target Variable - Churn

Our target variable has two categories which include True and False values.

True = Customer will move or churn out.

False = Customer won't move



We can clearly see that our data is highly imbalanced. The occurrence of false is higher than True. Thereare2850(85.51%)customerswhochurnoutand483(14.49%) customersretained

2.1.2 Uniqueness in Variable

The following represents the number of unique values each variable has:

Variables	**Unique Counts**
state	51
Account length	212
Area code	3
Phone number	3333
International plan	2
Voice mail plan	2
Number vmail messages	46
Total day minutes	1667
Total day calls	119
Total day charge	1667
Total eve minutes	1611
Total eve calls	123
Total eve charge	1440
Total night minutes	1591
Total night calls	120
Total night charge	933
Total intl minutes	162
Total intl calls	21
Total intl charge	162
Number customer service calls	10
churn	2

From the above data we can conclude that :

- area code has only 3 values,so convert it to categorical variable.
- phone number has 3333, which makes it a full unique variable. Whcich can be removed because it doesn’t contain any important information

2.1.3 Missing Value Analysis

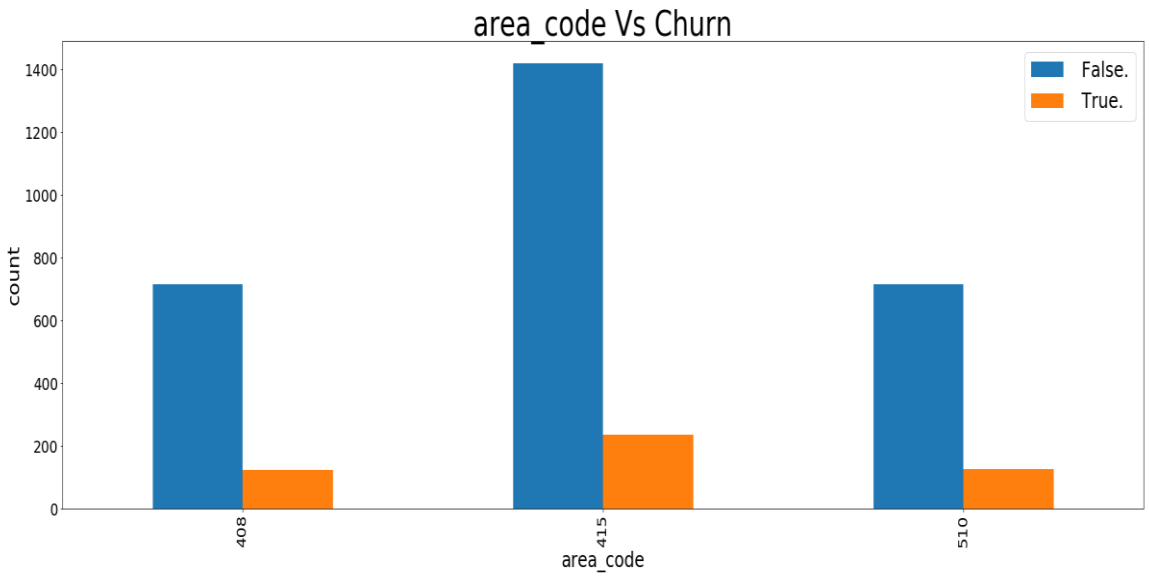
The following represents the number of missing values for each variable has:

Variables	**Unique Counts**
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 we can see that there are no missing values present in the dataset so, we need not apply the missing value analysis on the dataset

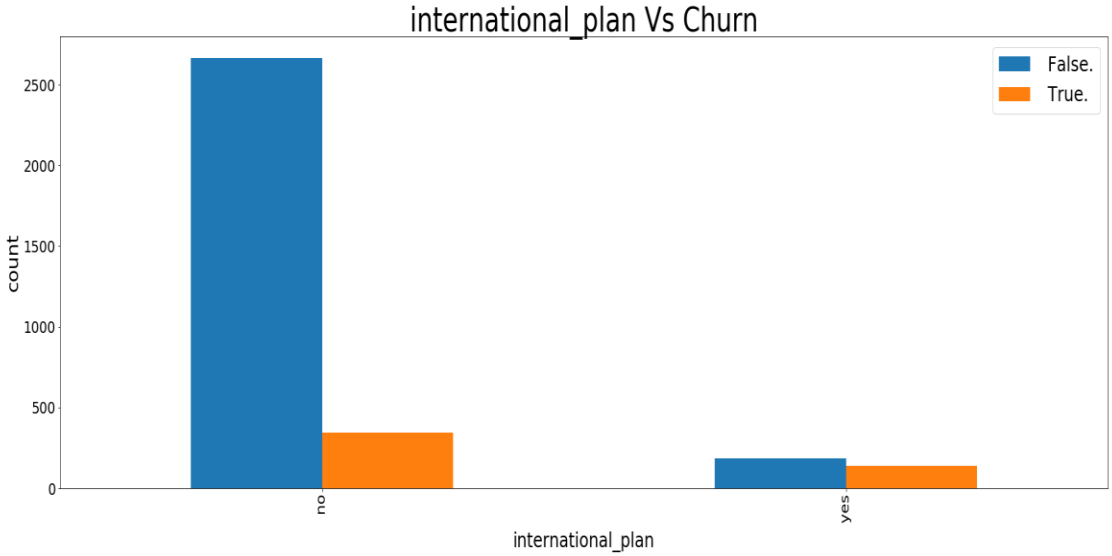
2.1.4 Churning of Customers according to different variables

1. Area code



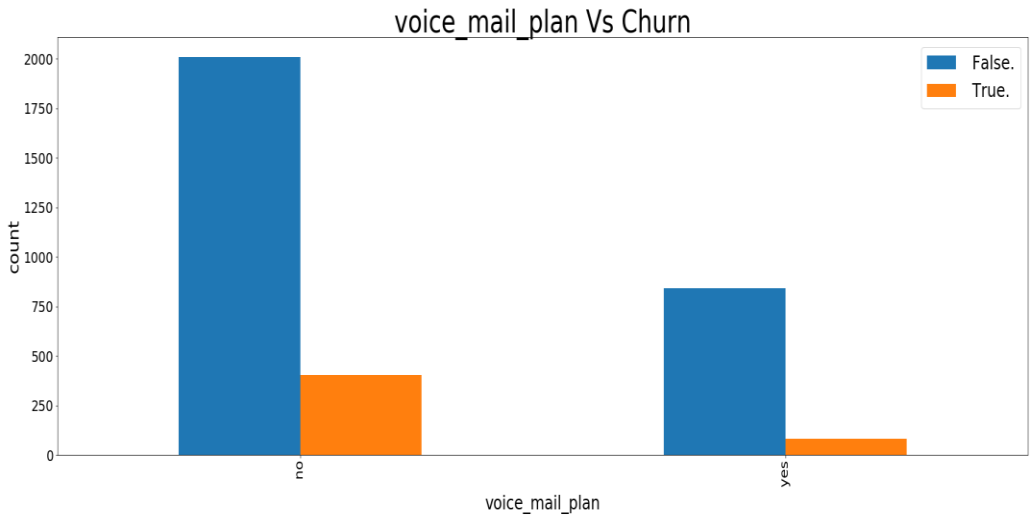
As we can see from the above graph that most people who churned are from area 415. That is an useful insight as company can use this to analyze what causes this high churn rate in that area.

2. International plan

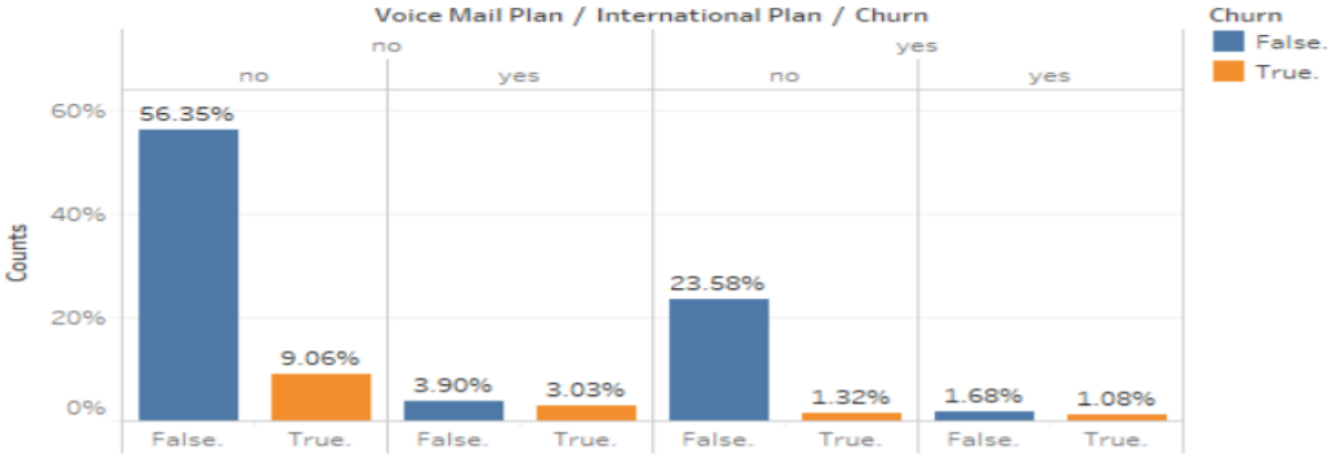


From the above graph we inferred that customers with international plans churn more than the customers who do not have international plans. That is a useful insight as company can use this to analyze what causes this high churn rate in international plans. May be the company need to revise their international plans.

