globsyn

Taking People To The Next Level

Globsyn finishing school, Globsyn Crystals, 1st Floor, XI – 11 and 12, Block EP, Sector V, Salt Lake,
Kolkata – 700091

PREDICTION OF CUSTOMER CHURNING IN TELECOM INDUSTRY

Group Members:

Sandipa Bhowmick, CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT, 151650110104

Sohom Banerjee, CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT, 151650110113

Soumita Dutta, CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT, 151650110115

Souvik Datta, CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT, 151650110117

Souvik Mitra, CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT, 151650110118

Swarup Kumar Das, KALYANI GOVERNMENT ENGINEERING COLLEGE, 151020120015

Table of Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, **Prof. Arnab Chakraborty** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

(Sandipa Bhowmick,

Sohom Banerjee,

Soumita Dutta,

Souvik Datta,

Souvik Mitra,

Swarup Kumar Das)

Project Objective

In this project we have a public dataset that consists of a customer usage pattern and the churning status of the customer.

Target Variable:

Churn: if the customer has churned (1=yes; 0 = no)

The predictors influencing the churn score are as follows:

- Account Length
- International Plan
- Voice mail plan
- Number of voice mail messages
- Total day minutes used
- Day calls made
- Total day charge
- Total evening minutes
- Total night calls
- Total night charge
- Total international minutes used
- Total international calls made
- Total international charge
- Number of Customer Service calls made

Goal:

- Predict the whether there will be customer churn based on the other features.
- Prepare dataset and perform **K-Fold Cross Validation**.
- Create three different types of models from the data – **Naive Bayes Classifier**, **KNN Classifier** and **Random Forest Classifier** based on the training set.
- Apply on the test set and **compare the differences in the accuracies** of the different models
- **Write the KNN algorithm** in Python from scratch, and apply it again on the data. Compute the accuracy and the confusion matrix.

Project Scope

The broad scope of the **Prediction of Customer Churning in Telecom Industry** project includes:

- In this project we analysed a dataset of customer usage pattern in a certain telecom enterprise. The dataset consists of various other factors that influence both the customer usage and customer churn.
- By this project we can predict the churn; we cannot provide any solution to prevent the churning.

Requirement Specification

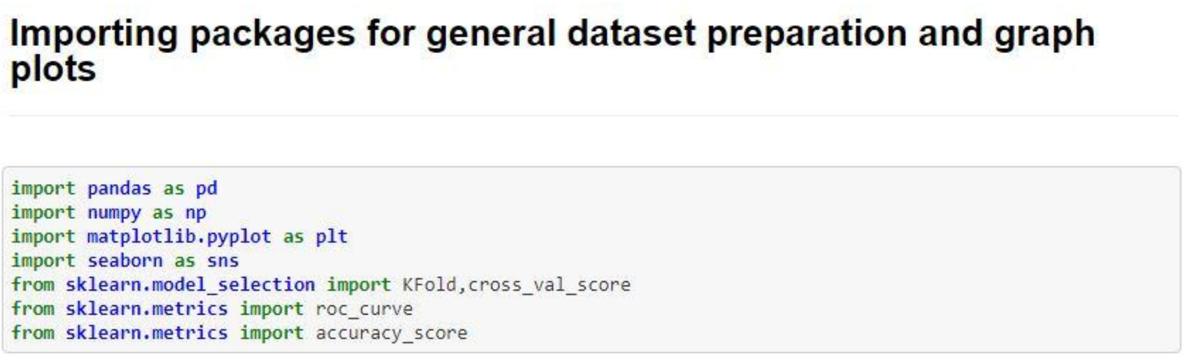
Hardware requirements:

- CPU: Dual core 64-bit 2.8 GHz 8.00 GT/s CPUs
- RAM: 2 GB RAM (recommended 4 GB RAM)
- Storage: 2 GB for installation of Anaconda Navigator.
- Internet access to download the files from Anaconda Cloud or a USB drive containing all of the files you need with alternate instructions for air gapped installations.

Software requirements:

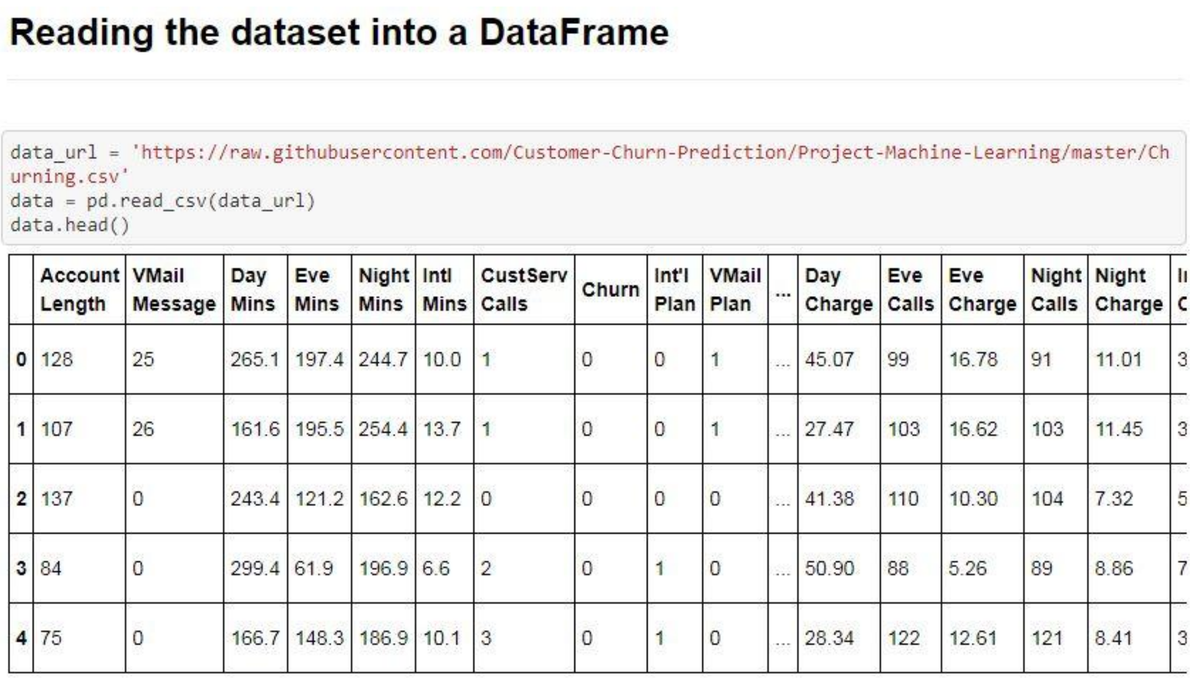
- Anaconda Navigator v3.6.4
- SciKit learn package for Python
- Any web browser like Google Chrome

