CODE

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May  1 02:11:37 2018

@author: deepikakanade
"""
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import MinMaxScaler,Imputer
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import f1_score, roc_auc_score, confusion_matrix
from imblearn.combine import SMOTEENN
from sklearn.decomposition import PCA
import warnings
import itertools
from sklearn.metrics import roc_curve, auc, classification_report

warnings.simplefilter('ignore')

def classifiers(name,rf,param_grid,x_train_scaled,y_train,x_test_scaled,y_test):
   CV_rf = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)
   CV_rf.fit(x_train_scaled, y_train)

   y_pred_train = CV_rf.predict(x_train_scaled)
   y_pred_test = CV_rf.predict(x_test_scaled)

   print(' ')
   print('---------'+name+' Classifier--------')

   #Accuracy score
   print(name+" Train Accuracy:
{0:.3f}".format(float((y_pred_train==y_train).sum())/float(len(y_train))))
```

```
    print(name+" Test Accuracy:
{0:.3f}".format(float((y_pred_test==y_test).sum())/float(len(y_test))))

    #Classification Report
    print(name+" Classification Report: ")
    print(classification_report(y_test, y_pred_test))

    #F1-Score
    print(name+" Training F1 Score: {0:.3f}".format(f1_score(y_train, y_pred_train,
average='weighted')))
    print(name+" Testing F1 Score: {0:.3f}".format(f1_score(y_test, y_pred_test,
average='weighted')))

    #AUC score
    AUC_score_train=roc_auc_score(y_train, y_pred_train)
    AUC_score_test=roc_auc_score(y_test, y_pred_test)
    print(name+" Training AUC Score: {0:.3f}".format(AUC_score_train))
    print(name+" Testing AUC Score: {0:.3f}".format(AUC_score_test))


    #Confusion Matrix for test data
    conf = confusion_matrix(y_test,y_pred_test).ravel()
    plt.figure()
    plt.title(name+' :Confusion Matrix')
    conf=np.reshape(conf,(2,2))
    plt.imshow(conf, cmap=plt.cm.Blues, interpolation='nearest')
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, ['Predicted No','Predicted Yes'])
    plt.yticks(tick_marks, ['Actual No','Actual Yes'], rotation='vertical')
    thresh = conf.max() / 2.
    for i, j in itertools.product(range(conf.shape[0]), range(conf.shape[1])):
        plt.text(j, i, conf[i, j],horizontalalignment="center",color="white" if conf[i, j] > thresh else
"black")
    plt.show()

    #Visualize ROC-Curve for test data
    fpr, tpr, threshold = roc_curve(y_test, y_pred_test)
    plt.figure()
    plt.plot(fpr,tpr,label="data 1, auc="+str(AUC_score_test))
    plt.xlabel('False Positive Rate (fpr)')
    plt.ylabel('True Positive Rate (tpr)')
    plt.title('Receiver operating characteristics (ROC): '+name)
    plt.legend(loc=4)
    plt.show()


################Loading of data from csv into dataframe#############################
train=('/Users/deepikakanade/Desktop/bank-additional.csv')
df=pd.read_csv(train,sep=',')
print(' ')
print('---------Loading of data is done-------')

############## Visualize the data by plotting box plots ################
#Visualize Numerical Data
numerical = ['age','campaign','previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx',
'euribor3m', 'nr.employed', 'pdays']
for i in numerical:
    df[i].value_counts().plot(kind='hist',figsize = (10, 6),title='Histogram plot of '+i)
    plt.figure()
    plt.show()

#Visualizing Numerical Data using Box Plot to search for outliers
for i in numerical:
    plt.boxplot(df[i])
    plt.figure()
    plt.show()
    plt.title('Box plot of: '+i)


print(' ')
print('---------Visualization of data is done----------')

#Removing 2 features from the dataset
df=df.drop(labels=['pdays','default'],axis=1)

df = df[df['education'].str.contains("illiterate")==False]

#Remove Outliers form the feature: age, campaign and cons.conf.idx
df = df[df['age'] < 69]
df = df[df['campaign'] < 20]
df = df[df['cons.conf.idx'] < -29]

############# Label Encoding the features ######
df.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'),(1,2,3,4,5,6,7,
8,9,10,11,12),inplace=True)
df.day_of_week.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),inplace=True)
df.y.replace(('yes','no'),(1,0),inplace=True)
df.job.replace(('admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management',
      'retired', 'self-employed', 'services', 'student', 'technician',
      'unemployed', 'unknown'),(1,2,3,4,5,6,7,8,9,10,11,12),inplace=True)
df.marital.replace(('divorced', 'married', 'single', 'unknown'),(1,2,3,4),inplace=True)
df.education.replace(('basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate',
      'professional.course', 'university.degree', 'unknown'),(1,2,3,4,5,6,7,8),inplace=True)
df.housing.replace(('no', 'unknown', 'yes'),(1,2,3),inplace=True)
df.loan.replace(('no', 'unknown', 'yes'),(1,2,3),inplace=True)

#Bifurcarting features and labels
y=df['y']
X=df.drop('y',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