3. Voice mail plan

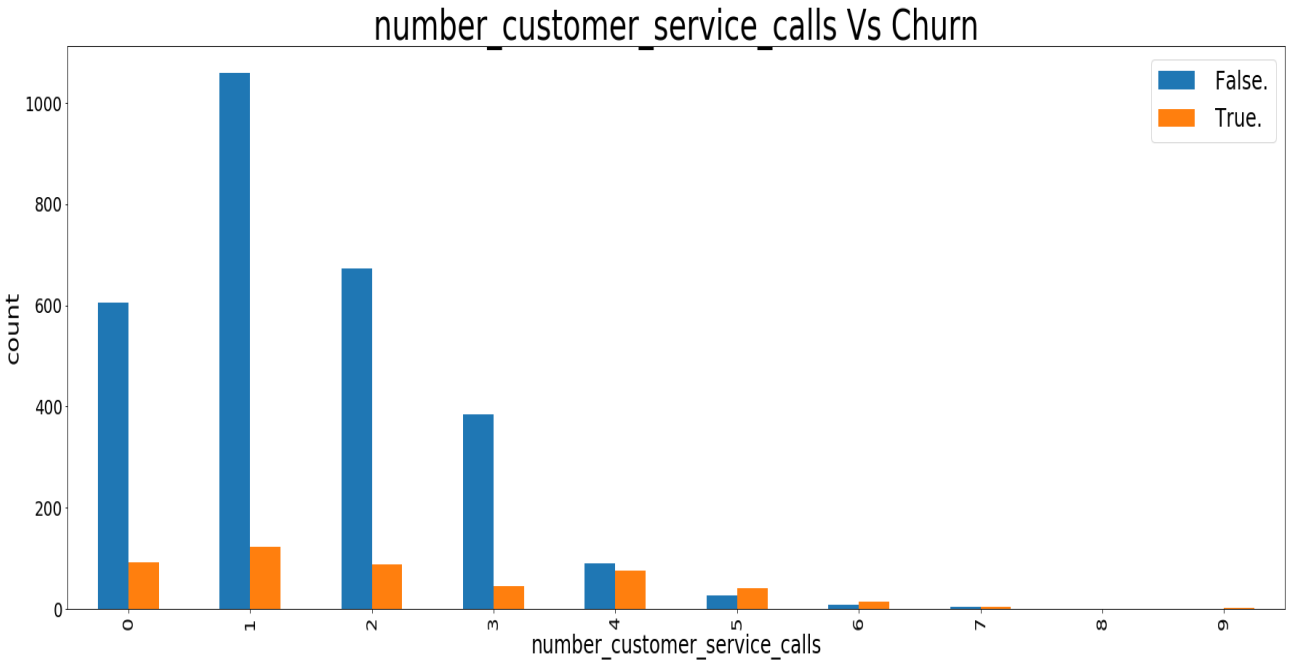


From the above graph we inferred that customers with voice mail plans churn more than the customers who do not have voice mail plans. That is a useful insight as company can use this to analyze what causes this high churn rate in voice mail plans. May be the company need to revise their voice mail plans.



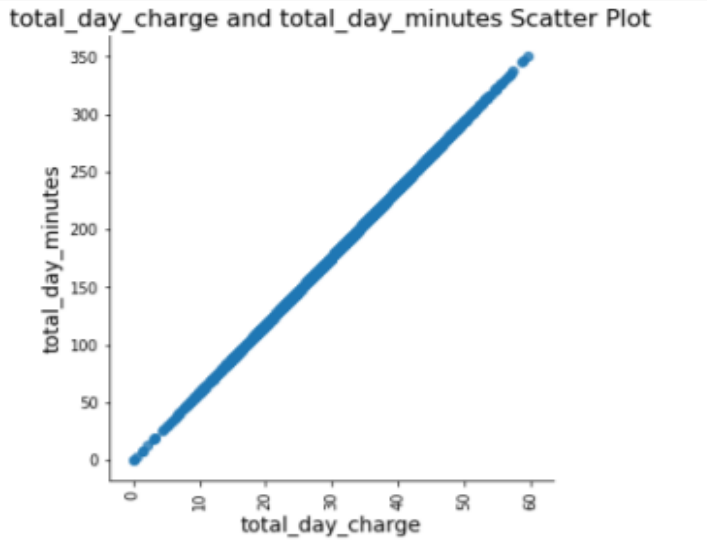
Churn rate for Customer neither having voice mail plan nor international plan is 9.06%. Churning rate for customer having International plan but don't have voice mail plan is 3.03% out of 6.93% customers. Churning of customer having both voice mail plan&international plan is 1.08%out of 2.76%

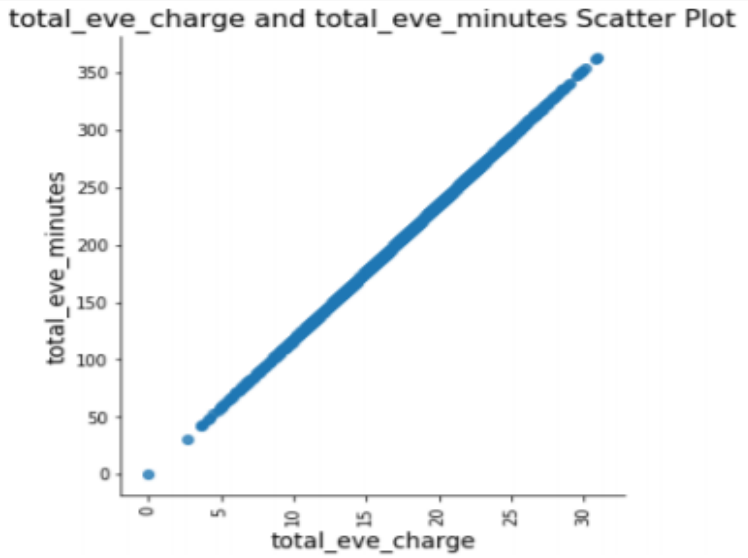
4. Customer service calls impact on churn

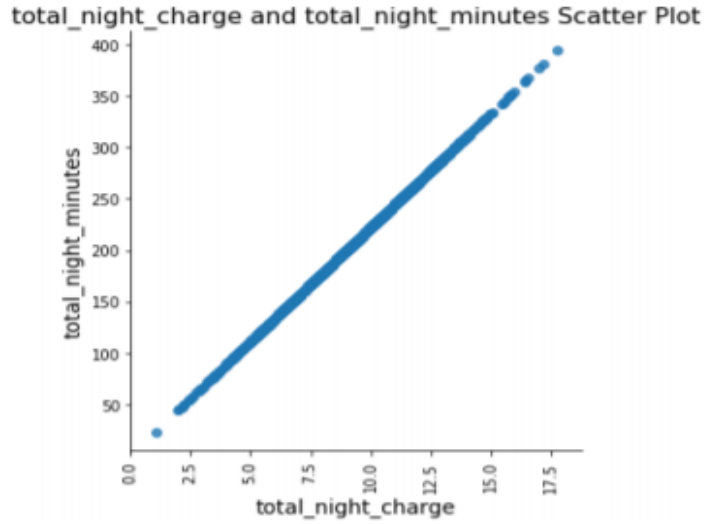


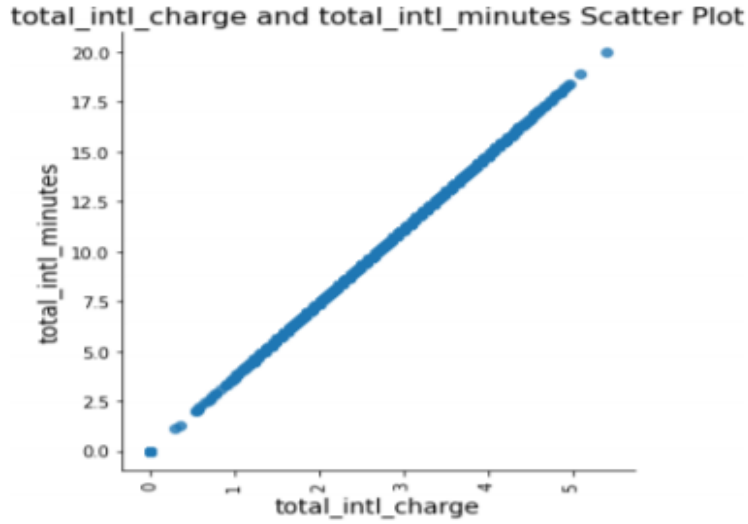
From the above graph we inferred that Churn rate increasing with increase in customer service call frequency. That is a useful insight as company can use this to analyze what causes this high churn rate in voice mail plans. May be the company need to revise their voice mail plans.

