Face Recognition

December 19, 2020

Face Recognition System using Random Forest Qusai Issa 21710198

in this work I used the Random forest method to recognise faces from sk-learn database then to compare it with the SVM face recognition system. Also i implemented my work into a streamlit app.

1.1 Loading Data

First, lets load the dataset from within Pyhon library (scikit-learn) using the fetch method.

[1]:
```
# import libraries
import warnings
warnings.filterwarnings('ignore')# ignore warnings
from sklearn.datasets import fetch_lfw_people # this is the dataset
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
#faces = fetch_lfw_people(min_faces_per_person=60, color=False,  resize=0.5,
#                         slice_=(slice(70, 196, None), slice(75, 171, None)))
faces = fetch_lfw_people(min_faces_per_person=60)
print(faces.target_names)
print(faces.images.shape)
```

```
['Ariel Sharon' 'Donald Rumsfeld' 'George W Bush' 'Gerhard Schroeder'
 'Junichiro Koizumi' 'Tony Blair']
(1041, 62, 47)
```

1.2 Quick Exploratory Data and preparing

this will include finding the size of the dataset, showing few records, and also show the columns names in the data.

1.2.1 Display Faces

We can view some of the images in the dataset.each one with his name

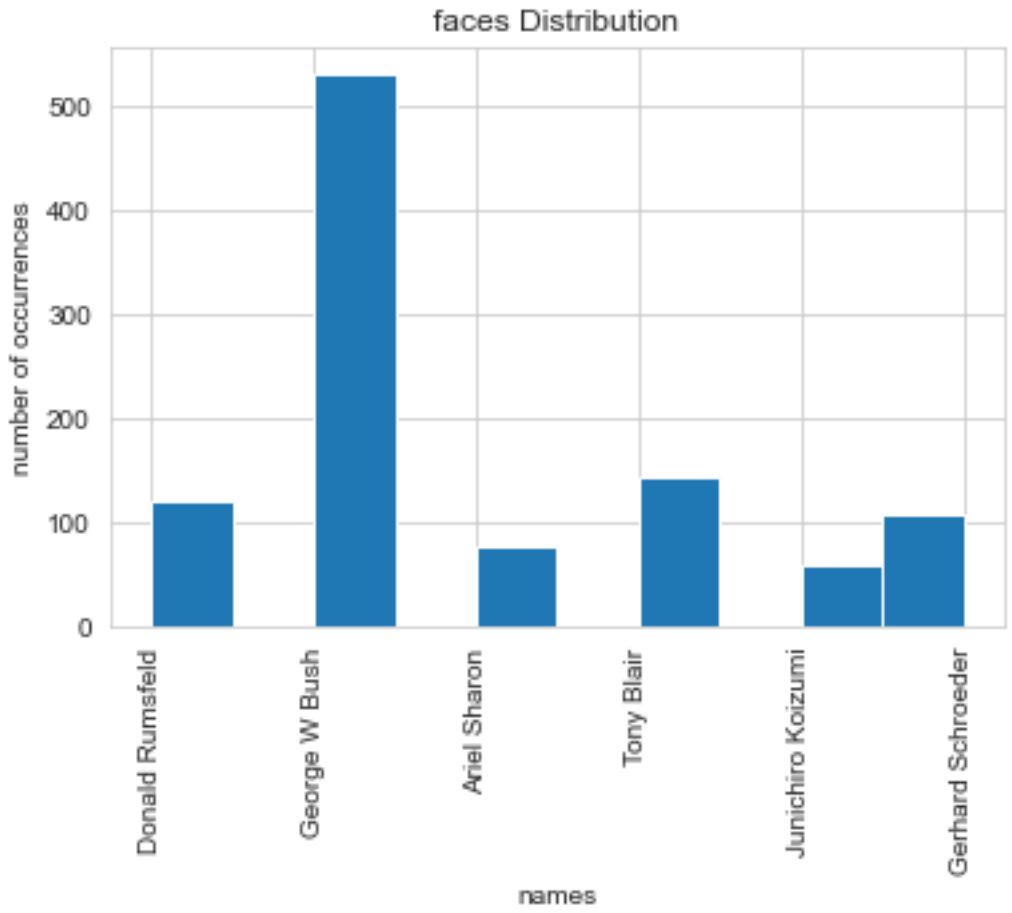
[2]:
```
#
# create grid of 5 x 2 to sohw images
fig, ax = plt.subplots(3,8,figsize=(12,6))
for i, axi in enumerate(ax.flat):
```

```
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],xlabel=faces.target_names[faces.target[i]])
```

1.2.2 Class Distribution

As always, one of the first things we should always check is the class distribution in the dataset. How many symbol of each type is represented in this dataset.

