­­

Churn Prediction

* CH SAI SASHANK

**Contents**

1 Introduction

1.1 Problem Statement

1.2 Data

2 Methodology

2.1 Pre Processing

2.2 Outlier Analysis

2.3 Feature Selection

2.4 Feature Engineering

3 Modeling

3.1 Model Selection

3.2 Model Evaluation

3 Conclusion

Appendix A –Python code

**Chapter 1**

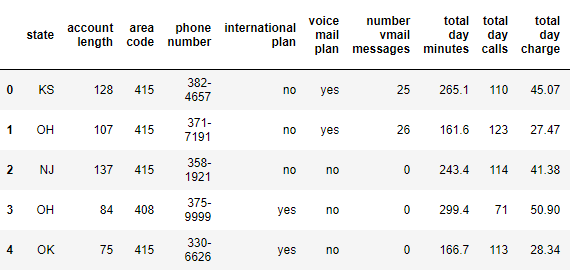
**Introduction**

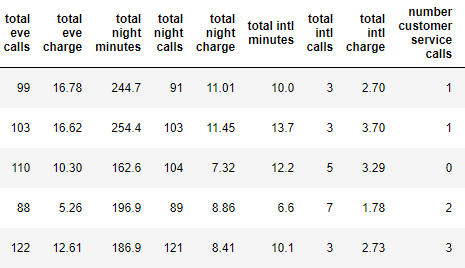
* 1. **Problem Statement**

The objective of this Case is to predict customer behaviour. Based on the provided dataset that has customer usage pattern and if the customer has moved or not. With the help of given predictor variables and old data, we would like to predict the customer churn in future.

* 1. **Data Understanding**

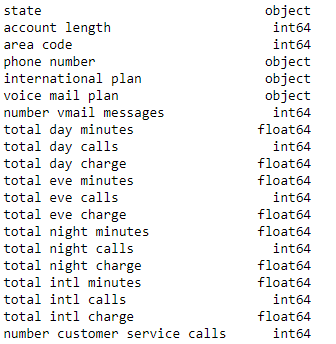
Our task is to build the classification model which will predict the behaviour of customer depending on the multiple factors provided in the dataset. Given below is sample of dataset we are using to predict the customer churn.





As you can see in the table below we have the following variables and their data types, using which we have to correctly predict the customer churn.

Predictor Variables



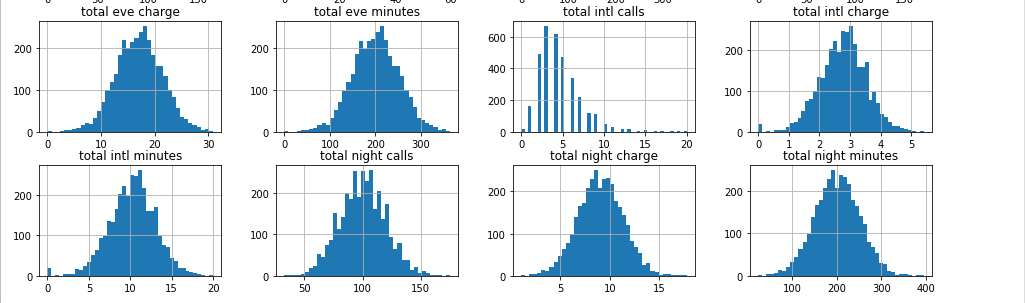
**Chapter 2**

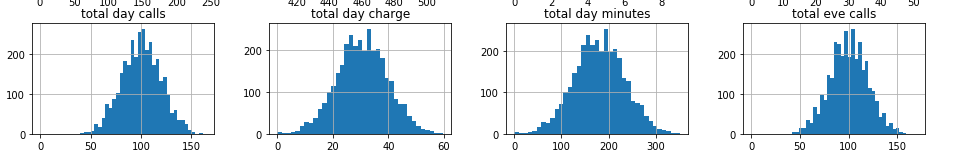
**Methodology**

**2.1 Pre-processing**

The most important thing required is need to look at the data before we do modelling. The data must be tidy and clean to proceed further. The data must be free from noisy data known as outliers and missing values need to be handled properly. This process is known as exploratory data analysis (EDA).The pre-requisites of many machine learning algorithm is that data must be normally distributed.

