```
"""
==================================================
Faces recognition example using eigenfaces and SVMs
==================================================

The dataset used in this example is a preprocessed excerpt of the
"Labeled Faces in the Wild", aka LFW_:

  http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz (233MB)

.. _LFW: http://vis-www.cs.umass.edu/lfw/

Expected results for the top 5 most represented people in the dataset:

================== ============ ======= ========== =======
                   precision    recall  f1-score   support
================== ============ ======= ========== =======
     Ariel Sharon       0.67      0.92      0.77        13
     Colin Powell       0.75      0.78      0.76        60
  Donald Rumsfeld       0.78      0.67      0.72        27
    George W Bush       0.86      0.86      0.86       146
Gerhard Schroeder       0.76      0.76      0.76        25
      Hugo Chavez       0.67      0.67      0.67        15
       Tony Blair       0.81      0.69      0.75        36

      avg / total       0.80      0.80      0.80       322
================== ============ ======= ========== =======

"""
from __future__ import print_function

from time import time
import logging
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import fetch_lfw_people
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
from sklearn.svm import SVC


```

```
45 print(__doc__)
46
47 # Display progress logs on stdout
48 logging.basicConfig(level=logging.INFO, format='%(asctime)
   s %(message)s')
49
50
51 ###############################################################
   ####################
52 # Download the data, if not already on disk and load it as
    numpy arrays
53
54 lfw_people = fetch_lfw_people(min_faces_per_person=70,
   resize=0.4)
55
56 # introspect the images arrays to find the shapes (for
   plotting)
57 n_samples, h, w = lfw_people.images.shape
58
59 # for machine learning we use the 2 data directly (as
   relative pixel
60 # positions info is ignored by this model)
61 X = lfw_people.data
62 n_features = X.shape[1]
63
64 # the label to predict is the id of the person
65 y = lfw_people.target
66 target_names = lfw_people.target_names
67 n_classes = target_names.shape[0]
68
69 print("Total dataset size:")
70 print("n_samples: %d" % n_samples)
71 print("n_features: %d" % n_features)
72 print("n_classes: %d" % n_classes)
73
74
75 ###############################################################
   ####################
76 # Split into a training set and a test set using a
   stratified k fold
77
78 # split into a training and testing set
79 X_train, X_test, y_train, y_test = train_test_split(
80     X, y, test_size=0.25, random_state=42)
81
82
```

```
 83 ##############################################################
    ######################
 84 # Compute a PCA (eigenfaces) on the face dataset (treated
     as unlabeled
 85 # dataset): unsupervised feature extraction /
    dimensionality reduction
 86 n_components = 150
 87
 88 print("Extracting the top %d eigenfaces from %d faces"
 89       % (n_components, X_train.shape[0]))
 90 t0 = time()
 91 pca = PCA(n_components=n_components, svd_solver='
    randomized',
 92           whiten=True).fit(X_train)
 93 print("done in %0.3fs" % (time() - t0))
 94
 95 eigenfaces = pca.components_.reshape((n_components, h, w)
    )
 96
 97 print("Projecting the input data on the eigenfaces
    orthonormal basis")
 98 t0 = time()
 99 X_train_pca = pca.transform(X_train)
100 X_test_pca = pca.transform(X_test)
101 print("done in %0.3fs" % (time() - t0))
102
103
104 ##############################################################
    ######################
105 # Train a SVM classification model
106
107 print("Fitting the classifier to the training set")
108 t0 = time()
109 param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
110               'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.
    01, 0.1], }
111 clf = GridSearchCV(SVC(kernel='rbf', class_weight='
    balanced'), param_grid)
112 clf = clf.fit(X_train_pca, y_train)
113 print("done in %0.3fs" % (time() - t0))
114 print("Best estimator found by grid search:")
115 print(clf.best_estimator_)
116
117
118 ##############################################################
    ######################
```

```
119 # Quantitative evaluation of the model quality on the
    test set
120
121 print("Predicting people's names on the test set")
122 t0 = time()
123 y_pred = clf.predict(X_test_pca)
124 print("done in %0.3fs" % (time() - t0))
125
126 print(classification_report(y_test, y_pred, target_names=
    target_names))
127 print(confusion_matrix(y_test, y_pred, labels=range(
    n_classes)))
128
129
130 ###############################################################
    ######################
131 # Qualitative evaluation of the predictions using
    matplotlib
132
133 def plot_gallery(images, titles, h, w, n_row=3, n_col=4):
134     """Helper function to plot a gallery of portraits"""
135     plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
136     plt.subplots_adjust(bottom=0, left=.01, right=.99,
    top=.90, hspace=.35)
137     for i in range(n_row * n_col):
138         plt.subplot(n_row, n_col, i + 1)
139         plt.imshow(images[i].reshape((h, w)), cmap=plt.cm
    .gray)
140         plt.title(titles[i], size=12)
141         plt.xticks(())
142         plt.yticks(())
143
144
145 # plot the result of the prediction on a portion of the
    test set
146
147 def title(y_pred, y_test, target_names, i):
148     pred_name = target_names[y_pred[i]].rsplit(' ', 1)[-1
    ]
149     true_name = target_names[y_test[i]].rsplit(' ', 1)[-1
    ]
150     return 'predicted: %s\ntrue:      %s' % (pred_name,
    true_name)
151
152 prediction_titles = [title(y_pred, y_test, target_names,
    i)
```

```
153                     for i in range(y_pred.shape[0])]
154
155 plot_gallery(X_test, prediction_titles, h, w)
156
157 # plot the gallery of the most significative eigenfaces
158
159 eigenface_titles = ["eigenface %d" % i for i in range(
    eigenfaces.shape[0])]
160 plot_gallery(eigenfaces, eigenface_titles, h, w)
161
162 plt.show()
163
```