[3]:
```
sns.set_style("whitegrid")
plt.figure()
plt.hist(faces.target_names[faces.target])
plt.title('faces Distribution')
plt.xlabel('names')
plt.ylabel('number of occurrences')
plt.xticks(rotation=90)
plt.show()
faces.target_names
#faces.target_names[faces.target[i]])
```



[3]:
```
array(['Ariel Sharon', 'Donald Rumsfeld', 'George W Bush',
       'Gerhard Schroeder', 'Junichiro Koizumi', 'Tony Blair'],
      dtype='<U17')
```

after running the code withowut imbalance I saw that 50% of the false predectios was on georg bosh, becaus we have more than 500 image for him.

1.2.3 Preparing the Dataset

Data Imbalance

We Will remove some data.

scens we have so much data of George W Bush, we will remove some of it to improve the quality of the training.

[4]:
```
#George W Bush label is 2 but les's get sure of that
faces.target_names[2]
```

[4]:
```
'George W Bush'
```

[5]:
```
# this is a list includs all of George W Bush locations
George =[]
for i in range(1041):
    if(faces.target[i]==2):
        George.append(i)
#print the first 15 location
George[:15]
```

[5]:
```
[1, 4, 6, 7, 8, 9, 10, 11, 14, 18, 19, 22, 23, 24, 25]
```

now we have all the locations but we don't want to remove all of it, we just need to get the data balanced.

so we will randomly delete 400 pictures of him.

[6]:
```
import random
if(len(George)>400):
    while(len(George)>400):
        George.pop(random.randint(0, len(George)))

len(George)
```

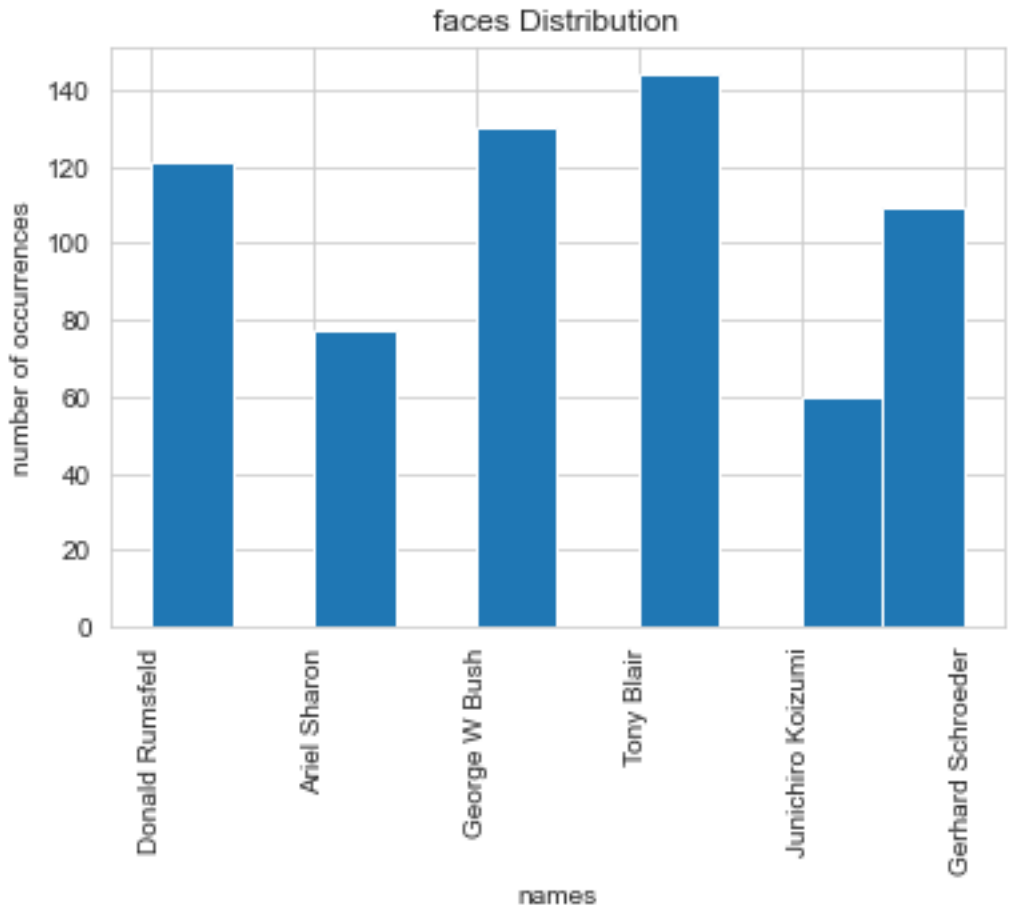
[6]:
```
400
```

Now we have 400 location of George that we will remove at the next cell

[7]:
```
for i in range(len(George)):
    faces.data = np.delete(faces.data ,George[400-i-1],0)
    faces.target = np.delete(faces.target ,George[400-i-1],0)
    faces.images = np.delete(faces.images ,George[400-i-1],0)
```