**Checking for normality with the histograms**

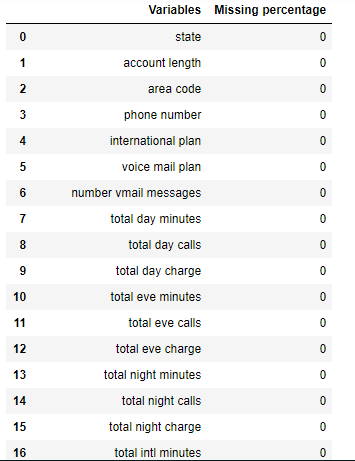


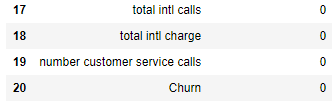


Could see that the data is normally distributed, one of pre-requisite before building a machine learning model.

**Missing values**

As there are no missing values in the dataset, proceeding with further steps.





**2.2 Outliers analysis**

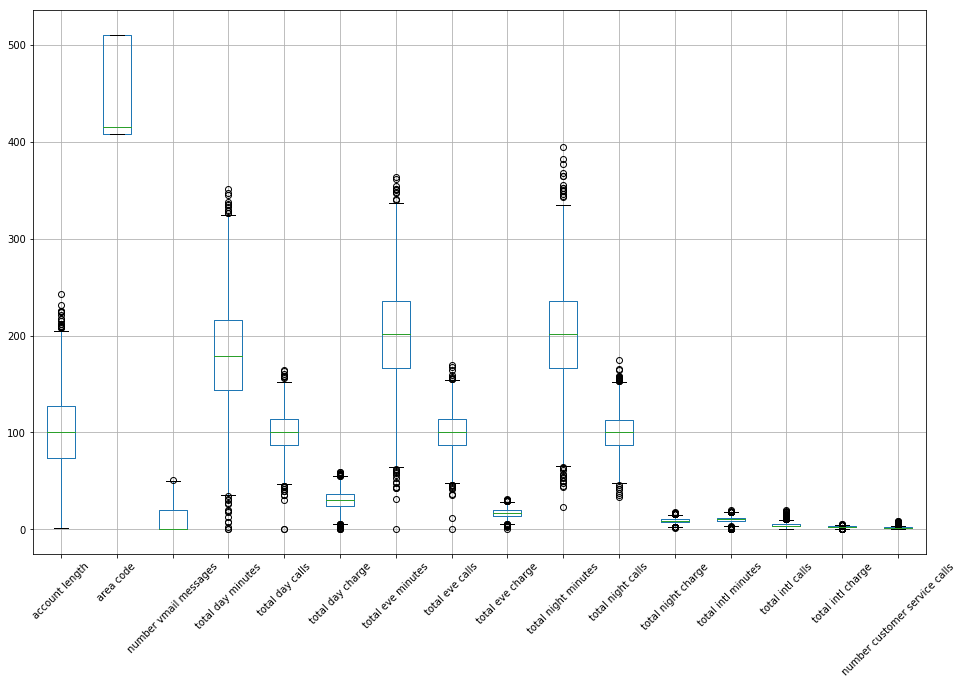
One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers, visualize the outliers using boxplots. With the help of inter-quartile range, we can either replace or delete the outliers by setting the benchmark

min=Q1-(1.5 \* IQR)  
max=Q3+(1.5 \* IQR)

IQR=Q3-Q1

The values which are below minimum and above maximum can either be deleted or replaced with mean values.

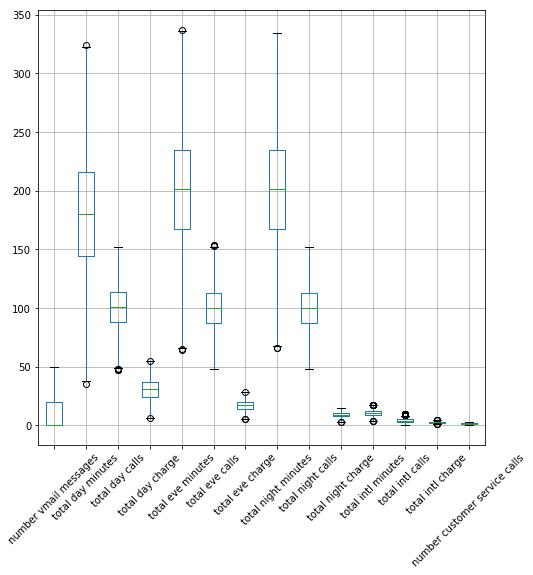
Before handling outliers



Could see in the above picture there are lot of outliers in the data which needs to be addressed.  
Otherwise outliers usually should be adversely affecting the modelling result and distorting the distributions.

