Exercise 8: Design and implement a radial basis function neural network to solve function approximation or regression problem.

```
import math
import pandas as pd
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
```

```
from google.colab import drive
drive.mount('/content/gdrive')
Data= pd.read_csv("gdrive/My Drive/ml/bank-full.csv")
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)
```

```
Mounted at /content/gdrive
```

```
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_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```

```
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 ('The accuracy of the RBF Neural Network is: ' , "{0:0.2f}".format(score.mean()*100), '%')
```

```
The accuracy of the RBF Neural Network is:  88.77 %
```

Start coding or generate with AI.