Correlation between variables:









Some of the Variable are highly correlated :-

- Total_day_charge & total_day_minute
- Total_eve_charge & total_eve_minute
- Total_night_charge & total_night_minute
- Total_intl_charge & total_intl_minute

2.1.5 Feature Selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for four reasons:

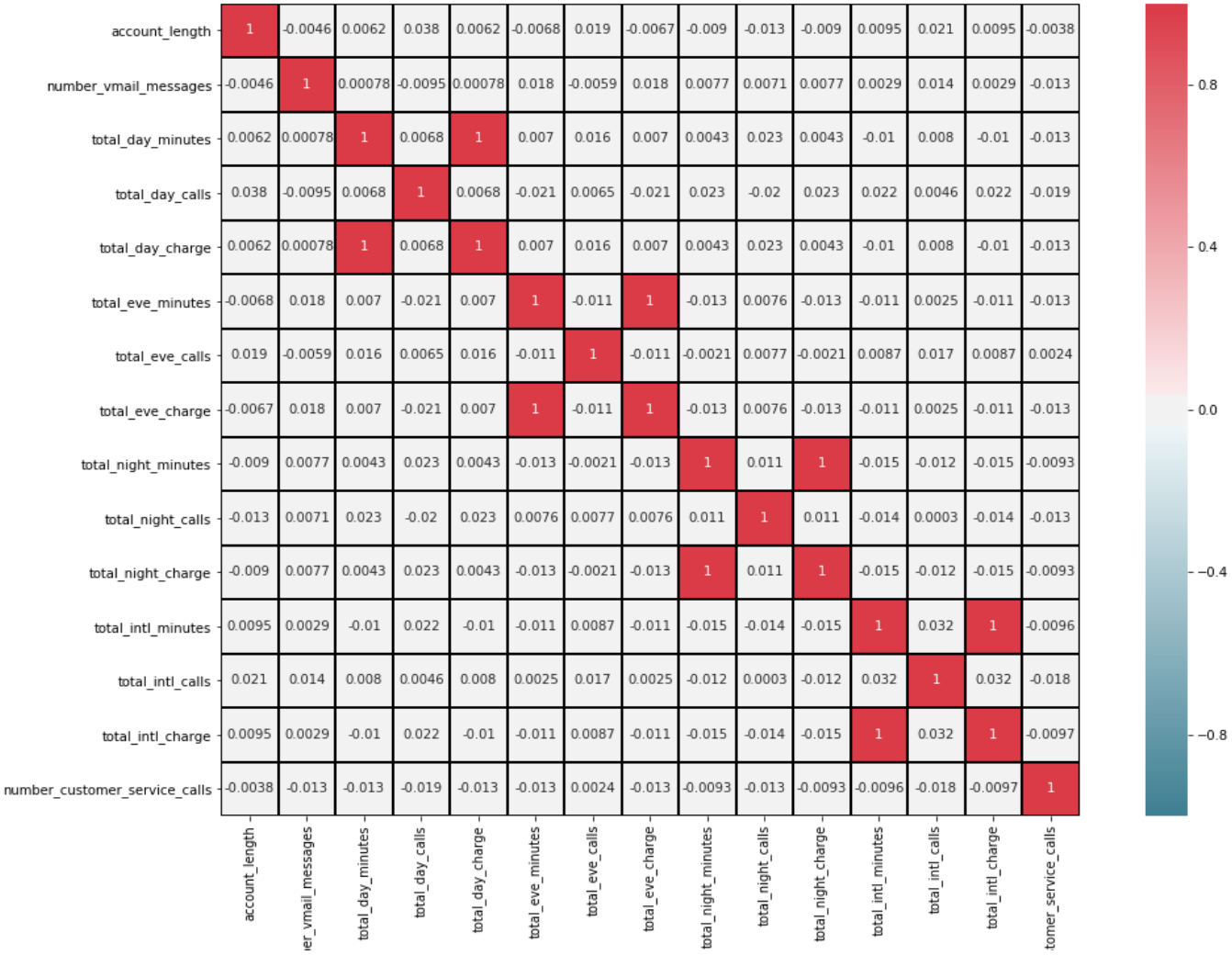
- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

For Continuous variable using Correlation Matrix

For categorical variable using Chi Square test

★ **Correlation Analysis**

Correlation is used to test relationships between quantitative variables or categorical variables. In other words, it's a measure of how things are related. We cannot use all the features because some features may be carrying the same information or irrelevant information which can increase overhead. To reduce overhead, we adopt feature selection technique to extract meaningful features out of data. This in turn helps us to avoid the problem of multi collinearity.



Some of the Variable are highly correlated :-

- Total_day_charge & total_day_minute
- Total_eve_charge & total_eve_minute
- Total_night_charge & total_night_minute
- Total_intl_charge & total_intl_minute

Chi Square test – Categorical Variables

The chi-squared test is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories.

In the standard applications of this test, the observations are classified into mutually exclusive classes, and there is some theory, or say null hypothesis, which gives the probability that any observation falls into the corresponding class. The purpose of the test is to evaluate how likely the observations that are made would be, assuming the null hypothesis is true.

Variables	P-value
State	0.002296221552011188
Area code	0.9150556960243712
International plan	2.4931077033159556e-50
Voice mail plan	5.15063965903898e-09

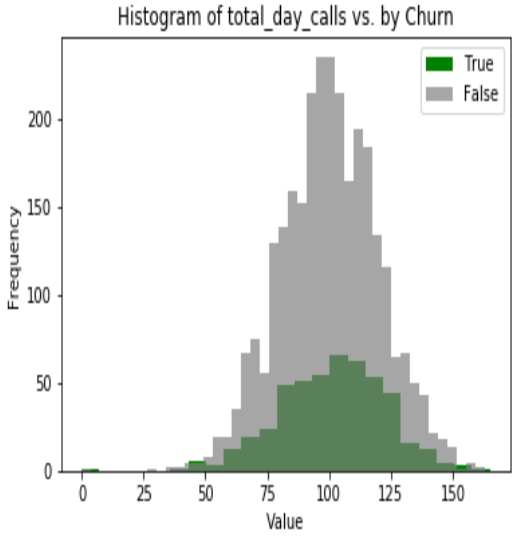
★ **Removing redundant variables**

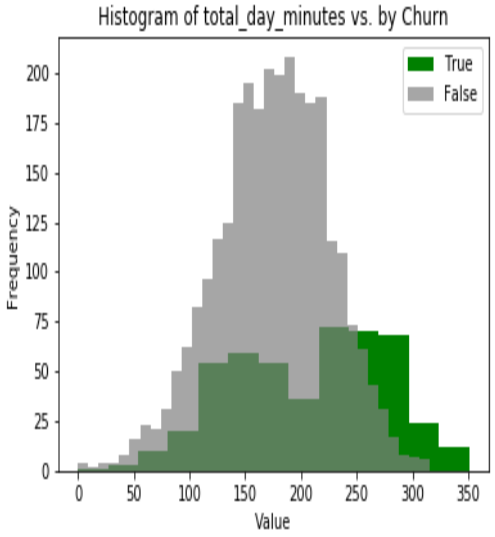
- 'state'
- 'total_day_charge'
- 'total_eve_charge'
- 'total_night_charge'
- 'total_intl_charge'

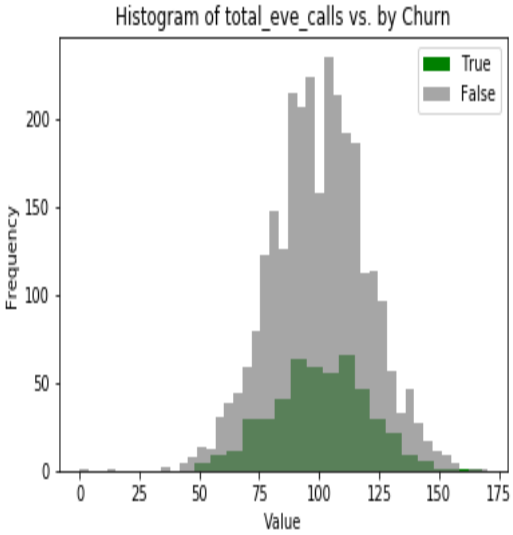
2.1.6 Feature Scaling

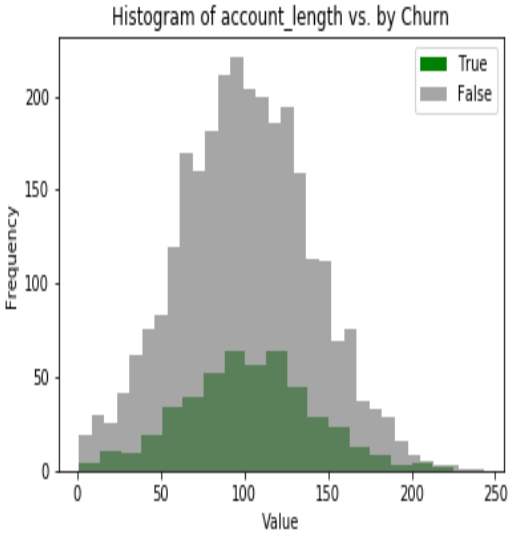
It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.

