FacialRecognition

December 8, 2020

1 Facial Recognition with Machine Learning

2 Data Science Academy

3 Vinicius Biazon

3.0.1 Facial Recognition with Machine Learning Using SVM and PCA

We are going to create a model for facial recognition, using SVM and PCA.

The dataset used in this project is the Labeled Faces in the Wild Home, a set of face images prepared for Computer Vision tasks. It is available both on Keras and at http://vis-www.cs.umass.edu/lfw/.

3.0.2 Loading Packages

[1]:
```
# Image storage
import numpy as np

# Machine Learning
from sklearn.model_selection import train_test_split
from sklearn import decomposition
from sklearn import svm

# Image Dataset
from sklearn import datasets

# Graph creation
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# Loading dataset (at least 70 images per person with a 0.4 scaling factor)
dataset_faces = datasets.fetch_lfw_people(min_faces_per_person = 70, resize = 0.
 ↪4)
```

[3]:
```
# Checking the dataset shape
dataset_faces.data.shape
```

[3]: (1288, 1850)

3.0.3 Preparing the Dataset

[4]:
```
# Extracting the shape details from the images
num_samples, height, width = dataset_faces.images.shape
```

[5]:
```
# Putting data in X (input variables) and target in y (output variable)
X = dataset_faces.data
```

[6]:
```
# Number of X attributes
num_attributes = X.shape[1]
```

[7]:
```
print(X)
```

```
[[254.        254.        251.66667  ...  87.333336  88.666664  86.666664]
 [ 39.666668  50.333332  47.        ... 117.666664 115.        133.66667 ]
 [ 89.333336 104.        126.        ... 175.33333  183.33333  183.      ]
 ...
 [ 86.        80.333336  74.666664 ...  44.         49.666668  44.666668]
 [ 50.333332  65.666664  88.        ... 197.       179.33333  166.33333 ]
 [ 30.        27.        32.666668 ...  35.         35.333332  61.      ]]
```

Each pixel can have a value from 0 to 255, for black and white images.

[8]:
```
# Putting the target in y
y = dataset_faces.target
```

[9]:
```
# Extracting class names
target_names = dataset_faces.target_names
```

[10]:
```
# Number of classes
num_class = target_names.shape[0]
```

[11]:
```
# Printing a summary of the data
print ("\nTotal Dataset Size: \n")
print ("Number of samples (images):% d"% num_samples)
print ("Height (pixels):% d"% height)
print ("Width (pixels):% d"% width)
print ("Number of Attributes (variables):% d"% num_attributes)
print ("Number of Classes (people):% d"% num_class)
```

```

Total Dataset Size:

Number of samples (images): 1288
Height (pixels): 50
Width (pixels): 37
```

```
Number of Attributes (variables): 1850
Number of Classes (people): 7
```

3.0.4 Viewing the Data

[12]:
```python
# Images Plot

# Setting the plot area size
fig = plt.figure(figsize = (12, 8))

# 15 images Plot
for i in range(15):

    # Dividing images into 5 columns and 3 rows
    ax = fig.add_subplot(3, 5, i + 1, xticks = [], yticks = [])

    # Showing the images
    ax.imshow(dataset_faces.images[i], cmap = plt.cm.bone)
```

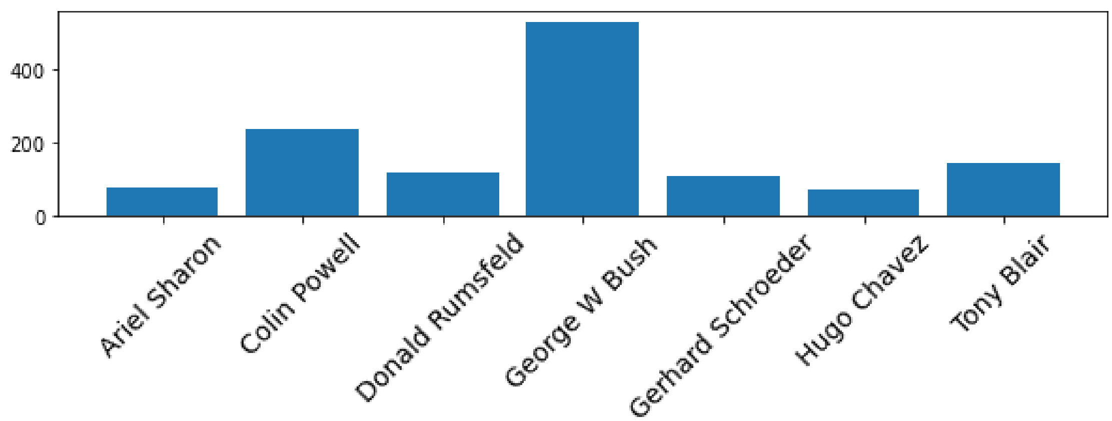
3.0.5 Viewing the distribution of people from the Dataset

[13]:
```
# Setting the plot area size
plt.figure(figsize = (10, 2))

# Capturing unique target (class) values
unique_targets = np.unique(dataset_faces.target)

# Counting total of each class
counts = [(dataset_faces.target == i).sum() for i in unique_targets]

# Result plot
plt.xticks(unique_targets, dataset_faces.target_names[unique_targets])
locs, labels = plt.xticks()
plt.setp(labels, rotation = 45, size = 14)
_ = plt.bar(unique_targets, counts)
```



3.0.6 Splitting the data in training and testing

[14]:
```
# Splitting data into training and testing
X_train, X_test, y_train, y_test = train_test_split(dataset_faces.data,␣
↪dataset_faces.target, random_state = 0)
```

[15]:
```
# Print
print(X_train.shape, X_test.shape)
```

```
(966, 1850) (322, 1850)
```

- For training we have 966 images and 1850 attributes, or images pixels.
- For testing we have 322 images and 1850 attributes, or images pixels.