Screenshots



Importing packages for general dataset preparation and graph plots

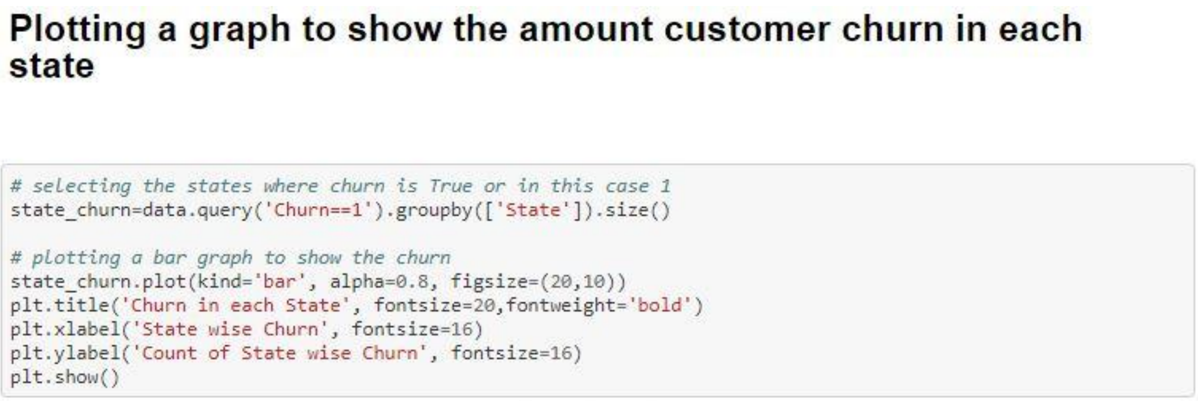
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold,cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score
```



Reading the dataset into a DataFrame

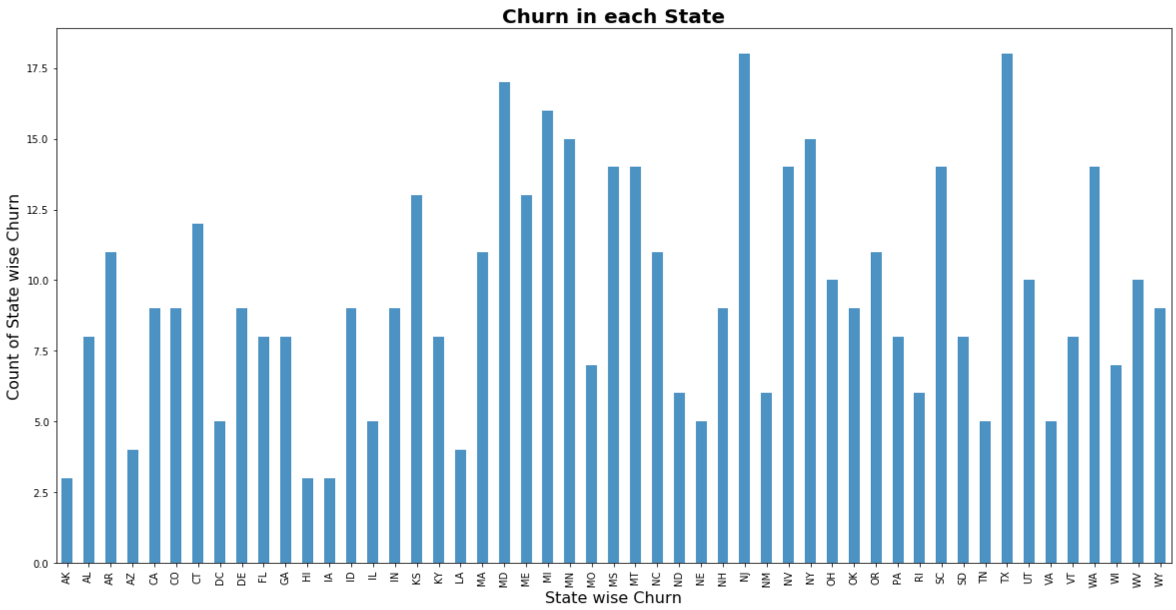
```
data_url = 'https://raw.githubusercontent.com/Customer-Churn-Prediction/Project-Machine-Learning/master/Ch
urning.csv'
data = pd.read_csv(data_url)
data.head()
```

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Int'l Plan	VMail Plan	...	Day Charge	Eve Calls	Eve Charge	Night Calls	Night Charge	I C
0	128	25	265.1	197.4	244.7	10.0	1	0	0	1	...	45.07	99	16.78	91	11.01	3
1	107	26	161.6	195.5	254.4	13.7	1	0	0	1	...	27.47	103	16.62	103	11.45	3
2	137	0	243.4	121.2	162.6	12.2	0	0	0	0	...	41.38	110	10.30	104	7.32	5
3	84	0	299.4	61.9	196.9	6.6	2	0	1	0	...	50.90	88	5.26	89	8.86	7
4	75	0	166.7	148.3	186.9	10.1	3	0	1	0	...	28.34	122	12.61	121	8.41	3

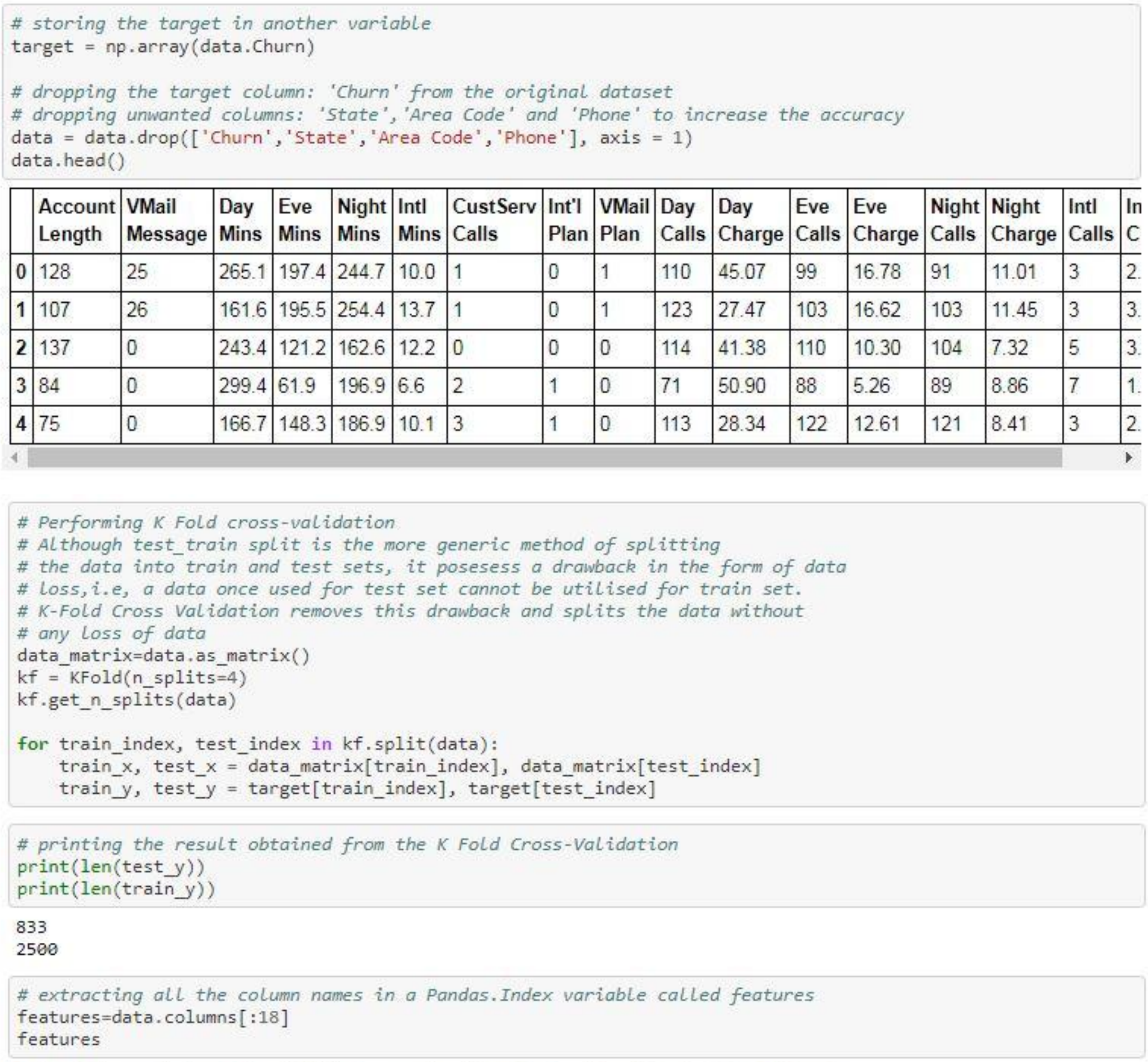


Plotting a graph to show the amount customer churn in each state

