In [1]:

```
#setting working directory
import os
os.chdir("C:\\Users\\Sarthak Gupta\\Desktop\\DATA SCIENTIST\\EDWISOR\\PROJECT
\\PROJECT 2")
```

In [ ]:

```
#check current working directory
os.getcwd()
```

In [3]:

```
#importing data set
import pandas as pd
train = pd.read_csv('C:\\Users\\Sarthak Gupta\\Desktop\\DATA SCIENTIST\\EDWISO
R\\PROJECT\\PROJECT 2\\Train_data.csv')
test = pd.read_csv('C:\\Users\\Sarthak Gupta\\Desktop\\DATA SCIENTIST\\EDWISOR
\\PROJECT\\PROJECT 2\\Test_data.csv')
```

In [ ]:

```
train.dtypes
```

In [ ]:

```
#Exploratory Data Analysis
train.columns
```

In [ ]:

```
#Getting the structure of dataset
type(train),type(test)
```

In [ ]:

```
#Getting the number of variables and observations
train.shape,test.shape
```

In [ ]:

```
train.head(30)
```

In [ ]:

```
test.head(30)
```

In [ ]:

```
train.tail(30)
```

In [ ]:

```
test.tail(30)
```

In [ ]:

```
train.describe()
```

In [ ]:

```
train['Churn'].value_counts()
```

In [ ]:

```
test['Churn'].value_counts()
```

In [ ]:

```
#checking for null values
train.isnull().any()
```

In [ ]:

```
test.isnull().any()
```

In [45]:

```
#Deleting variables which do not form part of problem statement and are not
 going to influence the result in any way.
train1 = train.drop(['state','area code','phone number'],axis = 1)
test1 = test.drop(['state','area code','phone number'],axis = 1)
```

In [46]:

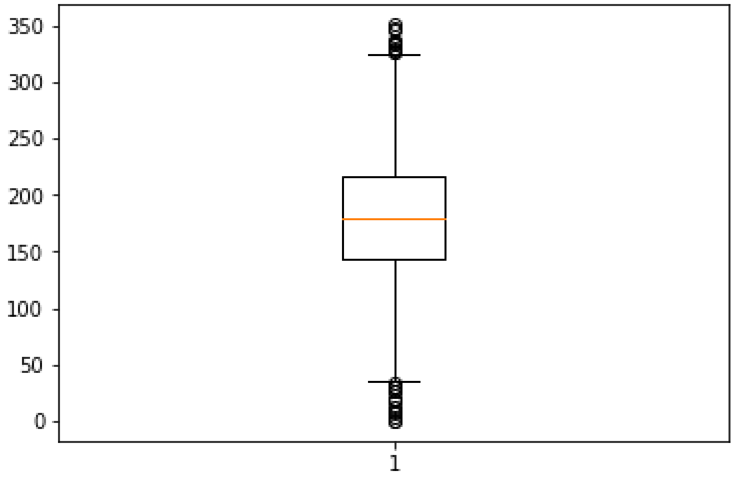
```
#outlier analysis
```

In [48]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.boxplot(train1['total day minutes'])
```

Out[48]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x9a2b2b8da0>],
 'caps': [<matplotlib.lines.Line2D at 0x9a2b2ca7f0>,
  <matplotlib.lines.Line2D at 0x9a2b2cac18>],
 'fliers': [<matplotlib.lines.Line2D at 0x9a2b2d94a8>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x9a2b2d9080>],
 'whiskers': [<matplotlib.lines.Line2D at 0x9a2b2b8ef0>,
  <matplotlib.lines.Line2D at 0x9a2b2ca3c8>]}
```



In [49]:

```
import numpy as np
df1 = train1.select_dtypes(include=[np.number])
```

