PCA - Face Feature Extraction

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement - Building a Machine Learning model to extract features of the face using PCA.

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

Dataset Used - LFW Dataset from sklearn library

1. Importing libraries and Dataset -

In [1]:
```
from sklearn.datasets import fetch_lfw_people ##Dataset
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
```

2. Exploratory Data Analysis -

In [2]:
```
dataset = fetch_lfw_people(resize=0.4, min_faces_per_person=70)
```

In [3]:
```
#Dataset
X = dataset.data
y = dataset.target
target_names = dataset.target_names
images = dataset.images
```

In [4]:
```
n,h,w = images.shape
print("No of Images - ",n)
print("Height - ",h)
print("Width - ",w)
```

```
No of Images -  1288
Height -  50
Width -  37
```

In [5]:
```
X.shape
```

Out[5]:
```
(1288, 1850)
```

In [6]:
```
len(target_names) ## No.of categories/classes
```

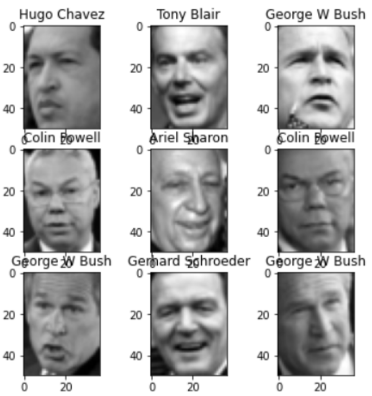
Out[6]:
```
7
```

In [7]:
```
np.unique(y, return_counts = True)
```

Out[7]:
```
(array([0, 1, 2, 3, 4, 5, 6], dtype=int64),
 array([ 77, 236, 121, 530, 109,  71, 144], dtype=int64))
```

3. Data Visualization -

In [9]:
```
def plot_grid(images, titles, h, w, rows=3, cols=3):
    plt.figure(figsize=(2*cols, 2 *rows))
    for i in range(rows*cols):
        plt.subplot(rows,cols,i+1)
        plt.imshow(images[i].reshape(h,w), cmap='gray')
        plt.title(target_names[titles[i]])
plot_grid(X,y,h,w)
```



4. Splitting the data -

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1) #Splitting the data
```

In [11]:
```
X_train.shape
```

Out[11]:
```
(1159, 1850)
```

5. PCA -

```
In [12]: p = PCA()
         p.fit(X_train)

Out[12]: PCA()

In [13]: p.transform(X_train).shape

Out[13]: (1159, 1159)

In [14]: var = p.explained_variance_
         print(var)
         com = p.components_
         print(com.shape)

[4.8989166e+05 3.9869944e+05 1.8790259e+05 ... 4.3856242e-01 4.2435777e-01
 2.8545223e-06]
(1159, 1850)

In [16]: val_sum = sum(p.explained_variance_)
         print(val_sum)
         sort_ind = np.argsort(var)
         sort_ind = sort_ind[::-1]

2607101.7939759726

In [17]: temp_sum = 0
         principle_vec = []
         principle_val = []
         i=0
         while(temp_sum < 0.98*val_sum):
             principle_vec.append(com[sort_ind[i]])
             principle_val.append(var[sort_ind[i]])
             temp_sum += var[sort_ind[i]]
             i += 1
         print("No of Components - ", i)

No of Components -  251

In [18]: principle_vec = np.matrix(principle_vec)

In [19]: print(principle_vec.shape)

(251, 1850)

In [20]: X_train_trans = np.dot(X_train, principle_vec.T)
         X_test_trans = np.dot(X_test, principle_vec.T)

In [21]: X_train_trans.shape

Out[21]: (1159, 251)
```

6. Training an MLP classifier -

```
In [28]: clf_t = MLPClassifier(hidden_layer_sizes=(512, ), batch_size=128, verbose=True, early_stopping=True)
         clf_t.fit(X_train_trans, y_train)

Iteration 1, loss = 11.69209498
Validation score: 0.387931
Iteration 2, loss = 8.36430909
Validation score: 0.431034
Iteration 3, loss = 6.15611128
Validation score: 0.508621
Iteration 4, loss = 3.61257026
Validation score: 0.732759
Iteration 5, loss = 2.32283689
Validation score: 0.663793
Iteration 6, loss = 1.42787377
Validation score: 0.818966
Iteration 7, loss = 0.89679108
Validation score: 0.810345
Iteration 8, loss = 0.49356899
Validation score: 0.750000
Iteration 9, loss = 0.21336988
Validation score: 0.836207
Iteration 10, loss = 0.12187043
Validation score: 0.741379
Iteration 11, loss = 0.09252084
Validation score: 0.836207
Iteration 12, loss = 0.06587441
Validation score: 0.836207
Iteration 13, loss = 0.04204553
Validation score: 0.827586
Iteration 14, loss = 0.01841497
Validation score: 0.801724
Iteration 15, loss = 0.00014825
Validation score: 0.827586
Iteration 16, loss = 0.00013642
Validation score: 0.827586
```

```
Iteration 17, loss = 0.00012832
Validation score: 0.836207
Iteration 18, loss = 0.00012817
Validation score: 0.836207
Iteration 19, loss = 0.00012690
Validation score: 0.836207
Iteration 20, loss = 0.00012580
Validation score: 0.836207
Validation score did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

Out[28]:
```
MLPClassifier(batch_size=128, early_stopping=True, hidden_layer_sizes=(512,),
              verbose=True)
```

7. Testing the MLP model on Test Data & Classification Report/Binary Confusion Matrix-

In [29]:
```python
y_pred = clf_t.predict(X_test_trans)
print(classification_report(y_test, y_pred, target_names = target_names))
```

```
                   precision    recall  f1-score   support

     Ariel Sharon       0.75      0.38      0.50         8
     Colin Powell       0.77      0.85      0.81        20
  Donald Rumsfeld       0.62      0.50      0.56        10
    George W Bush       0.83      0.93      0.88        58
Gerhard Schroeder       0.55      0.50      0.52        12
      Hugo Chavez       0.40      0.67      0.50         3
       Tony Blair       0.64      0.50      0.56        18

         accuracy                           0.74       129
        macro avg       0.65      0.62      0.62       129
     weighted avg       0.74      0.74      0.73       129
```

In [32]:
```python
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[ 3  2  0  1  1  0  1]
 [ 0 17  0  3  0  0  0]
 [ 0  0  5  3  1  0  1]
 [ 0  0  1 54  1  0  2]
 [ 0  0  2  0  6  3  1]
 [ 0  0  0  0  1  2  0]
 [ 1  3  0  4  1  0  9]]
```

8. Eigen Faces (Extracted Features) -

In [24]:
```python
n_components = 251
mean_imgs = []
for i in range (n_components):
    v  = principle_vec[i,:]
    img = v.reshape((h,w))
    mean_imgs.append(img)
mean_imgs = np.array(mean_imgs)
print(mean_imgs.shape)
```

```
(251, 50, 37)
```

In [25]:
```python
def plot_grid(images, titles, h, w, rows=3, cols=3):
    plt.figure(figsize=(2*cols, 2 *rows))
    for i in range(rows*cols):
        plt.subplot(rows,cols,i+1)
        plt.imshow(images[i].reshape(h,w), cmap='gray')
        plt.title(titles[i])
pca_tiles = [f"eigenvector-{i}" for i in range(n_components)]
plot_grid(mean_imgs, pca_tiles, h, w)
```