If we remove the outliers from the data we may lose valuable information so we are going to impute mean of the variable into outliers.

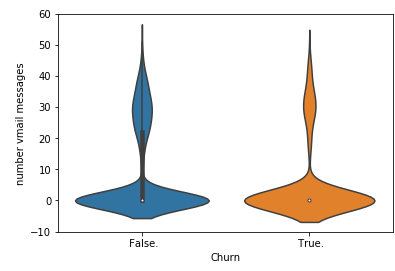
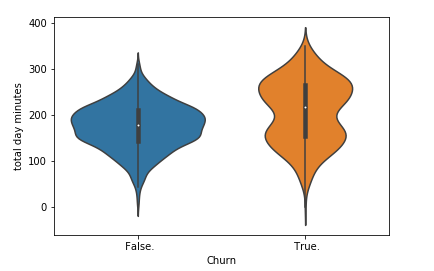
After handling outliers

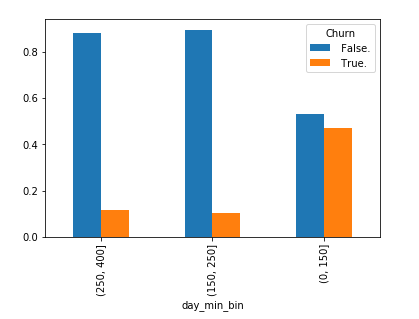


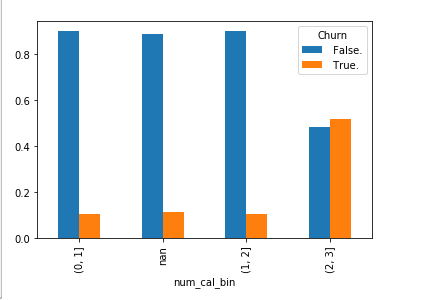
Now we can see that the presence of outliers is very low. Hence now the data cannot be skewed in the direction of outliers.

**Exploratory Data analysis**

Exploratory data analysis (EDA) is an approach [analysing](https://en.wikipedia.org/wiki/Data_analysis) [data sets](https://en.wikipedia.org/wiki/Data_set) to summarize their main characteristics, often with visual methods .Performed a lot of visualization on data to get insights which might be very helpful for feature selection.

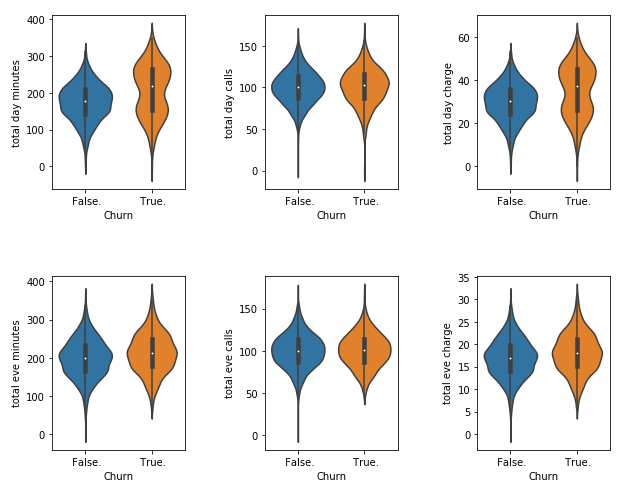


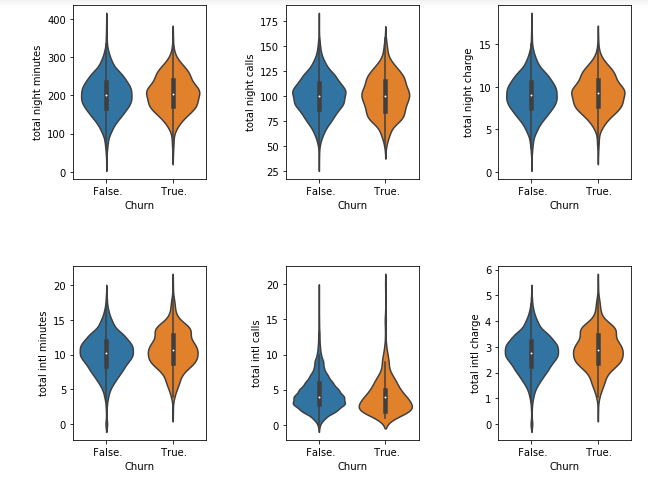




Above two images depict the binning of number of calls and day minutes and checking the data distribution.