```
# selecting the states where churn is True or in this case 1
state_churn=data.query('Churn==1').groupby(['State']).size()

# plotting a bar graph to show the churn
state_churn.plot(kind='bar', alpha=0.8, figsize=(20,10))
plt.title('Churn in each State', fontsize=20,fontweight='bold')
plt.xlabel('State wise Churn', fontsize=16)
plt.ylabel('Count of State wise Churn', fontsize=16)
plt.show()
```



Dataset preparation



```
# storing the target in another variable
target = np.array(data.Churn)

# dropping the target column: 'Churn' from the original dataset
# dropping unwanted columns: 'State','Area Code' and 'Phone' to increase the accuracy
data = data.drop(['Churn','State','Area Code','Phone'], axis = 1)
data.head()
```

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Int'l Plan	VMail Plan	Day Calls	Day Charge	Eve Calls	Eve Charge	Night Calls	Night Charge	Intl Calls	In C
0	128	25	265.1	197.4	244.7	10.0	1	0	1	110	45.07	99	16.78	91	11.01	3	2.
1	107	26	161.6	195.5	254.4	13.7	1	0	1	123	27.47	103	16.62	103	11.45	3	3.
2	137	0	243.4	121.2	162.6	12.2	0	0	0	114	41.38	110	10.30	104	7.32	5	3.
3	84	0	299.4	61.9	196.9	6.6	2	1	0	71	50.90	88	5.26	89	8.86	7	1.
4	75	0	166.7	148.3	186.9	10.1	3	1	0	113	28.34	122	12.61	121	8.41	3	2.

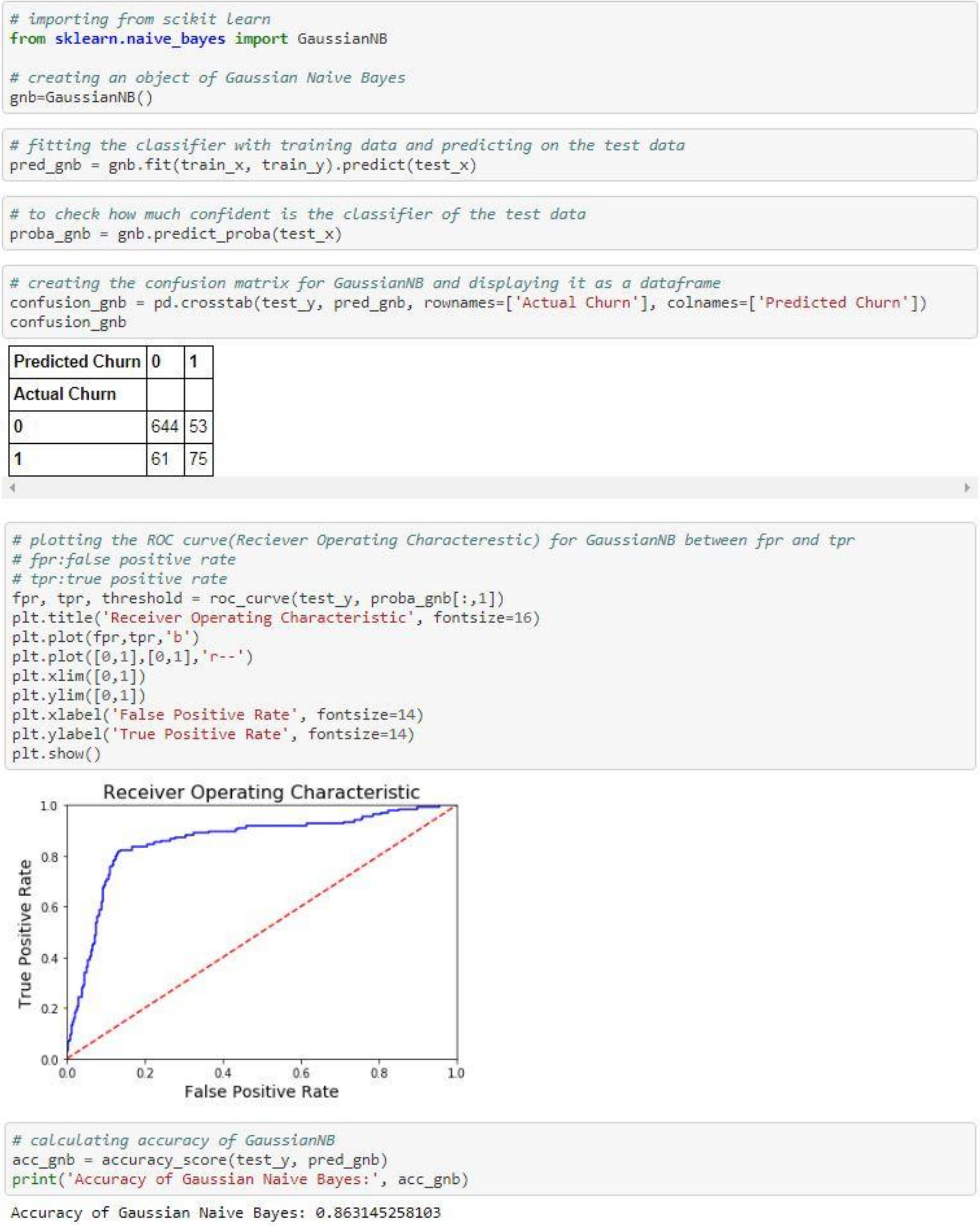
```
# Performing K Fold cross-validation
# Although test_train split is the more generic method of splitting
# the data into train and test sets, it posesess a drawback in the form of data
# loss,i.e, a data once used for test set cannot be utilised for train set.
# K-Fold Cross Validation removes this drawback and splits the data without
# any loss of data
data_matrix=data.as_matrix()
kf = KFold(n_splits=4)
kf.get_n_splits(data)