In [50]:

```
#saving numeric names
cnames = df1.columns
```

In [ ]:

```
cnames
```

In [52]:

```
#Detect and delete outliers from data
for i in cnames:
     print(i)
     q75, q25 = np.percentile(train1.loc[:,i], [75 ,25])
     iqr = q75 - q25

     min = q25 - (iqr*1.5)
     max = q75 + (iqr*1.5)
     print(min)
     print(max)

     train1 = train1.drop(train1[train1.loc[:,i] < min].index)
     train1 = train1.drop(train1[train1.loc[:,i] > max].index)
```

```
account length
-5.5
206.5
number vmail messages
-30.0
50.0
total day minutes
34.83749999999992
325.1375000000001
total day calls
46.5
154.5
total day charge
6.125
55.125
total eve minutes
64.42499999999995
337.82500000000005
total eve calls
46.5
154.5
total eve charge
5.5550000000000015
28.634999999999998
total night minutes
64.3
337.90000000000003
total night calls
48.0
152.0
total night charge
2.9449999999999985
15.145000000000001
total intl minutes
3.1000000000000005
17.5
total intl calls
-1.5
10.5
total intl charge
0.8949999999999996
4.695
number customer service calls
-0.5
3.5
```

In [ ]:

```
train1.shape
```

In [54]:

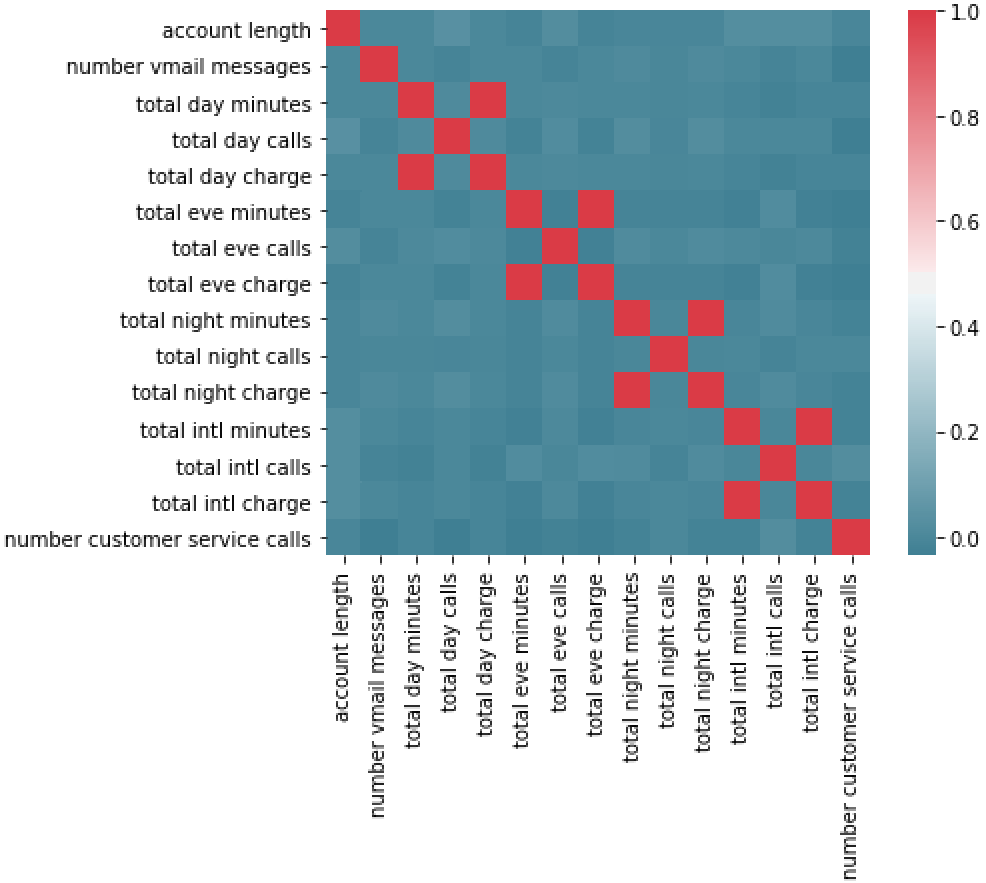
```
#Feature Selection
##Correlation analysis
#Correlation plot
df_corr = train1.loc[:,cnames]
```

```
In [55]: import seaborn as sns
         #Set the width and hieght of the plot
         f, ax = plt.subplots(figsize=(7, 5))

         #Generate correlation matrix
         corr = df_corr.corr()

         #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)
```

```
Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x9a2ce66a58>
```



```
In [56]: #Deleting variables having high corelation with another independent variable
         train_deleted = train1.drop(['total day charge','total eve charge','total nigh
         t charge','total intl charge'],axis=1)
```

```
In [57]: test_deleted = test1.drop(['total day charge','total eve charge','total night
          charge','total intl charge'],axis=1)
```

```
In [58]: #stripping of white space
         train_deleted['international plan'] = train_deleted['international plan'].appl
         y(lambda x: x.strip())
         train_deleted['Churn'] = train_deleted['Churn'].apply(lambda x: x.strip())
         train_deleted['voice mail plan'] = train_deleted['voice mail plan'].apply(lamb
         da x: x.strip())
```

