mina ilkhani

610398191

HW3-prat1

load data

In [1]:

```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people(min_faces_per_person=150, resize=0.4)
x=lfw_people.data
y=lfw_people.target
```

data shape:

In [2]:

```
lfw_people.images.shape
```

Out[2]:

```
(766, 50, 37)
```

In [3]:

```
x.shape
```

Out[3]:

```
(766, 1850)
```

In [4]:

```
y.shape
```

Out[4]:

```
(766,)
```

In [5]:

```
len(lfw_people)
```

Out[5]:

```
5
```

In [6]:

```
list(lfw_people.target_names)
```

Out[6]:

```
['Colin Powell', 'George W Bush']
```

split train and test (75-25)

In [7]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
```

In [8]:

```
X_test.shape
```

Out[8]:

```
(192, 1850)
```

In [9]:

```
X_train.shape
```

Out[9]:

```
(574, 1850)
```

192 record for test and 574 data for train. earch data have 1850 pixels

In [10]:

```
lfw_people.data[0].shape
```

Out[10]:

```
(1850,)
```

plot:

1850 = x_plot*y_plot

In [11]:

```
x_plot = 50
```

```
y_plot = 37
import matplotlib.pyplot as plt

for i in range(40):
    plt.subplot(4, 10, i + 1)
    plt.imshow(X_train[i].reshape((x_plot, y_plot)), )
    plt.xticks(())
    plt.yticks(())
```

normalization:

In [12]:

```
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()
X_train = min_max_scaler.fit_transform(X_train)
X_test = min_max_scaler.transform(X_test)
```

PCA:

In [13]:

```
import numpy as np
from sklearn.decomposition import PCA
def principal_component_analysis(n):
    pca = PCA(n_components=n)
    pca.fit(X_train)

    X_train_pca = pca.transform(X_train)
    X_test_pca = pca.transform(X_test)
    return X_train_pca, X_test_pca
```

In [14]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def model(x_train, x_test, y_test, y_train,clf):
    clf.fit(x_train, y_train)
    predY = clf.predict(x_test)
    print("on test:")
    print ("accuracy score:",accuracy_score(y_true=y_test, y_pred=predY)*100)
    print ("precision score:",precision_score(predY, y_test, average='weighted')*100)
    print("recall score:", recall_score(predY, y_test, average='weighted')*100)
    print("f1 score:", f1_score(y_true=y_test, y_pred=predY, average='weighted')*100)

    predY = clf.predict(x_train)
    print("\non train:")
    print ("accuracy score:",accuracy_score(y_true=y_train, y_pred=predY)*100)
    print ("precision score:",precision_score(predY, y_train, average='weighted')*100)
    print("recall score:", recall_score(predY, y_train, average='weighted')*100)
    print("f1 score:", f1_score(y_true=y_train, y_pred=predY, average='weighted')*100)
```

confusion matrix:

In [15]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

def plot_confusion_matrix(decision, x_train, Y_train, x_test, Y_test) :
    decision.fit(x_train, Y_train)
    predY = decision.predict(x_test)
    cf_matrix = confusion_matrix(Y_test, predY)
    sns.heatmap(cf_matrix, annot=True)
```

now we build a multilayer perceptron model:

In [16]:

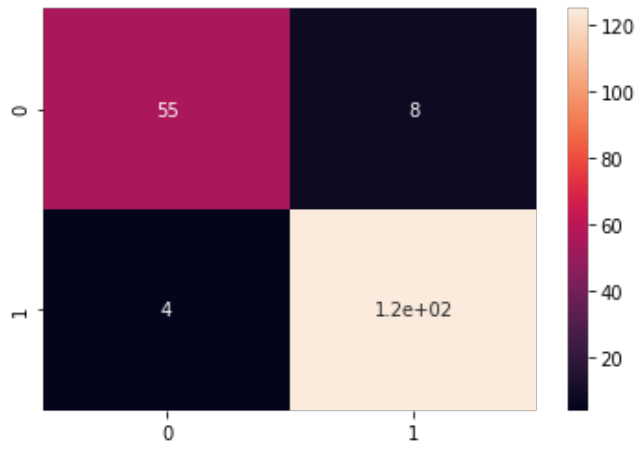
```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', alpha=0.00005, hidden_layer_sizes=(50, ), random_state=1)
model(X_train, X_test, y_test, y_train, clf)
```