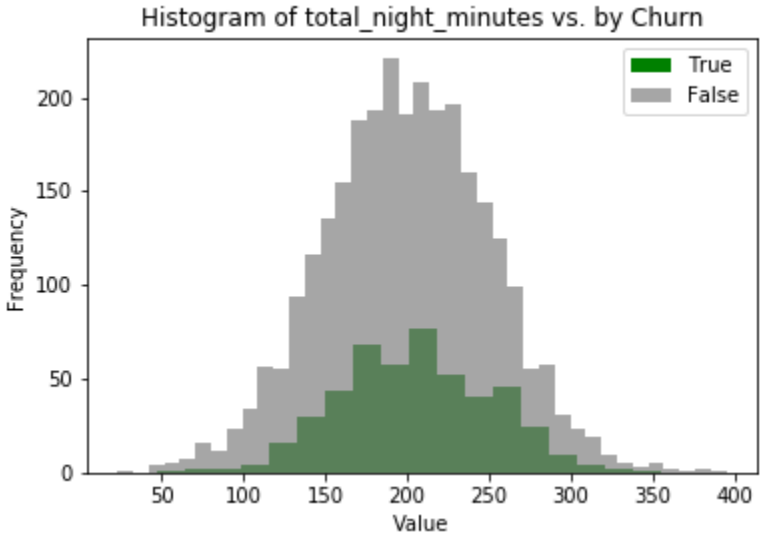
Here is a bar graph visualization of categorical variables :


Histogram of total_day_calls vs. by Churn
True
False
Frequency
Value
0
25
50
75
100
125
150
200


Histogram of total_day_minutes vs. by Churn
True
False
Frequency
Value
0
25
50
75
100
125
150
175
200
250
300
350







We can see from above histograms, most of our continuous data is uniformly distributed.

We will use Standardization \ Z - Score here.

Standardization replaces the values by their Z scores.

$$z = \frac{x - \mu}{\sigma}$$

account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	totaleve minutes
0.676388	1	0	1	1.234697	1.566532	0.476572	-0.070599
0.149043	1	0	1	1.307752	-0.333688	1.124334	-0.108064
0.902393	1	0	0	-0.591671	1.168128	0.675883	-1.573147
-0.428526	0	1	0	-0.591671	2.196267	-1.466716	-2.742453
-0.654531	1	1	0	-0.591671	-0.240054	0.626055	-1.038776

total night minutes	total night calls	total intl minutes	total intl calls	number customer service calls	Churn
0.866613	-0.465425	-0.084995	-0.601105	-0.427868	0
1.058412	0.147802	1.240296	-0.601105	-0.427868	0
-0.756756	0.198905	0.703015	0.211502	-1.18804	0
-0.078539	-0.567629	-1.302831	1.024109	0.332305	0
-0.27627	1.067643	-0.049177	-0.601105	1.092477	0

2.1.7 Sampling

Under sampling we will divide train data we have into train test split.

In Python we have used train_test_split() for sampling the train.csv data in to train and validation data.

In R we use createDataPartition() for randomly chosen values from each class.

Both methods using stratified sampling technique to cut the data into train and validation set. Our target variable class is imbalanced and after split of data in our train set we get

#False True

1881 319

SMOTE Oversampling (Python)

Before : False = 1895

True = 338

After : False = 1895

True = 1895

ROSE Oversampling (R)

Before : False = 1881

True = 319

After : False = 1101

True = 1019

Now, all the pre processing is done on the data. Our Data is all set to go into the machine learning model.

Chapter 3

Modeling

After a thorough preprocessing we will use some regression models on our processed data to predict the target variable. Following are the models which we have built –

Before evaluating the final model our of our all model let's get a brief about classification matrix.

1. **Accuracy** : the proportion of the total number of predictions that were correct.
2. **Sensitivity or Recall or True Positive Rate** : the proportion of actual positive cases which are correctly identified.
3. **True Negative Rate** : the proportion of actual negative cases which are correctly identified.
4. **False Positive Rate (Type –I error):** False positive, commonly called a "false alarm", is a result that indicates a given condition exists, when it does not .
5. **False Negative Rate(Type–II error)**: false negative, is a test result that indicates that a condition does not hold, while in fact it does.

Let's see False Negative and False Positive according to our problem statement. If in actual any customer is not churning and our model predict that he /she will churn. Then it's okay we can deal with it , like on prior we will start putting more effort on the customer so that he/she won't churn out. But in other case if our model predict that this particular customer won't churn out and in actual he will churn out, then there might be a big problem. Because in this scenario our client will lose some important clients. From both cases it's important to make a good trade off that my model would predict more accurate and have low false negative rate and also not much of false positive rate.

We will test four particular algorithms :

1- **Random Forest** :

It is a flexible, easy to use machine learning algorithm that produces a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks.

Parameter	R output	Python output
Accuracy	86.2312444836717	95.09
FNR	17.6829268292683	17.93
FPR	13.1062951496388	2.93
TPR	51.5267175572519	82.07
TNR	86.8937048503612	97.07

2- **Logistic Regression** :

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Parameter	R output	Python output
Accuracy	78.1994704324801	78.18
FNR	28.6585365853659	25.52
FPR	20.6398348813209	21.26
TPR	36.9085173501577	82.07
TNR	79.360165118679	78.74

3- **K- Nearest Neighbors** :

K-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors

Parameter	R output	Python output
Accuracy	78.8172992056487	78.09
FNR	46.3414634146341	31.03
FPR	16.9246646026832	20.52
TPR	34.9206349206349	68.97
TNR	83.0753353973168	79.48

4- **Naïve Bayes**

The Naïve Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods

Algorithm:

Bayes theorem provides a way of calculating the posterior probability,P(c|x),fromP(c),P(x),and P(x|c). Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

We will implement all four models on our preprocessed data in both Python and R in this chapter and then later on will select the final model

Parameter	R output	Python output
Accuracy	83.1421006178288	78.64
FNR	29.2682926829268	23.45
FPR	14.7574819401445	21.05
TPR	44.7876447876448	76.55
TNR	85.2425180598555	78.95

Final Model :- Random Forest

Results show that random forest fits best for our dataset out of all the tested models

Performance Tuning of Random forest

In performance tuning we apply different combination over our model and try to enhance the performance by applying different combination of hyper parameters. Random forest has a lot

of parameters so we will tune few of them which make a vast impact over accuracy and all result of model.

We have tuned random forest model in both R and Python using grid search and random search CV.

Python :

Hyper Parameter optimization with RandomSearch CV :

Under randomsearchCV we pass some parameters so that it will fit those parameters in the model and get theresult.

We get (ntree = 500, criterion = gini, max_features = auto)

Results after parameter optimization

CONFUSION MATRIX ------->>

	False	True
False	**927**	**28**
True	**26**	**119**

AUC -: 0.91

Classification paradox :------->>

Accuracy :- 95.09 %

Specificity // True Negative Rate :- 97.07 %

Sensitivity//True Positive Rate // Recall :- 82.07%

False Negative Rate :- 17.93 %

False Positive Rate :- 2.93 %

	precision	recall	f1-score	support
False	0.97	0.97	0.97	955
True	0.81	0.82	0.82	145
avg / total	0.95	0.95	0.95	1100