3.1 Pre-Processing: Principal Component Analysis (PCA)

We are going to use the PCA to reduce these 1850 resources to a manageable level, while keeping most of the information in the data set. We will create a PCA model with 150 components

[16]:
```
# Creating the PCA model
pca = decomposition.PCA(n_components = 150,
                        whiten = True,
                        random_state = 1999,
                        svd_solver = 'randomized')
```

[17]:
```
# Training the model
pca.fit(X_train)
```

[17]:
```
PCA(n_components=150, random_state=1999, svd_solver='randomized', whiten=True)
```

[18]:
```
# Applying the PCA model to train and test data
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

[19]:
```
# Shape
print(X_train_pca.shape)
print(X_test_pca.shape)
```

```
(966, 150)
(322, 150)
```

3.1.1 Creating the Machine Learning Model with SVM

[20]:
```
# Creating the model
svm_model = svm.SVC(C = 10., gamma = 0.001)
```

[21]:
```
# Training the model
svm_model.fit(X_train_pca, y_train)
```

[21]:
```
SVC(C=10.0, gamma=0.001)
```

3.1.2 Evaluating the Model

[22]:
```
# Shape of test data
print(X_test.shape)
```

```
(322, 1850)
```

[23]:
```
# Plot area size
fig = plt.figure(figsize = (12, 8))

# 15 imagens loop
for i in range(15):
```

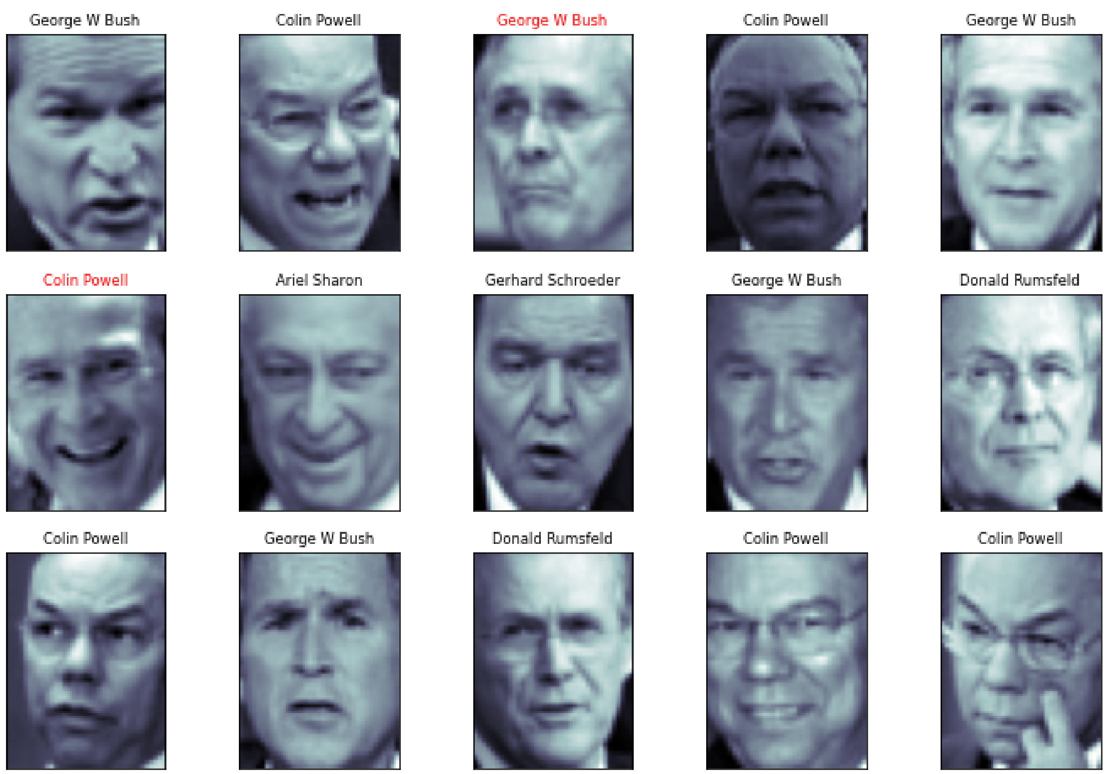
```
    # Subplots
    ax = fig.add_subplot(3, 5, i + 1, xticks = [], yticks = [])

    # Showing the real image in the test dataset
    ax.imshow(X_test[i].reshape((50, 37)), cmap = plt.cm.bone)

    # Making the class prediction with the trained model
    y_pred = svm_model.predict(X_test_pca[i].reshape(1,-1))[0]

    # Putting colors in the results
    color = 'black' if y_pred == y_test[i] else 'red'

    # Defining the title
    ax.set_title(dataset_faces.target_names[y_pred], fontsize = 'small', color = color)
```



Red names represent model errors. Black names mean the model is right.

3.1.3 Model Score

[24]:
```
print(svm_model.score(X_test_pca, y_test))
```

```
0.8416149068322981
```

This model has an efficiency around 84%, which means that for every 100 images the prediction is correct in 84 cases.