Categorical vs continuous variable:

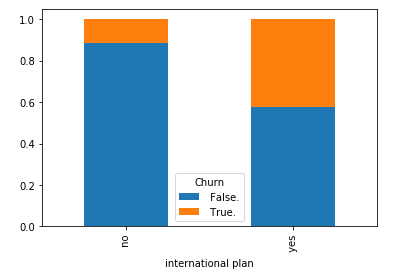


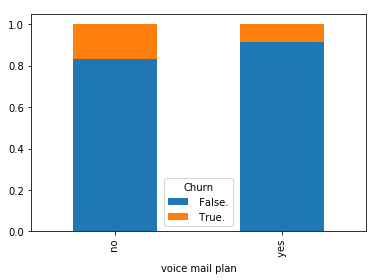


In Bivariate Analysis we will check the relation between any two variables using graphs or statistical techniques. Here I have analysed relation between target variable and other predictor variables. We use violin plots for categorical vs continuous variables and stacked bars for categorical vs categorical.

From above plots we can see that for most of the variables distribution for two categories in Churn is same. Data is concentrated around median. For variable ‘total day minutes’ the median is slightly higher for True category compared to false category.

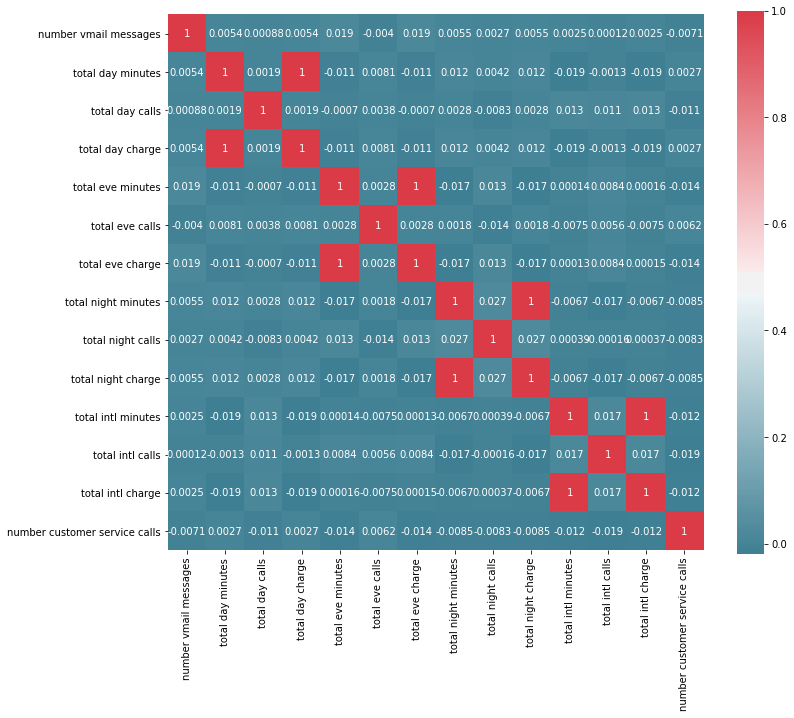
Categorical vs categorical variable:





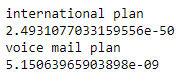
**2.3 Feature Selection**

Before performing any type of modelling 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 some variable are dependent on other variable which adds multi collinearity problem. We have to find the collinear variable and remove before training the model. There are several methods of doing that. Below we are using correlation plot to find the relation between continuous variables and chi-square test to find the relation between categorical variables.



From above we can find 4 highly correlated variable. It’s obvious that charges depends on minutes of call. We can remove these from data -total day minutes, total eve minutes, total night minutes, total intl minutes.

For the categorical variables: international plan, voice mail plan with the help of chi-square test calculated the probabilities which is less than 0.05.Hence the variables are dependent.



Also we can remove state, area code and phone number as they are not helpful for churn prediction and not significant for building the model.

**2.4 Feature Extraction**

Combining the attributes into new reduced set of features is called feature extraction. It leads to much smaller and rich set of attributes.