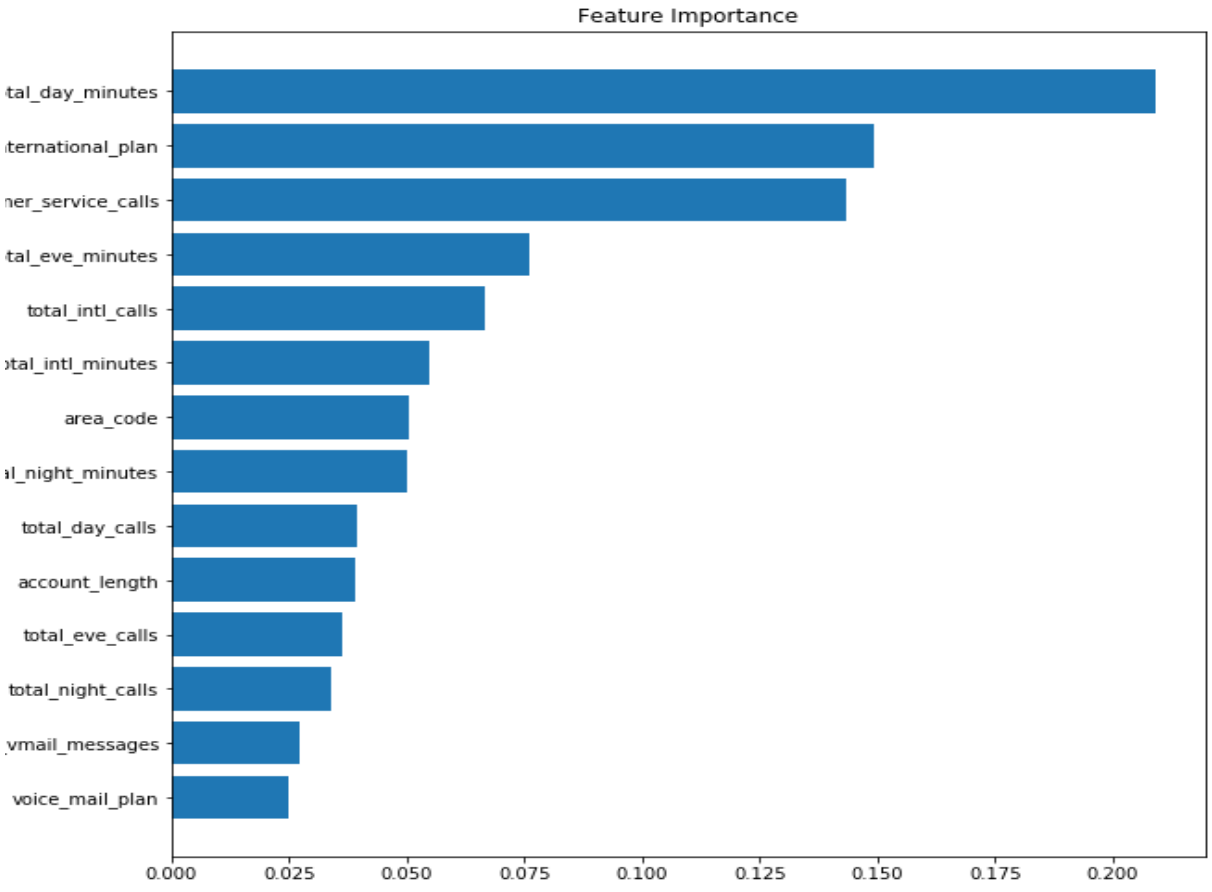
In R :

We use grid search CV to train our model and for searching hyper parameter tuning. We get (mtry = 4, ntree = 800)

CONFUSION MATRIX ------->>			Classification paradox :------->>
	False	True	Accuracy :- 86.5 %
False	844	125	False Negative Rate :- 17.07 %
True	28	136	False Positive Rate :- 12.90 %
AUC = 89.47			
95% CI		:	(0.8437, 0.8843)

No Information Rate	:	0.7696
P-Value[Acc > NIR]	:	4.453e-16
Kappa	:	0.5622
Mcnemar's Test P-Value	:	8.147e-15
Sensitivity	:	0.9679
Specificity	:	0.5211
Pos Pred Value	:	0.8710
Neg Pred Value	:	0.8293
Prevalence	:	0.7696
Detection Rate	:	0.7449
Detection Prevalence	:	0.8553
Balanced Accuracy	:	0.7445
'Positive' Class	:	1

Best Parameters we get after tuning the model :



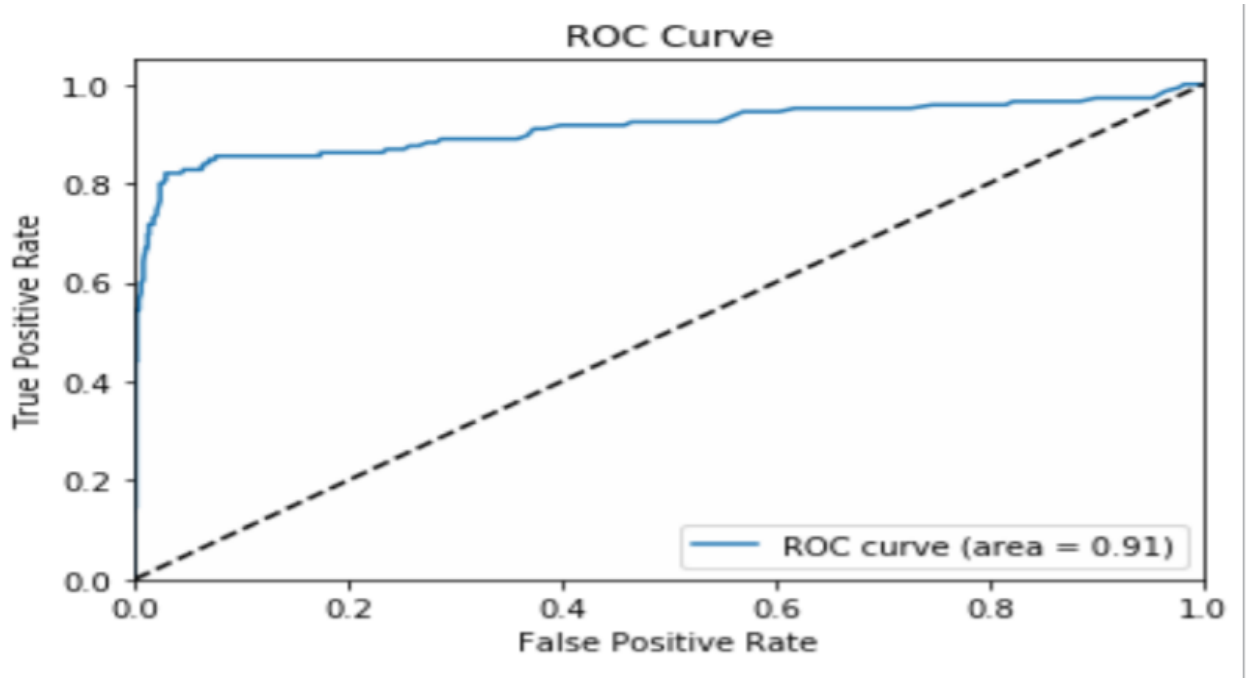
Variables :- total_dat_minutes, International_plan, number of customer service calls which makes majorimpact on performance of our predictions.

Cost Effective :

In case of cost effecting we can play a lot more with our model. As of till now we have used default threshold in R and Python both and get a good result. But we can enhance it more by making changes in the model threshold measure. While making any prediction that whether he/she will churn or not, there always single events occurs that either he will churn out or he will not. We can change that threshold so that then our model will become better. Overall still with imbalanced data our model is doing well.

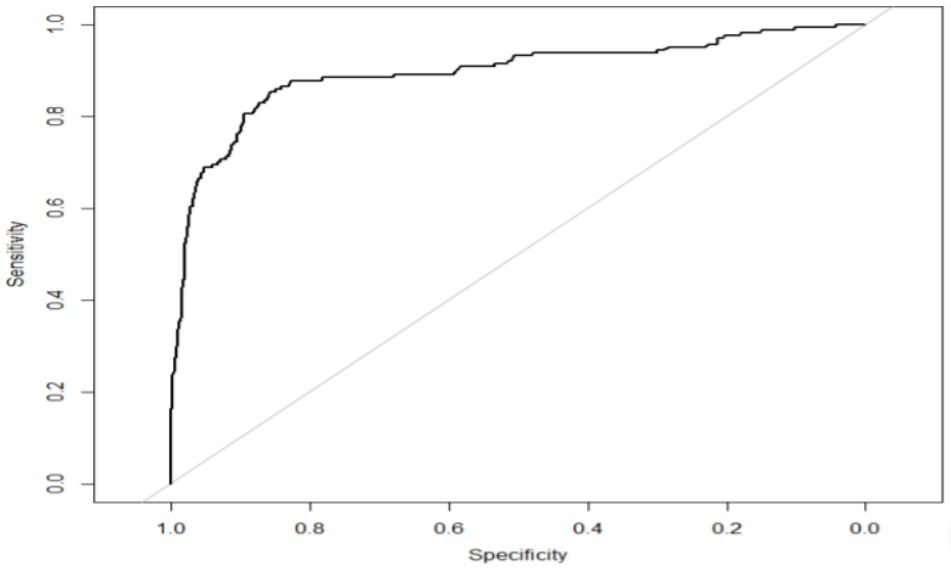
ROC curve:

Python model :-



R model :

Auc = 89.47



Final Test Data Prediction :

As in our test.csv file Churn target is given, Solet's Check the prediction accuracy on our final test data.