for train_index, test_index in kf.split(data):
    train_x, test_x = data_matrix[train_index], data_matrix[test_index]
    train_y, test_y = target[train_index], target[test_index]
```

```
# printing the result obtained from the K Fold Cross-Validation
print(len(test_y))
print(len(train_y))
```

```
833
2500
```

```
# extracting all the column names in a Pandas.Index variable called features
features=data.columns[:18]
features
```

Applying Gaussian Naive Bayes Classifier on the dataset



```
# importing from scikit learn
from sklearn.naive_bayes import GaussianNB

# creating an object of Gaussian Naive Bayes
gnb=GaussianNB()
```

```
# fitting the classifier with training data and predicting on the test data
pred_gnb = gnb.fit(train_x, train_y).predict(test_x)
```

```
# to check how much confident is the classifier of the test data
proba_gnb = gnb.predict_proba(test_x)
```

```
# creating the confusion matrix for GaussianNB and displaying it as a dataframe
confusion_gnb = pd.crosstab(test_y, pred_gnb, rownames=['Actual Churn'], colnames=['Predicted Churn'])
confusion_gnb
```

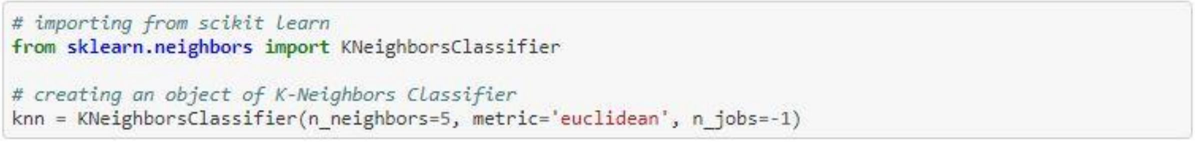
Predicted Churn	0	1
Actual Churn		
0	644	53
1	61	75

```
# plotting the ROC curve(Reciever Operating Characterestic) for GaussianNB between fpr and tpr
# fpr:false positive rate
# tpr:true positive rate
fpr, tpr, threshold = roc_curve(test_y, proba_gnb[:,1])
plt.title('Receiver Operating Characteristic', fontsize=16)
plt.plot(fpr,tpr,'b')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.show()
```

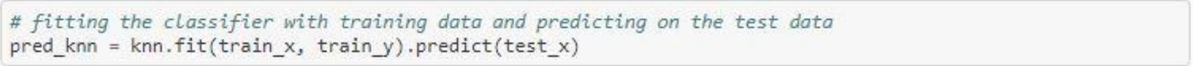
```
# calculating accuracy of GaussianNB
acc_gnb = accuracy_score(test_y, pred_gnb)
print('Accuracy of Gaussian Naive Bayes:', acc_gnb)
```

```
Accuracy of Gaussian Naive Bayes: 0.863145258103
```

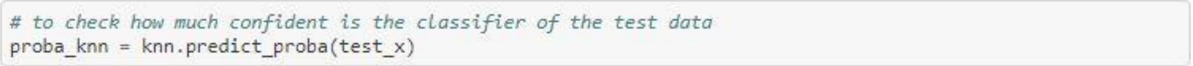
Applying k-Nearest Neighbors Classifier on the dataset



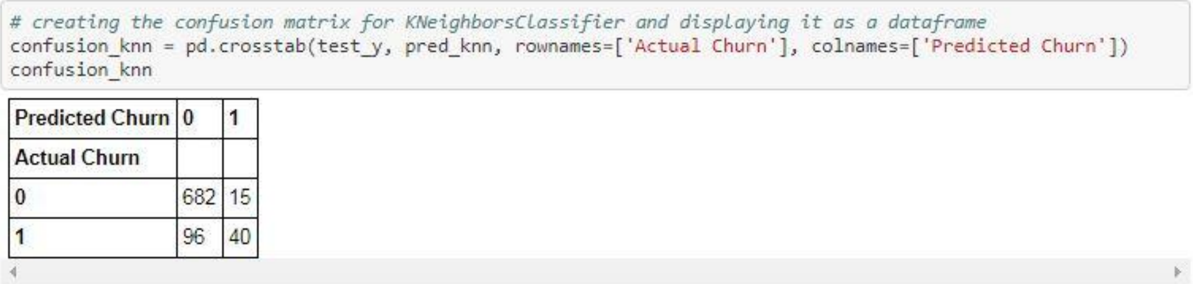
```
# importing from scikit learn
from sklearn.neighbors import KNeighborsClassifier

# creating an object of K-Neighbors Classifier
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean', n_jobs=-1)
```



```
# fitting the classifier with training data and predicting on the test data
pred_knn = knn.fit(train_x, train_y).predict(test_x)
```

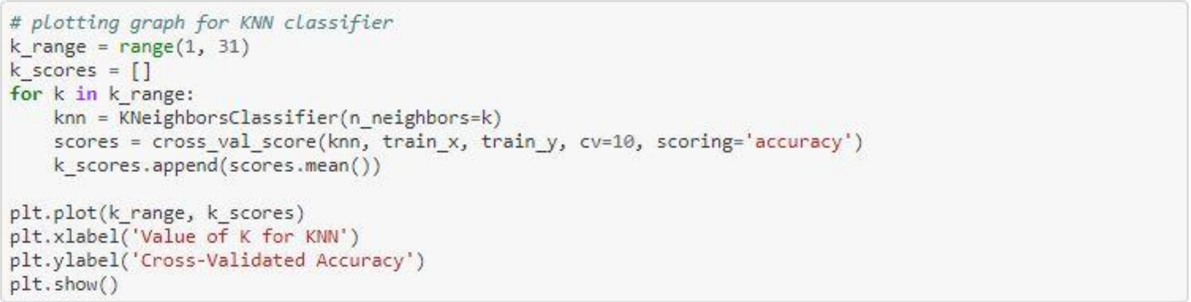