#Imputing mode value in 5 categorical features for train and test data
imp = Imputer(missing_values=12,strategy='most_frequent',axis=0)
X_train['job']=imp.fit_transform(X_train['job'].values.reshape(-1,1))
imp = Imputer(missing_values=12,strategy='most_frequent',axis=0)
X_test['job']=imp.fit_transform(X_test['job'].values.reshape(-1,1))

imp = Imputer(missing_values=4,strategy='most_frequent',axis=0)
X_train['marital']=imp.fit_transform(X_train['marital'].values.reshape(-1,1))
imp = Imputer(missing_values=4,strategy='most_frequent',axis=0)
X_test['marital']=imp.fit_transform(X_test['marital'].values.reshape(-1,1))

imp = Imputer(missing_values=8,strategy='most_frequent',axis=0)
X_train['education']=imp.fit_transform(X_train['education'].values.reshape(-1,1))
imp = Imputer(missing_values=8,strategy='most_frequent',axis=0)
X_test['education']=imp.fit_transform(X_test['education'].values.reshape(-1,1))

imp = Imputer(missing_values=2,strategy='most_frequent',axis=0)
X_train['housing']=imp.fit_transform(X_train['housing'].values.reshape(-1,1))
imp = Imputer(missing_values=2,strategy='most_frequent',axis=0)
X_test['housing']=imp.fit_transform(X_test['housing'].values.reshape(-1,1))

imp = Imputer(missing_values=2,strategy='most_frequent',axis=0)
X_train['loan']=imp.fit_transform(X_train['loan'].values.reshape(-1,1))
imp = Imputer(missing_values=2,strategy='most_frequent',axis=0)
X_test['loan']=imp.fit_transform(X_test['loan'].values.reshape(-1,1))

#Replacing missing data by one hot encoding
one_hot_1=pd.get_dummies(df['job'],prefix='job')
X_train=X_train.join(one_hot_1)
```

```
X_train=X_train.drop('job',axis=1)

one_hot_test_1=pd.get_dummies(df['job'],prefix='job')
X_test=X_test.join(one_hot_test_1)
X_test=X_test.drop('job',axis=1)

one_hot_3=pd.get_dummies(df['marital'],prefix='marital')
X_train=X_train.join(one_hot_3)
X_train=X_train.drop('marital',axis=1)

one_hot_test_2=pd.get_dummies(df['marital'],prefix='marital')
X_test=X_test.join(one_hot_test_2)
X_test=X_test.drop('marital',axis=1)

one_hot_4=pd.get_dummies(df['education'],prefix='education')
X_train=X_train.join(one_hot_4)
X_train=X_train.drop('education',axis=1)

one_hot_test_4=pd.get_dummies(df['education'],prefix='education')
X_test=X_test.join(one_hot_test_4)
X_test=X_test.drop('education',axis=1)

one_hot_5=pd.get_dummies(df['housing'],prefix='housing')
X_train=X_train.join(one_hot_5)
X_train=X_train.drop('housing',axis=1)

one_hot_test_5=pd.get_dummies(df['housing'],prefix='housing')
X_test=X_test.join(one_hot_test_5)
X_test=X_test.drop('housing',axis=1)

one_hot_6=pd.get_dummies(df['loan'],prefix='loan')
X_train=X_train.join(one_hot_6)
X_train=X_train.drop('loan',axis=1)

one_hot_test_6=pd.get_dummies(df['loan'],prefix='loan')
X_test=X_test.join(one_hot_test_6)
X_test=X_test.drop('loan',axis=1)

one_hot=pd.get_dummies(df['contact'])
X_train=X_train.join(one_hot)
X_train=X_train.drop('contact',axis=1)

one_hot_test_7=pd.get_dummies(df['contact'],prefix='contact')
X_test=X_test.join(one_hot_test_7)
```

```
X_test=X_test.drop('contact',axis=1)


one_hot_9=pd.get_dummies(df['poutcome'])
X_train=X_train.join(one_hot_9)
X_train=X_train.drop('poutcome',axis=1)

one_hot_test_9=pd.get_dummies(df['poutcome'],prefix='poutcome')
X_test=X_test.join(one_hot_test_9)
X_test=X_test.drop('poutcome',axis=1)

#SMOTE analysis for oversampling
smote_enn=SMOTEENN(random_state=0)
X_oversampled,Y_oversampled=smote_enn.fit_sample(X_train,y_train)

print(' ')
print('----------Pre-processing of data is done--------')

#Standardizing the feature values
scaler = MinMaxScaler()
scaler.fit(X_oversampled)
#x_train_scaled_dummy=scaler.transform(X_oversampled)
#x_test_scaled_dummy=scaler.transform(X_test)
x_train_scaled=scaler.transform(X_oversampled)
x_test_scaled=scaler.transform(X_test)
print(' ')
print('---------Normalization of data is done-----------')

#Principal Component Analysis
'''
pca = PCA(n_components=13)
x_train_scaled=pca.fit_transform(x_train_scaled_dummy)

pca = PCA(n_components=13)
x_test_scaled=pca.fit_transform(x_test_scaled_dummy
'''
#Training on different Classifiers
clsr_names=["Random Forest", "SVM", "K Nearest Neighbors", "Naive Bayes"]

rf = RandomForestClassifier()
param_grid = {
   'n_estimators': range(1, 30),
   'max_features': ['auto', 'sqrt', 'log2'],
   }
```

```
classifiers(clsr_names[0],rf,param_grid,x_train_scaled,Y_oversampled,x_test_scaled,y_test)

rf=SVC()
param_grid = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
               'C': [1, 10, 100, 1000]},
              {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
classifiers(clsr_names[1],rf,param_grid,x_train_scaled,Y_oversampled,x_test_scaled,y_test)

rf = KNeighborsClassifier()
param_grid = {
    'n_neighbors': range(1, 30),
    }
classifiers(clsr_names[2],rf,param_grid,x_train_scaled,Y_oversampled,x_test_scaled,y_test)

rf = GaussianNB()
param_grid = {}
classifiers(clsr_names[3],rf,param_grid,x_train_scaled,Y_oversampled,x_test_scaled,y_test)
```