Created features from existing for both test and train data:

day\_charge\_per\_min= total day charge / total day minutes

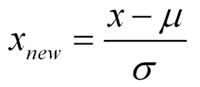
eve\_charge\_per\_min= total eve charge / total eve minutes

night\_charge\_per\_min= total night charge / total night minutes

intl\_charge\_per\_minute=total intl charge/total intl minutes

Feature Scaling:

The standardization assumes a data to be normal distribution. As we could see the data is normally distributed, proceeding for standardization. Standardization transform it to have zero mean and unit variance, for example using the equation below:

[](http://3.bp.blogspot.com/_xqXlcaQiGRk/RpO4yR0oKtI/AAAAAAAAABM/7rUWCMwizus/s1600-h/fig2.png)

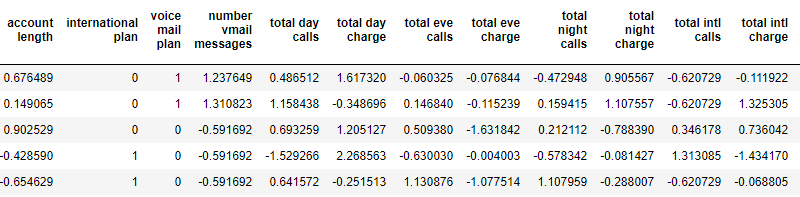
With the help of Standard scaler, standardized the data. The sign of Z is positive when the value is above mean, negative when the value is below mean. This pre-processing technique train the algorithms much faster by speeding up the calculations.

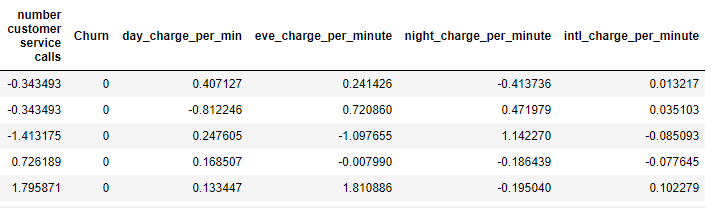
Feature scaling is to bring all the variables into proportion with one another. As variables have different scales model add more weight to variables which has bigger values. So we need to transform them into single scale.

Dummy encoding for categorical variables:

With the help of label encoder replaced the yes and no with 0 and 1.Hence now the data can be feed to the model.

After standardization and dummy encoding. Please look into below data





**3.Modelling**

**3.1 Model selection**

As the problem belongs to classification type, planning to use decision trees and random forest ensemble and logistic regression.

**Decision Tree**  
Decision trees is a predictive model based on branching series of Boolean tests which is very helpful in classification problems.   
Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable.

**Random forest**  
  
Random forest is ensemble that consists of many decision trees. In Random Forest, we’ve collection of decision trees (so known as “Forest”). To classify a new object based on attributes, each tree gives a classification and we say the tree “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest).  
  
**Logistic Regression**

Logistic regression is used to estimate discrete values ( Binary values like 0/1, yes/no, true/false ) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a [logit function](https://en.wikipedia.org/wiki/Logistic_function). Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).  
  
Logistic regression is an estimation of Logit function. Logit function is simply a log of odds in favor of the event. This function creates a s-shaped curve with the probability estimate, which is very similar to the required step wise function

**3.2 Model evaluation**

The model can be evaluated based on following metrics

* Confusion matrix: Describes the performance of classification model. Row represents actual class/values and columns represents predicted class/values.
* Accuracy : How accurately the model can be able to classify the observations.  
   accuracy = TP + TN / Total observations
* Specificity : The proportion of actual negative cases which are correctly identified.  
   Specificity = TN / TN+ FP
* Recall : The proportion of actual positive cases which are correctly identified.  
   Recall = TP/ TP + FN
* Precision : precision (also called [positive predictive value](https://en.wikipedia.org/wiki/Positive_predictive_value)) is the fraction of relevant instances among the retrieved instances.  
   Precision = TP/ TP+FP
* False negative rate : also known as type-II error. Accepting a false null hypothesis.  
   FNR =FN/FN+TP

**Some of rules extracted from decision tree** :

Rule 98/9: (26.8, lift 1.5)

account.length <= 135.4164

international.plan = 1

total.intl.calls > -0.07077914

total.intl.charge <= -0.481658

intl.charge.per.minute > -79.68206

-> class 0 [0.965]

Rule 98/10: (25.5, lift 1.5)

international.plan = 0

total.day.charge <= 41.36331

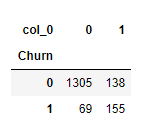
total.eve.calls > 111.8131

total.night.calls > 118.7325

-> class 0 [0.964]