```
# to check how much confident is the classifier of the test data
proba_knn = knn.predict_proba(test_x)
```

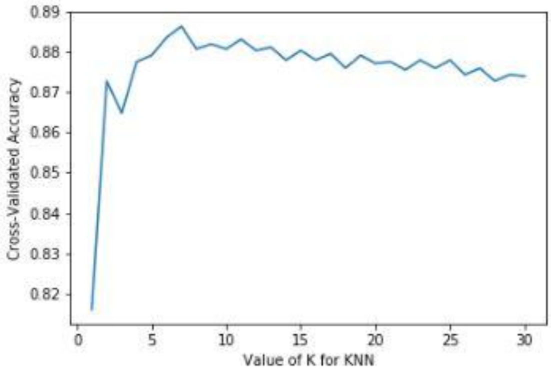


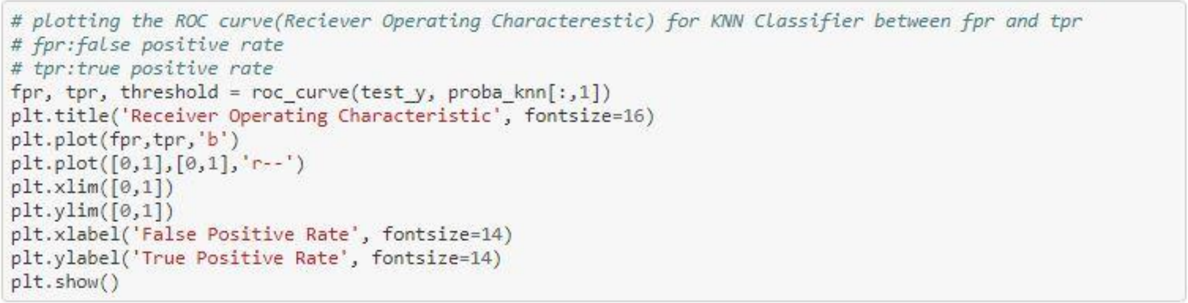
```
# creating the confusion matrix for KNeighborsClassifier and displaying it as a dataframe
confusion_knn = pd.crosstab(test_y, pred_knn, rownames=['Actual Churn'], colnames=['Predicted Churn'])
confusion_knn
```

Predicted Churn	0	1
Actual Churn		
0	682	15
1	96	40

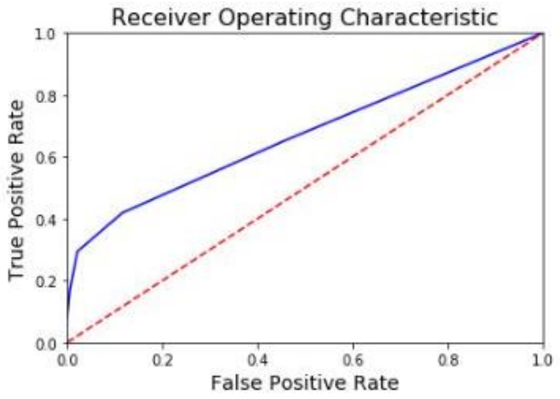


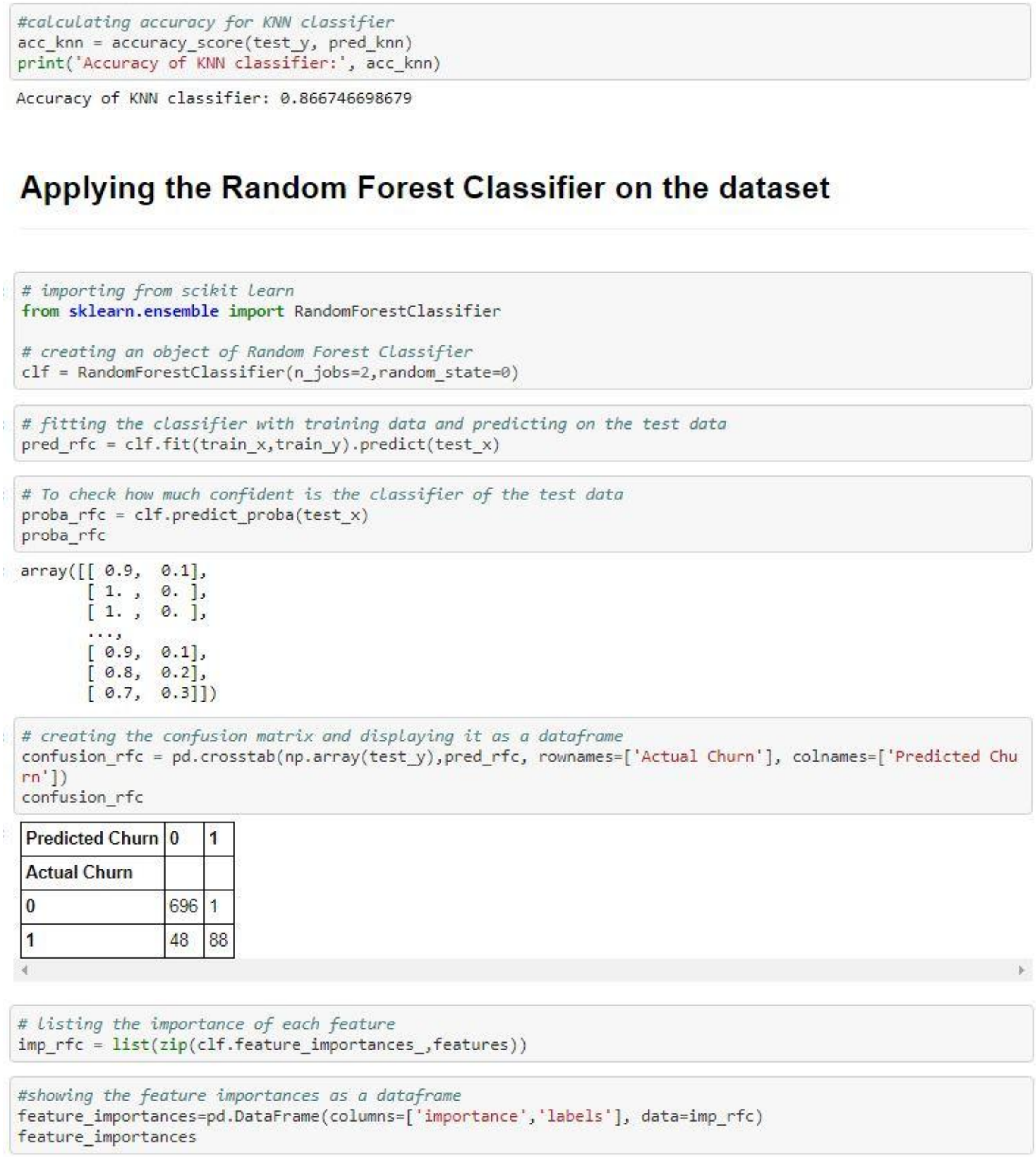
```
# plotting graph for KNN classifier
k_range = range(1, 31)
k_scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, train_x, train_y, cv=10, scoring='accuracy')
    k_scores.append(scores.mean())

plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```





```
# plotting the ROC curve(Reciever Operating Characterestic) for KNN Classifier between fpr and tpr
# fpr:false positive rate
# tpr:true positive rate
fpr, tpr, threshold = roc_curve(test_y, proba_knn[:,1])
plt.title('Receiver Operating Characteristic', fontsize=16)
plt.plot(fpr,tpr,'b')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.show()
```





```
#calculating accuracy for KNN classifier
acc_knn = accuracy_score(test_y, pred_knn)
print('Accuracy of KNN classifier:', acc_knn)
```

```
Accuracy of KNN classifier: 0.866746698679
```

Applying the Random Forest Classifier on the dataset

```
# importing from scikit learn
from sklearn.ensemble import RandomForestClassifier

# creating an object of Random Forest Classifier
clf = RandomForestClassifier(n_jobs=2,random_state=0)
```

```
# fitting the classifier with training data and predicting on the test data
pred_rfc = clf.fit(train_x,train_y).predict(test_x)
```

```
# To check how much confident is the classifier of the test data
proba_rfc = clf.predict_proba(test_x)
proba_rfc
```

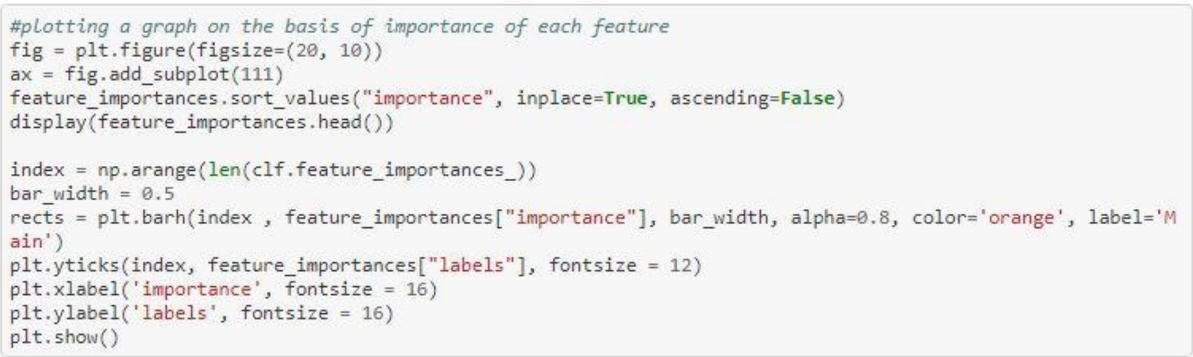
```
array([[ 0.9,  0.1],
       [ 1. ,  0. ],
       [ 1. ,  0. ],
       ...,
       [ 0.9,  0.1],
       [ 0.8,  0.2],
       [ 0.7,  0.3]])
```

```
# creating the confusion matrix and displaying it as a dataframe
confusion_rfc = pd.crosstab(np.array(test_y),pred_rfc, rownames=['Actual Churn'], colnames=['Predicted Chu
rn'])
confusion_rfc
```

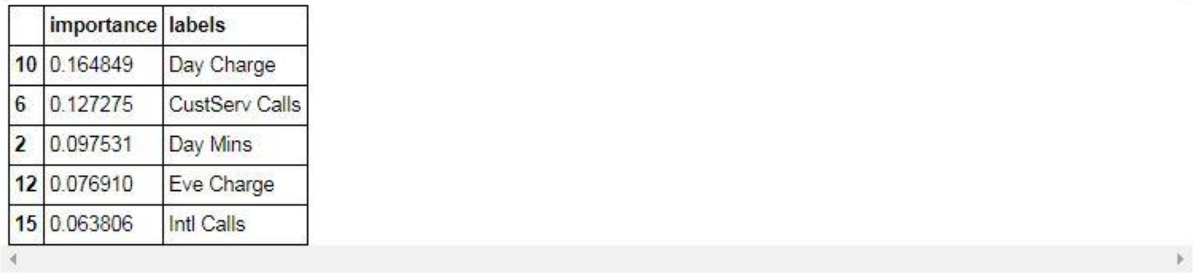
Predicted Churn	0	1
Actual Churn		
0	696	1
1	48	88

```
# listing the importance of each feature
imp_rfc = list(zip(clf.feature_importances_,features))
```

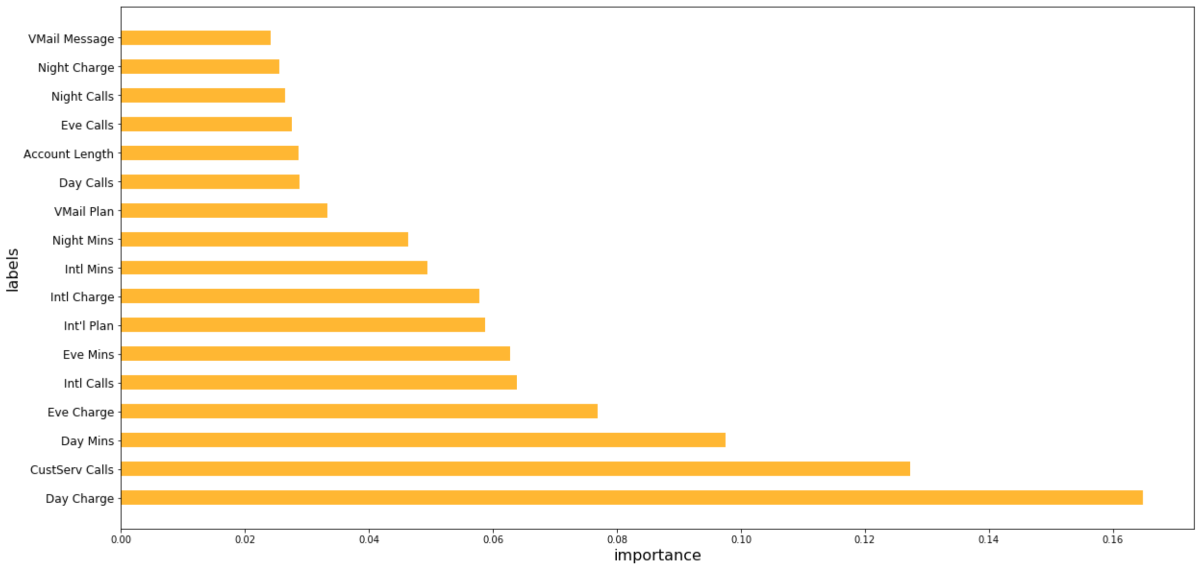
```
#showing the feature importances as a dataframe
feature_importances=pd.DataFrame(columns=['importance','labels'], data=imp_rfc)
feature_importances
```