```
on test:
accuracy score: 93.75
precision score: 93.94995078134613
recall score: 93.75
f1 score: 93.69525090727068

on train:
accuracy score: 100.0
precision score: 100.0
recall score: 100.0
f1 score: 100.0
```

In [17]:

```
plot_confusion_matrix(clf, X_train, y_train, X_test, y_test)
```



now PCA:

In [18]:

```
X_train_pca, X_test_pca = principal_component_analysis(570)
```

In [19]:

```
X_train_pca.shape
```

Out[19]:

```
(574, 570)
```

In [20]:

```
X_test_pca.shape
```

Out[20]:

```
(192, 570)
```

with the parameters that weee used before PCA:

In [21]:

```
model(X_train_pca, X_test_pca, y_test, y_train, clf )
```

```
on test:
accuracy score: 95.3125
precision score: 95.2996185554325
recall score: 95.3125
f1 score: 95.321835074604

on train:
accuracy score: 100.0
precision score: 100.0
recall score: 100.0
f1 score: 100.0
```

change parameters:

In [22]:

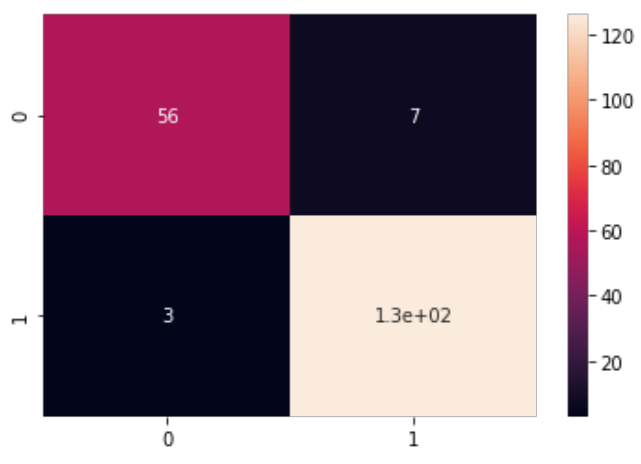
```
clf = MLPClassifier(solver='lbfgs', alpha=0.03, hidden_layer_sizes=(50, ), random_state=1)
model(X_train_pca, X_test_pca, y_test, y_train, clf )
```

```
on test:
accuracy score: 94.79166666666666
precision score: 94.97469853574503
recall score: 94.79166666666666
f1 score: 94.74604242272555

on train:
accuracy score: 100.0
precision score: 100.0
recall score: 100.0
f1 score: 100.0
```

In [23]:

```
plot_confusion_matrix(clf, X_train_pca, y_train, X_test_pca, y_test)
```



In [24]:

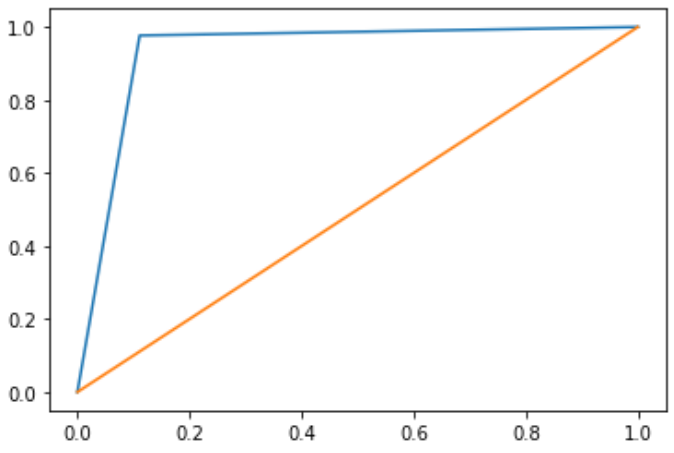
In [24]:

```
from sklearn.metrics import roc_auc_score, roc_curve
def receiver_operating_characteristic(x_test, Y_test):
    predY = clf.predict(x_test)
    print("ROC score:",roc_auc_score(Y_test, predY))
    fpr, tpr, thresholds = roc_curve(Y_test, predY)
    plt.plot(fpr, tpr)
    plt.plot([0,1], [0,1])
    plt.show()
```

In [25]:

```
receiver_operating_characteristic(X_test_pca, y_test)
```

```
ROC score: 0.9328165374677002
```



In [26]:

```
receiver_operating_characteristic(X_train_pca, y_train)
```

```
ROC score: 1.0
```