In [59]:
```
test_deleted['international plan'] = test_deleted['international plan'].apply(
lambda x: x.strip())
test_deleted['Churn'] = test_deleted['Churn'].apply(lambda x: x.strip())
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].apply(lambda
 x: x.strip())
```

In [60]:
```
#converting to lower case
train_deleted['international plan'] = train_deleted['international plan'].appl
y(lambda x: x.lower())
train_deleted['Churn'] = train_deleted['Churn'].apply(lambda x: x.lower())
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].apply(lamb
da x: x.lower())
```

In [61]:
```
test_deleted['international plan'] = test_deleted['international plan'].apply(
lambda x: x.lower())
test_deleted['Churn'] = test_deleted['Churn'].apply(lambda x: x.lower())
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].apply(lambda
 x: x.lower())
```

In [ ]:
```
train_deleted.head()
```

In [24]:
```
#Chisquare test of independence
#Save categorical variables
cat_names = ["international plan","voice mail plan"]
```

In [25]:
```
from scipy.stats import chi2_contingency
#Loop for chi square values
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(train_deleted['Churn'], tr
ain_deleted[i]))
    print(p)
```

```
international plan
1.6860769270699622e-53
voice mail plan
2.6438944498671704e-07
```

MODEL BUILDING & PREDICTION

DECISION TREE

In [26]:
```
#Import Libraries for decision tree
from sklearn import tree
from sklearn.metrics import accuracy_score
```

In [27]:

```
train_deleted['international plan'] = train_deleted['international plan'].replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].replace('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].replace('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('yes', 1)
```

In [ ]:

```
train_deleted.head()
```

In [29]:

```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

In [ ]:

```
type(X_train)
```

In [31]:

```
#Decision Tree
C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)
```

In [32]:

```
#predict new test cases
C50_Predictions = C50_model.predict(X_test)
```

In [33]:

```
#confusion matrix
CM = pd.crosstab(y_test, C50_Predictions)
```

In [34]:

```
CM
```

Out[34]:

col_0	**false.**	**true.**
row_0		
false.	1366	77
true.	96	128

In [35]:
```
#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

In [36]:
```
#check accuracy of model
#accuracy_score(y_test, y_pred)*100
((TP+TN)*100)/(TP+TN+FP+FN)
```

Out[36]:
```
89.62207558488302
```

In [37]:
```
#False Negative rate
(FN*100)/(FN+TP)
```

Out[37]:
```
42.857142857142854
```

In [38]:
```
#Recall
(TP*100)/(TP+FN)
```

Out[38]:
```
57.142857142857146
```

In [64]:
```
 type(C50_Predictions)
```

Out[64]:
```
numpy.ndarray
```

In [74]:
```
pred = pd.DataFrame(C50_Predictions)
```

In [77]:
```
pred = pred.rename(columns = {0:'Prediction'})
```

In [ ]:
```
pred.head()
```

In [79]:
```
submission_DT_python = pd.concat([test,pred], axis=1)
```

In [ ]:
```
submission_DT_python.head()
```

In [81]:
```
# Writting a csv output
submission_DT_python.to_csv("submission_DT_python.csv",index=False)
```