We have 1667 observations and 21 columns in test.csv

Python Result:-

CONFUSION MATRIX ------->>

	False	True
False	**1331**	**112**
True	**37**	**187**

AUC -: 0.92

Classification paradox :------->>

Accuracy :- 91.06 %

Specificity // True Negative Rate :- 92.24 %

Sensitivity//True Positive Rate // Recall :- 83.48%

False Negative Rate :- 16.52 %

False Positive Rate :- 7.76 %

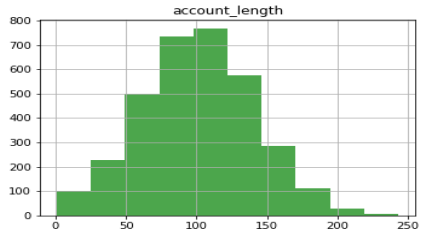
	precision	recall	f1-score	support
False	0.97	0.92	0.95	1443
True	0.63	0.83	0.72	224
avg / total	0.93	0.91	0.92	1667

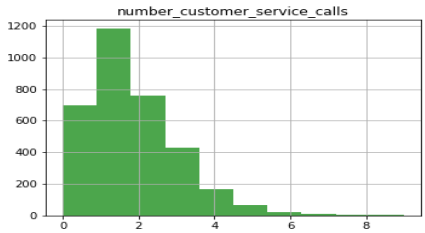
R result :

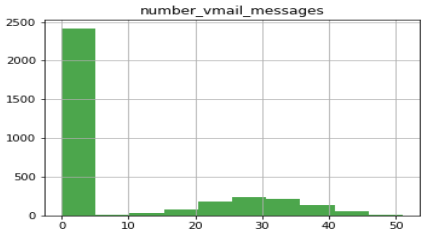
CONFUSION MATRIX ------->>			Classification paradox:------->> AUC =91.74
	False	True	Accuracy :- 85.92 %
False	1243	200	False Negative Rate :- 15.63 %
True	35	189	False Positive Rate :- 13.86 %

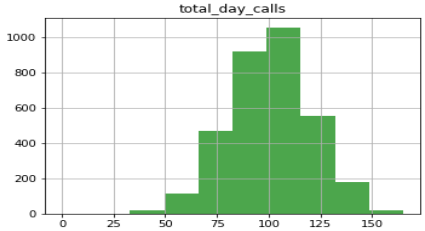
95% CI	:	(0.8414, 0.8754)
No Information Rate	:	0.7666
P-Value[Acc > NIR]	:	< 2.2e-16
Kappa	:	0.5378
Mcnemar's Test P-Value	:	< 2.2e-16
Sensitivity	:	0.9726
Specificity	:	0.4859
Pos Pred Value	:	0.8614
Neg Pred Value	:	0.8438
Prevalence	:	0.7666
Detection Rate	:	0.7457
Detection Prevalence	:	0.8656
Balanced Accuracy	:	0.7292
'Positive' Class	:	1

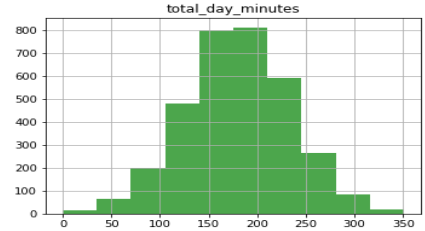
Appendix A: Extra Figures

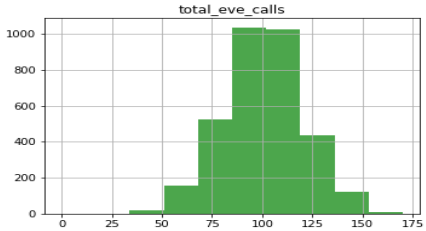


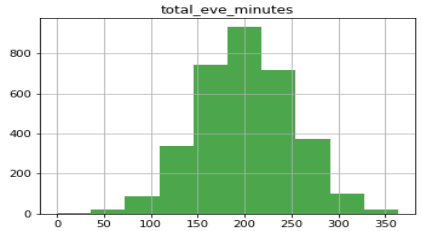


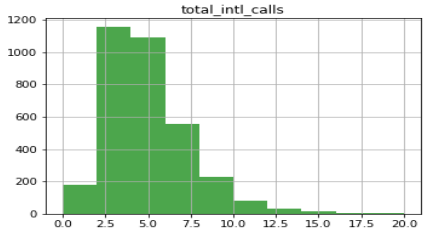


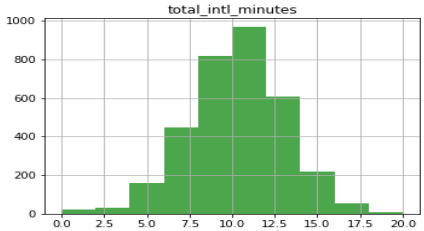


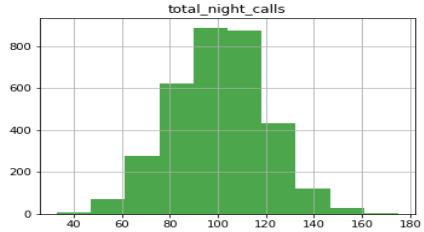


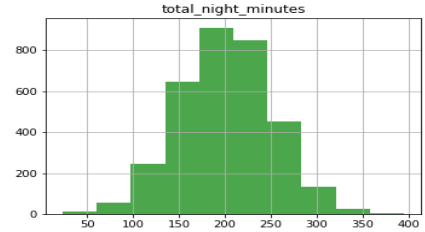


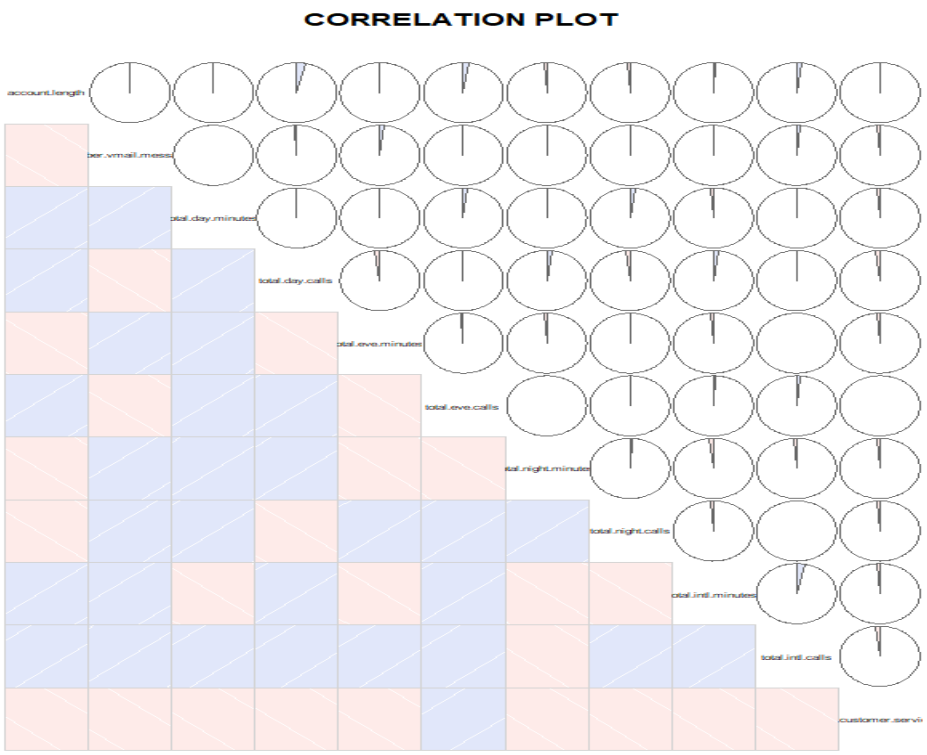


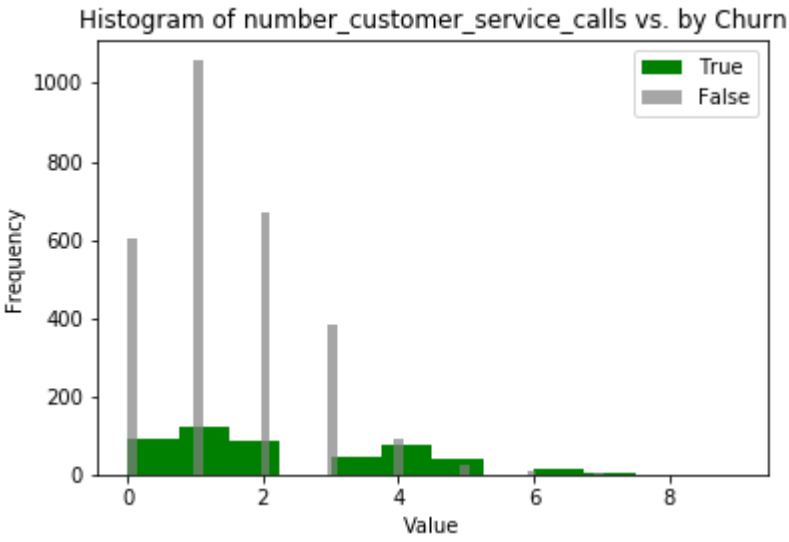


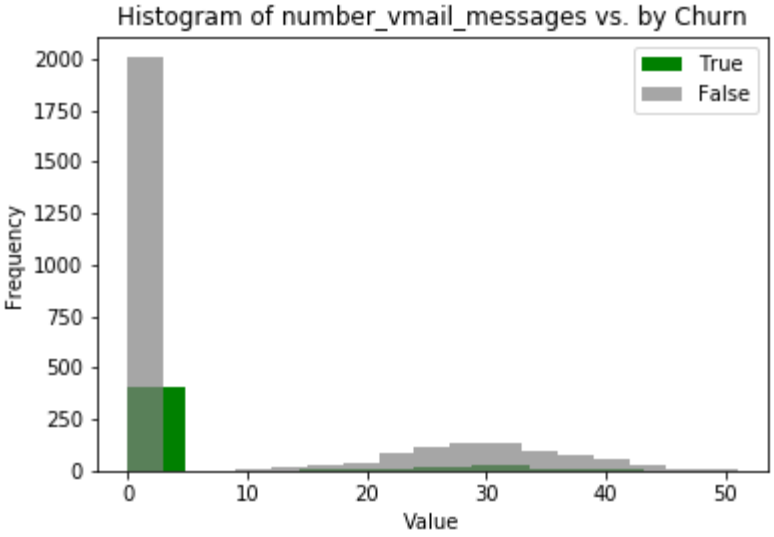







CORRELATION PLOT
account.length
total.day.minutes
total.day.calls
total.eve.minutes
total.eve.calls
total.night.minutes
total.night.calls
total.intl.minutes
total.intl.calls