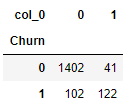
**3.3 Conclusion**

**For decision tree model:**

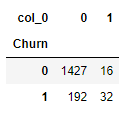
Confusion matrix :  


Accuracy : 0.875  
Specificity : 0.904  
Recall : 0.691  
Precision : 0.529  
FNR : 0.30

**For Random forest model**Confusion matrix :



Accuracy : 0.914  
Specificity : 0.97  
Recall : 0.54  
Precision : 0.74  
FNR : 0.45

**For Logistic regression model**  
Confusion matrix :  


Accuracy : 0.875  
Specificity : 0.98  
Recall : 0.14  
Precision : 0.66  
FNR : 0.85

We are more interested to know who is going to unsubscribe the plan so that we can take actions to reduce that. So we can choose **False Positive Rate** as our evaluation metric which calculates which calculate percentage of misclassification as positive. We try to reduce the False Positive Rate.

Since the precision, false positive rate and false negative rate of random forest is comparatively better than decision tree and logistic regression. I would like to choose the random forest model. Also the accuracy is pretty good for random forest model.

**Model Selection**:  
  
From above models we can see Random forest performed well so we choose random forest as our final model.

**APPENDIX**

**Python Code :**

# Loading libraries  
import os  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import seaborn as sns  
from scipy.stats import chi2\_contingency

#Setting working directory  
os.chdir("D:\Data Science edwisor\Projects\Churn reduction")  
os.getcwd()

#loading data  
Train=pd.read\_csv("Train\_data.csv")  
Test=pd.read\_csv("Test\_data.csv")

Train.head()

Train.dtypes

Train.columns

Train.describe()

#Categorical variables  
cat\_names=Train.select\_dtypes('object').columns  
cat\_names

#Continous variables  
cont\_names=Train.select\_dtypes(['float64','int64']).columns  
cont\_names

Exploratory Analysis

#checking normality  
Train.hist(figsize=(16,10),bins='auto')

#checking outliers  
Train.boxplot(figsize=(16,10),rot=45)

Train['Churn'].value\_counts().plot.bar(title="Churn count",rot=45)

Train['international plan'].value\_counts().plot.bar(title='International Plan',rot=45)

Train['voice mail plan'].value\_counts().plot.bar(title='Voicemail Plan',rot=45)

Train.groupby(Train['Churn'])['total intl calls'].mean().plot.bar()

plt.scatter(Train['total day minutes'],Train['total eve minutes'])

plt.figure(1)  
sns.violinplot(x=Train['Churn'],y=(Train['number vmail messages']))

plt.figure(1)  
sns.violinplot(x=Train['Churn'],y=Train['total day minutes'])

bins=[0,150,250,400]  
Train['day\_min\_bin']=pd.cut(Train['total day minutes'],bins=bins)

ct = pd.crosstab(Train['day\_min\_bin'],Train['Churn'])  
ct.div(ct.sum(1),axis=0).plot.bar()

plt.figure(figsize=(10,8))  
plt.subplot(231)  
plt.subplots\_adjust(wspace=0.6,hspace=0.5)  
sns.violinplot(x=Train['Churn'],y=Train['total day minutes'])  
plt.subplot(232)  
sns.violinplot(x=Train['Churn'],y=Train['total day calls'])  
plt.subplot(233)  
sns.violinplot(x=Train['Churn'],y=Train['total day charge'])  
plt.subplot(234)  
sns.violinplot(x=Train['Churn'],y=Train['total eve minutes'])  
plt.subplot(235)  
sns.violinplot(x=Train['Churn'],y=Train['total eve calls'])  
plt.subplot(236)  
sns.violinplot(x=Train['Churn'],y=Train['total eve charge'])

plt.figure(figsize=(10,8))  
plt.subplot(231)  
plt.subplots\_adjust(wspace=0.6,hspace=0.5)  
sns.violinplot(x=Train['Churn'],y=Train['total night minutes'])  
plt.subplot(232)  
sns.violinplot(x=Train['Churn'],y=Train['total night calls'])  
plt.subplot(233)  
sns.violinplot(x=Train['Churn'],y=Train['total night charge'])  
plt.subplot(234)  
sns.violinplot(x=Train['Churn'],y=Train['total intl minutes'])  
plt.subplot(235)  
sns.violinplot(x=Train['Churn'],y=Train['total intl calls'])  
plt.subplot(236)  
sns.violinplot(x=Train['Churn'],y=Train['total intl charge'])

bins = [0,1,2,3,10]  
Train['num\_cal\_bin']=pd.cut(Train['number customer service calls'],bins=bins)

num\_cal = pd.crosstab(Train['num\_cal\_bin'],Train['Churn'])  
num\_cal.div(num\_cal.sum(1),axis=0).plot.bar()