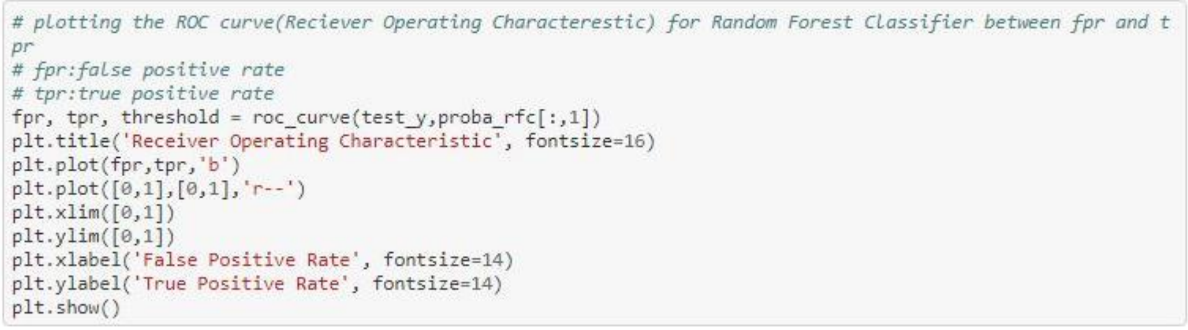


```
#plotting a graph on the basis of importance of each feature
fig = plt.figure(figsize=(20, 10))
ax = fig.add_subplot(111)
feature_importances.sort_values("importance", inplace=True, ascending=False)
display(feature_importances.head())

index = np.arange(len(clf.feature_importances_))
bar_width = 0.5
rects = plt.barh(index , feature_importances["importance"], bar_width, alpha=0.8, color='orange', label='M
ain')
plt.yticks(index, feature_importances["labels"], fontsize = 12)
plt.xlabel('importance', fontsize = 16)
plt.ylabel('labels', fontsize = 16)
plt.show()
```

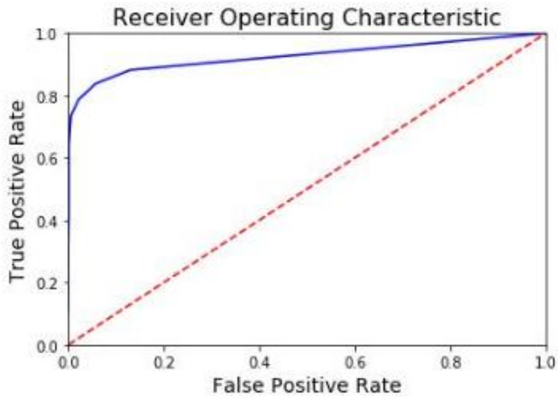


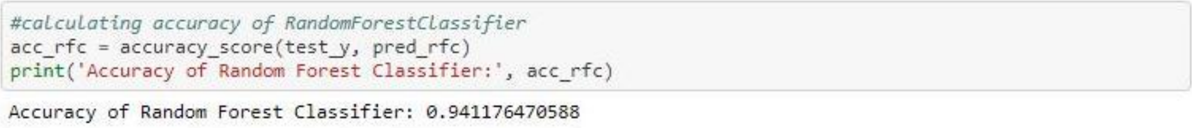
	importance	labels
10	0.164849	Day Charge
6	0.127275	CustServ Calls
2	0.097531	Day Mins
12	0.076910	Eve Charge
15	0.063806	Intl Calls





```
# plotting the ROC curve(Reciever Operating Characterestic) for Random Forest Classifier between fpr and t
pr
# fpr:false positive rate
# tpr:true positive rate
fpr, tpr, threshold = roc_curve(test_y,proba_rfc[:,1])
plt.title('Receiver Operating Characteristic', fontsize=16)
plt.plot(fpr,tpr,'b')
plt.plot([0,1],[0,1],'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.show()
```

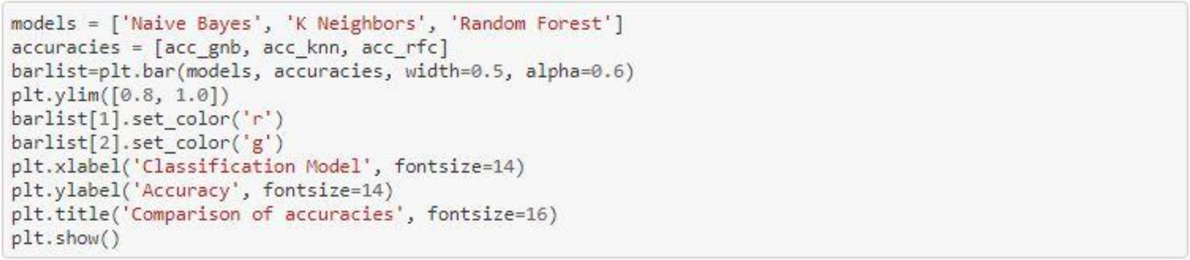




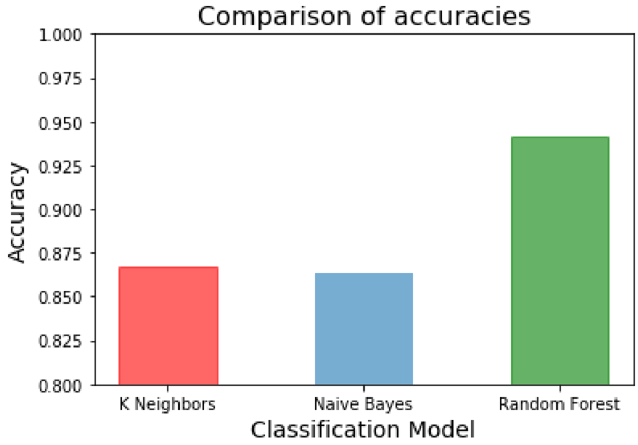
```
#calculating accuracy of RandomForestClassifier
acc_rfc = accuracy_score(test_y, pred_rfc)
print('Accuracy of Random Forest Classifier:', acc_rfc)
```

```
Accuracy of Random Forest Classifier: 0.941176470588
```