Histogram of number_customer_service_calls vs. by Churn
True
False
Frequency
Value
0
200
400
600
800
1000
0
2
4
6
8


Histogram of number_vmail_messages vs. by Churn
True
False
Frequency
Value
2000
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50

Appendix B: R Code

```
rm(list=ls(all=T))


setwd("C:/Users/sakshi/EdwisorProject")

#Load Libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information","MASS", "rpart", "gbm", "ROSE",

    'sampling', 'DataCombine', 'inTrees')


install.packages(x)


lapply(x, require, character.only = TRUE)


rm(x)


## Read the data

customerData_train = read.csv("C:/Users/saksh/EdwisorProject/Train_data.csv", header = T)

customerData_test = read.csv("C:/Users/saksh/EdwisorProject/Test_data.csv", header = T)

test_Original = read.csv("C:/Users/saksh/EdwisorProject/Test_data.csv", header = T, na.strings
= c("","  ","NA"))

#############################################Explore the
data#############################################

#structure of data or data types

str(customerData_train)


head(customerData_train)
```

```
#count of unique values for each variable
apply(train, 2,function(x) length(table(x)))

#Converting area code as factor
customerData_train$area.code <- as.factor(customerData_train$area.code)
customerData_test$area.code <- as.factor(customerData_test$area.code)

#Removing phone number variable as it is not much of the use in predicting the target variable
customerData_train$phone.number <- NULL
customerData_test$phone.number <- NULL

#percentage of our target variable for True or False
round(prop.table(table(customerData_train$Churn))*100,2)
# False=85.51   True=14.49

#As we can see Our target Class is suffering from target imbalance

####################################Missing Values
Analysis#############################################
sapply(customerData_train, function(x) sum(is.na(x)))
sapply(customerData_test, function(x) sum(is.na(x)))
#no missing values are found


####################################Visualizing the
data#############################################

#Target class distribution
```

```
library(ggplot2)

ggplot(data = customerData_train,aes(x = Churn))+geom_bar() +  labs(y='Churn Count', title = 'Customer Churn or Not')


# Churning of customer according to State

ggplot(customerData_train, aes(fill=Churn, x=state)) +geom_bar(position="dodge") + labs(title="Churning ~ State")


# Churning of customer according to Voice Mail Plan

ggplot(customerData_train, aes(fill=Churn, x=voice.mail.plan)) +geom_bar(position="dodge") + labs(title="Churning ~ Voice Mail Plan")


# Churning of customer according to international_plan

ggplot(customerData_train, aes(fill=Churn, x=international.plan)) +geom_bar(position="dodge") + labs(title="Churning ~ international_plan")


# Churning of customer according to area_code

ggplot(customerData_train, aes(fill=Churn, x=area.code)) +geom_bar(position="dodge") + labs(title="Churning ~ Area Code")


# Churning of customer according to area_code by international_plan

ggplot(customerData_train, aes(fill=Churn, x=area.code)) +geom_bar(position="dodge") + facet_wrap(~international.plan)+labs(title="Churning ~ Area Code by International Plan")


# Churning of customer according to area_code by voicemail_plan

ggplot(customerData_train, aes(fill=Churn, x=area.code)) +geom_bar(position="dodge") + facet_wrap(~voice.mail.plan)+labs(title="Churning ~ Area Code by Voice Mail Plan")


# Churn of international_plan by voice_mail_plan and Area Code
```

```
ggplot(customerData_train, aes(fill=Churn, x=international.plan))
+geom_bar(position="dodge") + facet_wrap(area.code~voice.mail.plan)+labs(title="Churn of
international_plan by voice_mail_plan and Area Code")


# Churn ~ international_plan by voice_mail_plan

ggplot(customerData_train, aes(fill=Churn, x=international.plan))
+geom_bar(position="dodge") + facet_wrap(~voice.mail.plan)+labs(title="Churn ~
international_plan by voice_mail_plan")


####################################Exploratory data
analysis##############################################

##Data Manupulation; convert string categories into factor numeric

CategoricalToNumeric <- function(df){

  for(i in 1:ncol(df)){

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

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

    }

  }

  return(df)

}

#Converting Categorical to level -> factors

customerData_train = CategoricalToNumeric(customerData_train)

customerData_test = CategoricalToNumeric(customerData_test)


# storing all numeric variables

NumericIndex = sapply(customerData_train, is.numeric)

NumericData = customerData_train[,NumericIndex]

NumericColumnNames = colnames(NumericData)
```

```
#Checking for categorical features
cat_index = sapply(customerData_train,is.factor)
cat_data = customerData_train[,cat_index]
cat_col = colnames(cat_data)[-5]  #Removing target variable

####################################Outlier
Analysis#############################################

# We are skipping outliers analysis becoz we already have an Class Imbalance problem.

# for (i in 1:length(NumericColumnNames))
# {
#   assign(paste0("gn",i),
#          ggplot(aes_string(y = (NumericColumnNames[i]), x = 'Churn'),data = train) +
#           stat_boxplot(geom = "errorbar", width = 0.5) +
#           geom_boxplot(outlier.colour="blue", fill = "skyblue",
#                        outlier.shape=18,outlier.size=1, notch=FALSE) +
#           labs(y=NumericColumnNames[i],x="Churn")+
#           ggtitle(paste("Box plot of responded for",NumericColumnNames[i])))
# }

#gn1
#
## Plotting plots together
#gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
#gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
#gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
```

```
#gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
# gridExtra::grid.arrange(gn13,gn14,gn15,ncol=3)

# #Removing oulier by replacing with NA and then impute
# for(i in NumericColumnNames){
#   print(i)
#   outv = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
#   print(length(outv))
#   train[,i][train[,i] %in% outv] = NA
# }
#
# #checking all the missing values
# library(DMwR)
# sum(is.na(train))
# train = knnImputation(train, k=3)  #as it gives error so we going via mean or median



####################################Feature
Selection#############################################

#corrgram library is used to find correlation

library(corrgram)

corrgram(customerData_train[,NumericIndex],
      order = F,
      upper.panel=panel.pie,
```

```
        lower.panel=panel.shade,
        text.panel=panel.txt,
        main = 'CORRELATION PLOT')

#Dark blue color means highly positive correlation

#Chi Square Analysis

for(i in cat_col){
  print(names(cat_data[i]))
  print((chisq.test(table(cat_data$Churn,cat_data[,i])))[3])
  #print pvalue
}

#Removing Highly Corelated and Independent var
customerData_train = subset(customerData_train,select= -
c(state,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

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



####################################Feature
Scaling################################################

#all numeric var
NumericIndex = sapply(customerData_train, is.numeric)
NumericData = customerData_train[,NumericIndex]
```

```
NumericColumnNames = colnames(NumericData) #storing all the column name



#Checking Data of Continuous Variable


################  Histogram  ##################

hist(customerData_train$total.day.calls)

hist(customerData_train$total.day.minutes)

hist(customerData_train$account.length)


# as we can see from histograms Most of the data is uniformally distributed

#Using data Standardization/Z-Score


for(i in NumericColumnNames){

  print(i)

  customerData_train[,i] = (customerData_train[,i] -
mean(customerData_train[,i]))/sd(customerData_train[,i])

  customerData_test[,i] = (customerData_test[,i] -
mean(customerData_test[,i]))/sd(customerData_test[,i])

}



###################################Sampling of
data##############################################v


# #Divide data into train and test using stratified sampling method


library(caret)
```

```
set.seed(101)

split_index = createDataPartition(customerData_train$Churn, p = 0.66, list = FALSE)

trainset = customerData_train[split_index,]

validation_set  = customerData_train[-split_index,]


#Checking Train Set Target Class

table(trainset$Churn)

# False=1881    True=319

#


# #Our class is Imbalanced

# Synthetic Over Sampling the minority class & Under Sampling Majority Class to have a good Training Set


library(ROSE)  #---> Lib for Over and Under Sampling


trainset <- ROSE(Churn~.,data = trainset,p = 0.5,seed = 101)$data


table(trainset$Churn)

# False = 1101  True = 1099


##################################Model Selection#####################################


#function for CalculateMetricesulating the FNR,FPR,Accuracy

CalculateMetrices <- function(con_mat){

  TN = con_mat[1,1]

  FP = con_mat[1,2]
```

```
  FN = con_mat[2,1]
  TP = con_mat[2,2]
  # #CalculateMetricesulations
  print(paste0('Accuracy :- ',((TN+TP)/(TN+TP+FN+FP))*100))
  print(paste0('FNR :- ',((FN)/(TP+FN))*100))
  print(paste0('FPR :- ',((FP)/(TN+FP))*100))
  print(paste0('precision :-  ',((TP)/(TP+FP))*100))
  print(paste0('recall/TPR/Sensitivity :-  ',((TP)/(TP+FN))*100))
  print(paste0('Specificity/TNR :-  ',((TN)/(TN+FP))*100))
  plot(con_mat)
}

##############################Random
Forest########################################
library(randomForest)

set.seed(101)
RF_model = randomForest(Churn ~ ., trainset,ntree= 500,importance=T,type='class')
plot(RF_model)

#Predict test data using random forest model
RF_Predictions = predict(RF_model, validation_set[,-15])

##Evaluate the performance of classification model
con_mat_RF = table(validation_set$Churn,RF_Predictions)
CalculateMetrices(con_mat_RF)
plot(RF_model)
```

```
# Result on validaton set
# [1] "Accuracy :- 86.2312444836717"
# [1] "FNR :- 17.6829268292683"
# [1] "FPR :- 13.1062951496388"
# [1] "precision :-  51.5267175572519"
# [1] "recall/TPR :-  51.5267175572519"
# [1] "recall/TPR/Sensitivity :-  82.3170731707317"
# [1] "Specificity/TNR :-  86.8937048503612"

############################LOGISTIC
REGRESSION#####################################
set.seed(101)
logit_model = glm(Churn ~., data = trainset, family =binomial(link="logit"))
summary(logit_model)
#Prediction
logit_pred = predict(logit_model,newdata = validation_set[,-15],type = 'response')
#Converting Prob to number or class
logit_pred = ifelse(logit_pred > 0.5,2,1)
logit_pred = as.factor(logit_pred)
##Evaluate the performance of classification model
con_mat_logit = table(validation_set$Churn, logit_pred)
CalculateMetrices(con_mat_logit)
plot(logit_model)
#roc(validation_set$Churn~logit_pred)

# Result on validaton set
# [1] "Accuracy :- 78.1994704324801"
```

```
# [1] "FNR :- 28.6585365853659"
# [1] "FPR :- 20.6398348813209"
# [1] "precision :-  36.9085173501577"
# [1] "recall/TPR :-  36.9085173501577"
# [1] "recall/TPR/Sensitivity :-  71.3414634146341"
# [1] "Specificity/TNR :-  79.360165118679"
# ROC = 0.8017

##############################NN#######################################
set.seed(101)
## KNN impletation
library(class)

##Predicting Test data

knn_Pred = knn(train = trainset[,1:14],test = validation_set[,1:14],cl = trainset$Churn, k = 5)
#Confusion matrix
con_mat_knn = table(validation_set$Churn,knn_Pred)
confusionMatrix(con_mat_knn)
CalculateMetrices(con_mat_knn)

# Result on validaton set
# [1] "Accuracy :- 78.8172992056487"
# [1] "FNR :- 46.3414634146341"
# [1] "FPR :- 16.9246646026832"
# [1] "FPR :- 16.9246646026832"
# [1] "precision :-  34.9206349206349"
```

```
# [1] "recall//TPR :-  34.9206349206349"
# [1] "Sensitivity :-  53.6585365853659"
# [1] "Specificity :-  83.0753353973168"

################################## Naive Bayes ############################

library(e1071) #lib for Naive bayes
set.seed(101)
#Model Development and Training
naive_model = naiveBayes(Churn ~., data = trainset, type = 'class')
#prediction
naive_pred = predict(naive_model,validation_set[,1:14])

#Confusion matrix
con_mat_naive = table(validation_set[,15],naive_pred)
confusionMatrix(con_mat_naive)
CalculateMetrices(con_mat_naive)

# Result on validaton set
# [1] "Accuracy :- 83.1421006178288"
# [1] "FNR :- 29.2682926829268"
# [1] "FPR :- 14.7574819401445"
# [1] "FPR :- 14.7574819401445"
# [1] "precision :-  44.7876447876448"
# [1] "recall//TPR :-  44.7876447876448"
```

