RBF Network for dataset

Code :-

```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import numpy as numpy
import math
import pandas as pd

Data= pd.read_table("bank.csv", sep= None, engine= "python")
cols= ["age","balance","day","duration","campaign","pdays","previous"]
data_encode= Data.drop(cols, axis= 1)
data_encode= data_encode.apply(LabelEncoder().fit_transform)
data_rest= Data[cols]
Data= pd.concat([data_rest,data_encode], axis= 1)
data_train, data_test= train_test_split(Data, test_size= 0.33, random_state= 4)

X_train= data_train.drop("y", axis= 1)
Y_train= data_train["y"]
X_test= data_test.drop("y", axis=1)
Y_test= data_test["y"]

scaler= StandardScaler()
scaler.fit(X_train)

X_train= scaler.transform(X_train)
X_test= scaler.transform(X_test)
K_cent= 8
km= KMeans(n_clusters= K_cent, max_iter= 100)
km.fit(X_train)
cent= km.cluster_centers_
max=0
for i in range(K_cent):
    for j in range(K_cent):
        d= numpy.linalg.norm(cent[i]-cent[j])
        if(d> max):
            max= d
d= max
sigma= d/math.sqrt(2*K_cent)
shape= X_train.shape
row= shape[0]
column= K_cent
G= numpy.empty((row,column), dtype= float)
```

```
for i in range(row):
    for j in range(column):
        dist= numpy.linalg.norm(X_train[i]-cent[j])
        G[i][j]= math.exp(-math.pow(dist,2)/math.pow(2*sigma,2))
GTG= numpy.dot(G.T,G)
GTG_inv= numpy.linalg.inv(GTG)
fac= numpy.dot(GTG_inv,G.T)
W= numpy.dot(fac,Y_train)
row= X_test.shape[0]
column= K_cent
G_test= numpy.empty((row,column), dtype= float)
for i in range(row):
    for j in range(column):
        dist= numpy.linalg.norm(X_test[i]-cent[j])
        G_test[i][j]= math.exp(-math.pow(dist,2)/math.pow(2*sigma,2))
prediction= numpy.dot(G_test,W)
prediction= 0.5*(numpy.sign(prediction-0.5)+1)
score= accuracy_score(prediction,Y_test)
print("Accuracy = ",score.mean())
```

Output :-

```
Accuracy =  0.8873994638069705
```

Performance metrics for RBF

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

print("Confusion Matrix : ")
confusion_matrix(Y_test,prediction)
```

```
Confusion Matrix :

array([[1308,   18],
       [ 150,   16]], dtype=int64)
```

```
print("Performace Metrics : ")
print(classification_report(Y_test,prediction))
```

```
Performace Metrics :
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      1326
           1       0.47      0.10      0.16       166

    accuracy                           0.89      1492
   macro avg       0.68      0.54      0.55      1492
weighted avg       0.85      0.89      0.85      1492
```

Performance metrics for MLP

```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(10,5), max_iter=100)

mlp.fit(X_train,Y_train)
predictions_test = mlp.predict(X_test)
print("Accuracy = ",accuracy_score(predictions_test,Y_test))
```

```
Accuracy =  0.8967828418230563
```

```
print("Confusion Matrix : ")
confusion_matrix(predictions_test,Y_test)
```

```
Confusion Matrix :

array([[1284,  112],
       [  42,   54]], dtype=int64)
```

```
print("Performace Metrics : ")
print(classification_report(prediction,Y_test))
```

```
Performace Metrics :
              precision    recall  f1-score   support

         0.0       0.99      0.90      0.94      1458
         1.0       0.10      0.47      0.16        34

    accuracy                           0.89      1492
   macro avg       0.54      0.68      0.55      1492
weighted avg       0.97      0.89      0.92      1492
```

Performance metrics for MLP is same as that of RBF but the time taken for training the data is much higher for MLP. Hence RBF is preferred