lets plot the data one more time to make sure that every thing went well.

[8]:
```
sns.set_style("whitegrid")
plt.figure()
plt.hist(faces.target_names[faces.target])
plt.title('faces Distribution')
plt.xlabel('names')
plt.ylabel('number of occurrences')
plt.xticks(rotation=90)
plt.show()
faces.target_names
#faces.target_names[faces.target[i]])
print(len(faces.data))
print(len(faces.target))
```




```
641
641
```

1.3 Prepare Data

• Lets first prepare the data as X, representing all the input features (pixel values columns), and y to represent the names label in the dataset.

[9]:
```
X = faces.data
y = faces.target
```

[10]:
```
print ('X reprsnt the data, and its type is:      '+str(type(X)) )
print ('Y reprsnt the labels, and its type is:    '+str(type(y)) )
```

```
X reprsnt the data, and its type is:      <class 'numpy.ndarray'>
Y reprsnt the labels, and its type is:    <class 'numpy.ndarray'>
```

spliting the data

Here we split the data 90% for training, 10% for testing using sklearn and then we will check the shape of the data

[11]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier

print(f'The size of the original data is {X.shape[0]}')
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size = 0.
 ↪1,random_state=30)
# check the shape of the data
print(f'The size of the Training set is {X_train.shape[0]}')
print(f'The size of the Training set is {X_test.shape[0]}')
# notice the number of column in the dataset
print(f'The number of columns in the training and testing set is {X_train.
 ↪shape[1]}')
```

```
The size of the original data is 641
The size of the Training set is 576
The size of the Training set is 65
The number of columns in the training and testing set is 2914
```

Now let's store the test data into a csv file to use it in the streamlit app later.

[12]:
```
df = pd.DataFrame(X_test)
df['Label']=y_test
df.to_csv('Xy_test.csv', index=False)
```

1.4 Build Random Forest

here we call the RF class with trees number of 200

[17]:
```
from sklearn.ensemble import RandomForestClassifier
#trees number 100
rf=RandomForestClassifier(n_estimators=200,)
# fit the model on training data
rf.fit(X_train,y_train)
```

[17]:
```
RandomForestClassifier(n_estimators=200)
```

1.4.1 Make Predictions

[18]:
```
# Predicted classes of the testing set
rf_predictions = rf.predict(X_test)


# get probabilities of predictions
#rf_probs = rf.predict_proba(X_test)
```

```
# show the class probs for the first image (note the output)
#rf_probs[:1]
```

1.5.1 Results

Lets start by computing the accuracy of the RF model using the accuracy_score of Scikit-Learn.

[19]:
```
import numpy as np

correct = 0
for i in range(len(rf_predictions)):
    if(rf_predictions[i] == y_test[i]):
        correct = correct +1


print ('The overall accuracy of RF is',np.round(correct/len(rf_predictions)*100))
```

```
The overall accuracy of RF is 80.0
```

1.5.2 Manual Error Analysis

[16]:
```
 # store actual class labels and predicted ones in a dataframe
results = pd.DataFrame({'Actual':faces.target_names[y_test],'Predicted':faces.
 ↪target_names[rf_predictions]})
results
```

[16]:

	Actual	Predicted
0	George W Bush	George W Bush
1	Junichiro Koizumi	Gerhard Schroeder
2	George W Bush	George W Bush
3	Tony Blair	Tony Blair
4	Gerhard Schroeder	Gerhard Schroeder
..	...	...
60	Gerhard Schroeder	Gerhard Schroeder
61	George W Bush	George W Bush
62	Tony Blair	Tony Blair
63	Junichiro Koizumi	Junichiro Koizumi
64	Tony Blair	Tony Blair

[65 rows x 2 columns]

[17]:
```
incorrect = results[results.Actual!=results.Predicted]
incorrect.head()
#
```

[17]:

	Actual	Predicted
1	Junichiro Koizumi	Gerhard Schroeder
5	Junichiro Koizumi	Gerhard Schroeder