#categorical vs categorical  
international\_plan = pd.crosstab(Train['international plan'],Train['Churn'])  
international\_plan.div(international\_plan.sum(1),axis=0).plot.bar(stacked=True)

Voice\_plan = pd.crosstab(Train['voice mail plan'],Train['Churn'])  
Voice\_plan.div(Voice\_plan.sum(1),axis=0).plot.bar(stacked=True)

Train.columns

#dropping columns  
Train.drop(['day\_min\_bin', 'num\_cal\_bin'],axis=1,inplace=True)

Train\_copy = Train.copy()  
Test\_copy = Test.copy()

Combined=pd.concat([Train,Test],axis=0)

Combined.shape

#Handling Missing values

missing\_val = pd.DataFrame(Train.isnull().sum()).reset\_index()  
missing\_val.rename(columns={'index' :'Variables', 0 : 'Missing percentage'})

#Handling outliers

column\_names=list(Train.select\_dtypes(['int64','float64']).columns)  
column\_names.remove('account length')  
column\_names.remove('area code')  
column\_names

#for train data  
for i in column\_names:  
 print(i)  
 q25,q75=np.percentile(Train.loc[:,i],[25,75])  
 iqr=q75-q25  
 min=q25-(1.5 \* iqr)  
 max=q75+(1.5 \* iqr)  
 print(min,max)  
 Train[i].iloc[Train[Train.loc[:,i]<min].index]=np.mean(Train.loc[:,i])  
 Train[i].iloc[Train[Train.loc[:,i]>max].index]=np.mean(Train.loc[:,i])

#checking handliers after replacing with mean value  
Train[column\_names].boxplot(figsize=(8,8),rot=45)

# for test data  
for i in column\_names:  
print(i)  
q25,q75=np.percentile(Test.loc[:,i],[25,75])  
iqr=q75-q25  
min=q25-(1.5 \* iqr)  
max=q75+(1.5 \* iqr)  
print(min,max)  
Test[i].iloc[Test[Test.loc[:,i]<min].index]=np.mean(Test.loc[:,i])  
Test[i].iloc[Test[Test.loc[:,i]>max].index]=np.mean(Test.loc[:,i])

Test[column\_names].boxplot(figsize=(8,8),rot=45)

#Feature Selection

# will check the correlation between all continuous variables  
Cont = Combined.loc[:,column\_names]  
Corr = Cont.corr()  
f, ax = plt.subplots(figsize=(12, 10))

#Plot using seaborn library  
sns.heatmap(Corr, mask=np.zeros\_like(Corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),square=True, ax=ax,annot=True)

Cat\_names = Combined.select\_dtypes('object').columns  
Cat\_names

#chi-square test for categorical variables  
for i in ['international plan', 'voice mail plan']:  
print(i)  
chi2, p, dof, ex = chi2\_contingency(pd.crosstab(Train['Churn'], Train[i]))  
print(p)

#Creating features from existing   
Train['day\_charge\_per\_min']=Train['total day charge']/Train['total day minutes']  
Train['eve\_charge\_per\_minute']=Train['total eve charge']/Train['total eve minutes']  
Train['night\_charge\_per\_minute']=Train['total night charge']/Train['total night minutes']  
Train['intl\_charge\_per\_minute']=Train['total intl charge']/Train['total intl minutes']

Test['day\_charge\_per\_min']=Test['total day charge']/Test['total day minutes']  
Test['eve\_charge\_per\_minute']=Test['total eve charge']/Test['total eve minutes']  
Test['night\_charge\_per\_minute']=Test['total night charge']/Test['total night minutes']  
Test['intl\_charge\_per\_minute']=Test['total intl charge']/Test['total intl minutes']

col\_names=list(Combined.select\_dtypes(['int64','float64']).columns)  
col\_names.remove('account length')  
col\_names.remove('area code')

#after removing account lenght,area code checking the correlation  
df\_cont = Combined.loc[:,col\_names]  
corr = df\_cont.corr()  
f, ax = plt.subplots(figsize=(12, 10))

#Plot using seaborn library  
sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),square=True, ax=ax,annot=True)