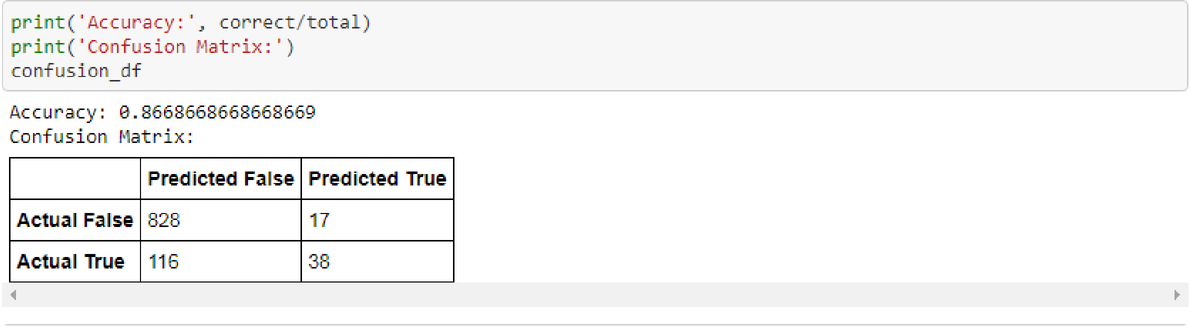
Plotting a bar graph of accuracies of the three classifier models



```
models = ['Naive Bayes', 'K Neighbors', 'Random Forest']
accuracies = [acc_gnb, acc_knn, acc_rfc]
barlist=plt.bar(models, accuracies, width=0.5, alpha=0.6)
plt.ylim([0.8, 1.0])
barlist[1].set_color('r')
barlist[2].set_color('g')
plt.xlabel('Classification Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)
plt.title('Comparison of accuracies', fontsize=16)
plt.show()
```



Computed accuracy and confusion matrix from the KNN algorithm written from scratch



```
print('Accuracy:', correct/total)
print('Confusion Matrix:')
confusion_df
```

```
Accuracy: 0.8668668668668669
Confusion Matrix:
```

	Predicted False	Predicted True
Actual False	828	17
Actual True	116	38

Code

Given below is the working code for KNN algorithm:

```
# all imports required
import numpy as np
from collections import Counter
import pandas as pd
import random

# defining the k nearest neighbors classifier algorithm
# we pass the training data and the test data as list
# we pass k number of neighbors with 3 neighbors as default
def k_neighbors_classifier(train_set, test_set, k=3):

   # declaring a list for calculating and storing euclidean distances
   distances = []

   # calculating the euclidean distances and storing it in the list 'distances'
   for group in train_set:
      for features in train_set[group]:
         euclidean_dist = np.linalg.norm(np.array(features)-np.array(test_set))

         # calculated distances are stored in the list along with their respective groups
         distances.append([euclidean_dist, group])

   # sorting the 'distances' and storing the names of the k nearest neighboring groups in a list
   neighbors = [i[1] for i in sorted(distances)[:k]]

   # prediction of the most common neighbor and finding the probability of the prediction
   # the most_common() method stores the data in a list of a tuple,
   # in this case we could write the tuple of list as: [(group_name, frequency)]
   # therefore we choose the first element as the predicted group
   prediction = Counter(neighbors).most_common(1)[0][0]

   # we know k is the total no. of neighbors
   # so we find the probability of our predicted group by division: (frequency / k)
   probability = Counter(neighbors).most_common(1)[0][1] / k

   return prediction, probability

# url of the source file
url = 'https://raw.githubusercontent.com/Customer-Churn-Prediction/Project-Machine-Learning/master/Churning.csv'

# reading the source file and storing it in a pandas DataFrame
df = pd.read_csv(url)
```

```
# the 'Churn' column is not the last column in the original dataset
# appending the 'Churn' data i.e the target containing the two groups 0 and 1 as the last
column
# storing the 'Churn' in another DataFrame variable
target = df['Churn']

# deleting unnecessary columns and the target column from the DataFrame
df.drop(['Churn', 'Phone', 'State','Area Code'], axis = 1, inplace=True)

# now we append the target data as the last column
df['Churn'] = target

# storing the datas present in the DataFrame as list
data_list = df.astype(float).values.tolist()

# shuffling the data so that the training and testing data can be chosen at random
random.shuffle(data_list)

# setting the percentage of test data as 30%
test_size = 0.3

# train_set and test_set are two dictionaries and their keys represent the two groups or
classes
# here it is 0(churn = False) or 1(churn = True)
train_set, test_set = {0:[], 1:[]}, {0:[], 1:[]}

# splitting the data into train and test sets
# selecting from the beginning upto the last 30% of data i.e. the first 70%
train_data = data_list[:-int(test_size*len(data_list))]

# selecting the last 30% of the data
test_data = data_list[-int(test_size*len(data_list)):]
# storing the grouping the data according to their group or class 0 or 1
for i in train_data:
   train_set[i[-1]].append(i[:-1])
for i in test_data:
   test_set[i[-1]].append(i[:-1])

correct = 0
total = 0
confusion_matrix = [[0, 0], [0, 0]]

# calculating the accuracy and computing the confusion matrix
for group in test_set:
   for data in test_set[group]:
        vote,confidence = k_neighbors_classifier(train_set, data, k=5)
        if group == vote:
            correct += 1
            confusion_matrix[group][group] += 1
        else:
            confusion_matrix[group][vote] += 1
        total += 1

confusion_df = pd.DataFrame(data=confusion_matrix,
                            columns=['Predicted False', 'Predicted True'],
                            index = ['Actual False', 'Actual True'])

print('Accuracy:', correct/total)
print('Confusion Matrix:')
```

Output:

```
Accuracy: 0.8668668668668669
Confusion Matrix:
              Predicted False  Predicted True
Actual False              828              17
Actual True               116              38
```

Certificate

This is to certify that Ms. Sandipa Bhowmick of Calcutta Institute of Engineering and Management, registration number: 151650110104, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Sohom Banerjee of Calcutta Institute of Engineering and Management, registration number: 151650110113, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Ms. Soumita Dutta of Calcutta Institute of Engineering and Management, registration number: 151650110115, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Souvik Datta of Calcutta Institute of Engineering and Management, registration number: 151650110117, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Souvik Mitra of Calcutta Institute of Engineering and Management, registration number: 151650110118, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School

Certificate

This is to certify that Mr. Swarup Kumar Das of Kalyani Government Engineering College, registration number: 151020120015, has successfully completed a project on Prediction of Customer Churning in Telecom Industry using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Globsyn Finishing School