RANDOM FOREST

In [39]:

```
train_deleted['international plan'] = train_deleted['international plan'].replace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].replace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('no', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('yes', 1)

test_deleted['international plan'] = test_deleted['international plan'].replace('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].replace('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('no', 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('yes', 1)
```

In [40]:

```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

In [41]:

```
type(X_train)
```

Out[41]:

```
numpy.ndarray
```

In [42]:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF_model = RandomForestClassifier(n_estimators = 250).fit(X_train, y_train)
```

In [43]:

```
RF_Predictions = RF_model.predict(X_test)
```

In [44]:

```
#confusion matrix
CM = pd.crosstab(y_test, RF_Predictions)
```

In [45]:

```
CM
```

Out[45]:

col_0	false.	true.
row_0		
false.	1441	2
true.	109	115

In [46]:

```
#Let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

In [47]:

```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
#accuracy has improved over decision tree
```

Out[47]:

```
93.34133173365326
```

In [48]:

```
#False Negative rate
(FN*100)/(FN+TP)
#FNR has increased over Decision Tree
```

Out[48]:

```
48.660714285714285
```

In [82]:

```
pred = pd.DataFrame(RF_Predictions)
```

In [83]:

```
pred = pred.rename(columns = {0:'Prediction'})
```

In [ ]:

```
pred.head()
```

In [86]:

```
submission_Randomforest_python =  pd.concat([test,pred], axis=1)
```

In [87]:

```
# Writting a csv output
submission_Randomforest_python.to_csv("submission_Randomforest_python.csv",ind
ex=False)
```

LOGISTIC REGRESSION

In [63]:

```
df2 = train_deleted.select_dtypes(include=[np.number])
cnames1 = df2.columns
```

In [64]:

```
cnames1
```

Out[64]:

```
Index(['account length', 'number vmail messages', 'total day minutes',
       'total day calls', 'total eve minutes', 'total eve calls',
       'total night minutes', 'total night calls', 'total intl minutes',
       'total intl calls', 'number customer service calls'],
      dtype='object')
```

In [65]:

```
#Making the data suitable for logistic regression
train_deleted['Churn'] = train_deleted['Churn'].replace('false.', 0)
train_deleted['Churn'] = train_deleted['Churn'].replace('true.', 1)
```

In [66]:

```
test_deleted['Churn'] = test_deleted['Churn'].replace('false.', 0)
test_deleted['Churn'] = test_deleted['Churn'].replace('true.', 1)
```

In [67]:

```
train_deleted['international plan'] = train_deleted['international plan'].repl
ace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].repl
ace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('n
o', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('y
es', 1)

test_deleted['international plan'] = test_deleted['international plan'].replac
e('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].replac
e('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('no'
, 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('ye
s', 1)
```

In [68]:

```
#Create logistic data. Save target variable first
train_logit = pd.DataFrame(train_deleted['Churn'])
test_logit = pd.DataFrame(test_deleted['Churn'])
```

In [ ]:

```
train_logit.head()
```

In [70]:

```
#Add continous variables
train_logit = train_logit.join(train_deleted[cnames1])
test_logit = test_logit.join(test_deleted[cnames1])
```

In [71]:

```
##Create dummies for categorical variables in train data
cat_names = ["international plan", "voice mail plan"]

for i in cat_names:
    temp = pd.get_dummies(train_deleted[i], prefix = i)
    train_logit = train_logit.join(temp)
```

In [ ]:

```
train_logit.head()
```

In [73]:

```
##Create dummies for categorical variables in test data
cat_names = ["international plan", "voice mail plan"]

for i in cat_names:
    temp = pd.get_dummies(test_deleted[i], prefix = i)
    test_logit = test_logit.join(temp)
```

In [ ]:

```
test_logit.head()
```

In [75]:
```
train_logit.shape
```

Out[75]:
```
(2797, 16)
```

In [76]:
```
#select column indexes for independent variables
train_cols = train_logit.columns[1:16]
```

In [ ]:
```
train_cols
```

In [ ]:
```
#Built Logistic Regression
np.warnings.filterwarnings("ignore")
import statsmodels.api as sm

logit = sm.Logit(train_logit['Churn'], train_logit[train_cols]).fit()
```

In [ ]:
```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
logit.summary()
```

In [80]:
```
#Predict test data
test_logit['Actual_prob'] = logit.predict(test_logit[train_cols])
```

In [ ]:
```
test_logit.head(10)
```

In [82]:
```
#If value of probability is more than 0.5 then assign 1 or if less than 0.5 th
en assifn 0 in the new variable
test_logit['ActualVal'] = 1
test_logit.loc[test_logit.Actual_prob < 0.5, 'ActualVal'] = 0
```

In [ ]:
```
test_logit.head()
```

In [84]:
```
#Build confusion matrix
CM = pd.crosstab(test_logit['Churn'], test_logit['ActualVal'])
```

In [85]:
```
CM
```

Out[85]:

ActualVal	**0**	**1**
Churn		
0	1424	19
1	164	60

In [86]:
```
#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

In [87]:
```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
```

Out[87]:
```
89.02219556088782
```

In [88]:
```
#FNR
(FN*100)/(FN+TP)
```

Out[88]:
```
73.21428571428571
```

In [89]:
```
submission_LogisticRegression_Python = pd.concat([test,test_logit['ActualVal'
]], axis=1)
```

In [90]:
```
submission_LogisticRegression_Python['ActualVal'] = "False."
submission_LogisticRegression_Python.loc[submission_LogisticRegression_Python.
ActualVal == 1,'ActualVal'] = "True."
```

In [91]:
```
submission_LogisticRegression_Python = submission_LogisticRegression_Python.re
name(columns = {'ActualVal':'Prediction'})
```

In [ ]:
```
submission_LogisticRegression_Python.head()
```

In [93]:
```
# Writting a csv output
submission_LogisticRegression_Python.to_csv("submission_LogisticRegression_Pyt
hon.csv",index=False)
```

NAIVE BAYES

In [23]:
```
train_deleted['international plan'] = train_deleted['international plan'].repl
ace('no', 0)
train_deleted['international plan'] = train_deleted['international plan'].repl
ace('yes', 1)

train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('n
o', 0)
train_deleted['voice mail plan'] = train_deleted['voice mail plan'].replace('y
es', 1)

test_deleted['international plan'] = test_deleted['international plan'].replac
e('no', 0)
test_deleted['international plan'] = test_deleted['international plan'].replac
e('yes', 1)

test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('no'
, 0)
test_deleted['voice mail plan'] = test_deleted['voice mail plan'].replace('ye
s', 1)
```

In [24]:
```
#converting into numpy array
X_train = train_deleted.values[:, 0:13]
y_train = train_deleted.values[:,13]
X_test =  test_deleted.values[:, 0:13]
y_test =  test_deleted.values[:,13]
```

In [25]:
```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB

#Naive Bayes implementation
NB_model = GaussianNB().fit(X_train, y_train)
```

In [26]:
```
#predict test cases
NB_Predictions = NB_model.predict(X_test)
```

In [27]:
```
#Build confusion matrix
CM = pd.crosstab(y_test, NB_Predictions)
```

In [128]:
```
CM
```

Out[128]:

col_0	0.0	1.0
row_0		
0.0	1365	78
1.0	147	77

In [28]:
```
#Let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

In [29]:
```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
```

Out[29]:
```
86.50269946010798
```

In [30]:
```
#False Negative rate
(FN*100)/(FN+TP)
```

Out[30]:
```
65.625
```

In [31]:
```
pred = pd.DataFrame(NB_Predictions)
```

In [32]:
```
pred = pred.rename(columns = {0:'Prediction'})
```

In [33]:
```
submission_NaiveBayes_python =  pd.concat([test,pred], axis=1)
```

In [34]:

```
# Writting a csv output
submission_NaiveBayes_python.to_csv("submission_NaiveBayes_python.csv",index=False)
```