#removing correlated continous variables  
remove\_var = ['total day minutes','total eve minutes','total night minutes','total intl minutes']  
Train.drop(columns=remove\_var,inplace = True)  
Test.drop(columns=remove\_var,inplace = True)

# we can remove state , area code and phone number as they are not helpful for churn prediction  
delete = ['state','phone number','area code']  
Train.drop(columns=delete,inplace = True)  
Test.drop(columns=delete,inplace = True)

#Feature scaling

Continous=Train.select\_dtypes(['int64','float64']).columns.tolist()  
Continous

#standardising the data

from sklearn.preprocessing import StandardScaler  
scale = StandardScaler(with\_std=True)  
Train.loc[:,Continous]=pd.DataFrame(scale.fit\_transform(Train.loc[:,Continous]),columns=Continous)Test.loc[:,Continous] = pd.DataFrame(scale.fit\_transform(Test.loc[:,Continous]),columns=Continous)

Categorical=Train.select\_dtypes('object').columns  
Categorical

# Changing catogorical variables  
#replace Yes with 1 and no with 0  
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
Train['international plan']= le.fit\_transform(Train['international plan'])  
Train['voice mail plan']= le.fit\_transform(Train['voice mail plan'])  
Train['Churn']= le.fit\_transform(Train['Churn'])

Test['international plan']= le.fit\_transform(Test['international plan'])  
Test['voice mail plan']= le.fit\_transform(Test['voice mail plan'])  
Test['Churn']= le.fit\_transform(Test['Churn'])

#checking the data after replacing categorical variable with 0 &1   
Train.head()

#Model building

#Decision tree  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy\_score

Y\_train = Train['Churn']  
X\_train = Train.drop('Churn',axis=1)  
Y\_test = Test['Churn']  
X\_test = Test.drop('Churn',axis=1)

model = DecisionTreeClassifier(criterion='entropy')  
model.fit(X\_train,Y\_train)

from sklearn.tree import export\_graphviz  
export\_graphviz(model,out\_file = "tree.dot",feature\_names = X\_train.columns)

churn\_prediction=model.predict(X\_test)

Conf\_matix=pd.crosstab(Y\_test,churn\_prediction)  
Conf\_matix

TN=Conf\_matix.iloc[0,0]  
FP=Conf\_matix.iloc[0,1]  
FN=Conf\_matix.iloc[1,0]  
TP=Conf\_matix.iloc[1,1]

Accuracy= (TN+TP)/(TN+FP+FN+TP)  
Accuracy

accuracy\_score(Y\_test,churn\_prediction)

Specificity = TN/(TN+FP)  
Specificity

Recall = TP/(TP+FN)  
Recall

FNR=(FN\*100)/(FN+TP)  
FNR

Precision = TP/(TP+FP)  
Precision

#Random Forest  
from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(n\_estimators=5)  
rf\_model.fit(X\_train,Y\_train)

churn\_pred1 = rf\_model.predict(X\_test)  
  
accuracy\_score(Y\_test,churn\_pred1)

Cm=pd.crosstab(Y\_test,churn\_pred1)  
Cm

TN=Cm.iloc[0,0]  
FP=Cm.iloc[0,1]  
FN=Cm.iloc[1,0]  
TP=Cm.iloc[1,1]

Accuracy= (TN+TP)/(TN+FP+FN+TP)  
Accuracy

Specificity = TN/(TN+FP)  
Specificity

Recall = TP/(TP+FN)  
Recall

FNR=(FN\*100)/(FN+TP)  
FNR

Precision = TP/(TP+FP)  
Precision

#Logistic regression  
from sklearn.linear\_model import LogisticRegression

lg\_model = LogisticRegression()  
lg\_model.fit(X\_train,Y\_train)

churn\_pred2=lg\_model.predict(X\_test)

accuracy\_score(Y\_test,churn\_pred2)  
  
Cm1=pd.crosstab(Y\_test,churn\_pred2)  
Cm1

TN=Cm1.iloc[0,0]  
FP=Cm1.iloc[0,1]  
FN=Cm1.iloc[1,0]  
TP=Cm1.iloc[1,1]

Accuracy= (TN+TP)/(TN+FP+FN+TP)  
Accuracy

Specificity = TN/(TN+FP)  
Specificity

Recall = TP/(TP+FN)  
Recall

FNR=(FN\*100)/(FN+TP)  
FNR  
Precision = TP/(TP+FP)  
Precision