```
8       George W Bush    Donald Rumsfeld
9        Ariel Sharon      George W Bush
16      George W Bush  Gerhard Schroeder
```

Show some Predictions as images

[19]:
```
fig, ax = plt.subplots(4, 4,figsize=(10,10))
for i, axi in enumerate(ax.flat):
    axi.imshow(X_test[i].reshape(62, 47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[rf_predictions[i]].
 ↪split()[-1],color='black' if rf_predictions[i] == y_test[i] else 'red')
fig.suptitle('Predicted Names; Incorrect Labels in Red', size=14);
```

1.6 Save Model

in order to use the streamlit app

[20]:
```
import pickle
# Save to file in the current working directory
pkl_filename = "my_model.pkl"
# save your model that was created above (lg_model)
with open(pkl_filename, 'wb') as file:
```

```
    pickle.dump(rf, file)
```

2 Face Recognetion using SVM

Down there we use SVM for face recognation as we did at lab 4 but on the same dataset that we use at random forest. printing the accuracy at last.

[20]:
```
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
# create dimenssion reduction
pca = PCA(n_components=150, whiten=True, random_state=42)
# create your SVM model with RBF kernel
svc = SVC(kernel='rbf', class_weight='balanced')
# your pipline
model = make_pipeline(pca, svc)
from sklearn.model_selection import train_test_split
# random_state is for results reproduction
X_train, X_test, y_train, y_test = train_test_split(faces.data, faces.
 ↪target,random_state=42)
from sklearn.model_selection import GridSearchCV
param_grid = {'svc__C': [1, 5, 10, 50],'svc__gamma': [0.0001, 0.0005, 0.001, 0.
 ↪005]}
grid = GridSearchCV(model, param_grid)
warnings.filterwarnings('ignore')# ignore warnings
grid.fit(X_train, y_train)
best_params = grid.best_params_
# best model we have
model = grid.best_estimator_
y_predicted = model.predict(X_test)
import pickle
# Save to file in the current working directory
pkl_filename = "svm_model.pkl"
# save your model that was created above (lg_model)
with open(pkl_filename, 'wb') as file:
    pickle.dump(model, file)
# Load from file
with open(pkl_filename, 'rb') as file:
    pickle_model = pickle.load(file)
# Lets test the mode loaded from a file and check results
score = pickle_model.score(X_test, y_test)
print("Test score: {0:.2f} %".format(100 * score))
y_hat = pickle_model.predict(X_test)
```

```
Test score: 85.71 %
```

it is obvious that SVM is way better than Random forest

SVM gives you “support vectors”, that is points in each class closest to the boundary between

classes.

3. Face Recognition with Local Binary Patterns critically review the paper that we are going to review is made by Timo Ahonen, Abdenour Hadid, and Matti Pietik¨ainen

this paper is about Face Recognition using "Local Binary Patterns" which is a type of visual descriptor used for classification in computer vision. In this paper they used a new technique to face recognition which divided the face into small regions from which Local Binary Pattern (LBP) histograms are extracted and concatenated into a single, spatially enhanced feature histogram efficiently representing the face image. To classify the faces: nearest neighbor classifier. They also used the Chi square as a dissimilarity measure, which at their time was better than PCA and A lot of other methods. Chi square has an advantageous property which it gave the same result what ever the light was, because it depend on the differences between cells, so if the light is up, that will change all the pixels which won't affect the difference. The data set they used is FERET database with slight modifications. At first they Determines the face coordination using eyes coordination then they cut the image with an elliptical shape to exclude non-face area from the image to get better recognition. To assess the performance of the three proposed distance measures. they use two different LBP operators in windows of varying size. We calculated the distance matrices for each of the different settings and used the permutation tool to calculate the probabilities of the measures outperforming each other. To improve the system, they try to find the importance of each region. This idea was because the psychophysical findings which indicate that some facial features (such as eyes) play more important roles in face recognition than other features (such as the nose). so they assigned weights from 0 to 4 to the regions 4 to the important ones, and 0 to neglected areas. After all the accuracy that came out with this work was Fantastic! It beats all other methods at that time the method achieved a recognition rate of 97All other methods did not exceed 90to prove the performance of LBP on datasets, they also used the method on the ORL face database. Experiments not only confirmed the correctness of the approach but the document's durability and tolerance of alignment changes.

Press Me to go to the Paper.

Thank You!

Eyad Elyan elyan.eyad@gmail.com.

Qusai ISSA qusai.2010.123@gmail.com.