[1] "Sensitivity :- 70.7317073170732"

[1] "Specificity :- 85.2425180598555"

#Random forest model out performs the all other model in FNR,FPR and Accuracy.

Final Random Forest Model with tuning parameters##

set.seed(101)

final_model = randomForest(Churn~.,data = trainset,ntree=800,mtry=4,importance=TRUE,type = 'class')

final_validation_pred = predict(final_model,validation_set[,-15])

cm_final_valid = table(validation_set[,15],final_validation_pred)

confusionMatrix(cm_final_valid)

CalculateMetrices(cm_final_valid)

#Result on validation set after parameter tuning

[1] "Accuracy :- 86.4960282436011"

[1] "FNR :- 17.0731707317073"

[1] "FPR :- 12.8998968008256"

[1] "FPR :- 12.8998968008256"

[1] "precision :- 52.1072796934866"

[1] "recall//TPR :- 52.1072796934866"

[1] "Sensitivity :- 82.9268292682927"

[1] "Specificity :- 87.1001031991744"

#Variable Importance

#builting function in Random forest lib

```
importance(final_model)
#builtin func
varImpPlot(final_model)

#Plotting ROC curve and CalculateMetricesulate AUC metric
library(pROC)
PredictionwithProb <-predict(final_model,validation_set[,-15],type = 'prob')
auc <- auc(validation_set$Churn,PredictionwithProb[,2])
auc
# # AUC = 89.47
plot(roc(validation_set$Churn,PredictionwithProb[,2]))
##################################### Final Prediction On test Data
set##############################################

rmExcept(c("final_model","train","test","train_Original","test_Original","CalculateMetrices"))

set.seed(101)
final_test_pred = predict(final_model,customerData_test[,-15])
cm_final_test = table(customerData_test[,15],final_test_pred)
confusionMatrix(cm_final_test)
CalculateMetrices(cm_final_test)

# #Final Test Prediction
# [1] "Accuracy :- 85.9028194361128"
# [1] "FNR :- 15.625"
```

```
# [1] "FPR :- 13.8600138600139"



# [1] "FPR :- 13.8600138600139"

# [1] "precision :-  48.586118251928"

# [1] "recall//TPR :-  48.586118251928"

# [1] "Sensitivity :-  84.375"

# [1] "Specificity :-  86.1399861399861"


#Plotting ROC curve and CalculateMetricesulate AUC metric

install.packages("pROC")

library(pROC)

finalPredictionwithProb <-predict(final_model,customerData_test[,-15],type = 'prob')

auc <- auc(customerData_test$Churn,finalPredictionwithProb[,2])

auc

# # AUC = 91.74

plot(roc(customerData_test$Churn,finalPredictionwithProb[,2]))

##################Saving output to
file##############################################################################
###


test_Original$predicted_output <- final_test_pred

test_Original$predicted_output <- gsub(1,"False",test_Original$predicted_output)

test_Original$predicted_output <- gsub(2,"True",test_Original$predicted_output)


#Entire Comparison

write.csv(test_Original,'C:/Users/saksh/EdwisorProject/Customer_Churn_outputR.csv',row.nam
es = F)
```

```
#Phonenumber and Churning class and probab
submit <- data.frame(test_Original$state
                     test_Original$area.code,
                     test_Original$international.plan,
                     test_Original$voice.mail.plan,
                     test_Original$phone.number,
                     test_Original$predicted_output,
                     finalPredictionwithProb[,1],
                     finalPredictionwithProb[,2])

colnames(submit) <- c("State","Area Code","International Plan","Voice Mail
Plan","Phone_Number",
                      "Predicted_Output","Probability_of_False","Probability_of_True")

write.csv(submit,file =
'C:/Users/saksh/EdwisorProject/Final_Churn_Class_Probab.csv',row.names = F)
rm(list = ls())
```

References

https://stackoverflow.com/questions/10438752/adding-x-and-y-axis-labels-in-ggplot2

Gradient boosting in R | DataScience+datascienceplus.com

How to Rename Columns in the Pandas Python Librarychartio.com

https://docs.python.org/3.4/library/statistics.html

https://seaborn.pydata.org/generated/seaborn.heatmap.html

https://www.researchgate.net/post/what_are_the_best_methods_for_filling_in_missing_values

https://scikit-learn.org/0.15/auto_examples/imputation.html

https://datascience.stackexchange.com/questions/24989/imputing-for-multiple-missingvariables-using-sklearn