Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

Followed by the explanation of another data set names as Face Recognition on Olivetti Dataset. The data set contains 10 face images for each subject. The data set has 40 different person-owned, facial images. Size of each image is 64x64. There are 40 distinct subjects, and each has 10 images taken differently. (Hence the data set has 400 rows and pixel size of each image is 64 X 64). Names of 40 people were encoded to an integer from 0 to 39, the images were taken at different times, by varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses).

Project Idea:

Face recognition is a hot topic in industry.

This article is about performing multioutput regression on the Olivetti images and predicting the faces. Below are the algorithums to be used.

1. Logistics Regression
2. RandomForestRegressor
3. KNN
4. SVM
5. Naive Bayes

3. Requirements

The python libraries are necessary for the rest of this mini project

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from skimage.io import imshow
import matplotlib.image as mpimg
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
```

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

4. Description of the Python program

Step 1: Load and explore the data

```
# Input data files consisting of the images
pics = np.load("olivetti_faces.npy")
labels = np.load("olivetti_faces_target.npy")
```

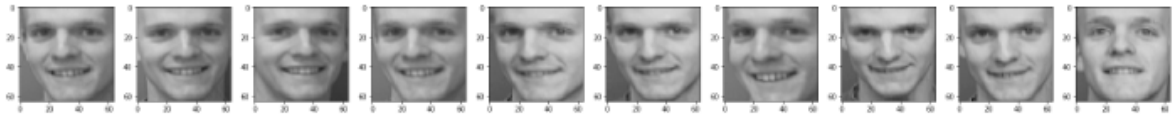
```
print("pics: ", pics.shape)
print("labels: ", labels.shape)

```

```
pics:  (400, 64, 64)
labels:  (400,)
```

```
# Sample images of a subject
img_cnt = 10
plt.figure(figsize=(24,24))
for i in range(img_cnt):
    plt.subplot(1,10,i+1)
    x=pics[i+40] # 4th subject
    imshow(x)
plt.show()


```



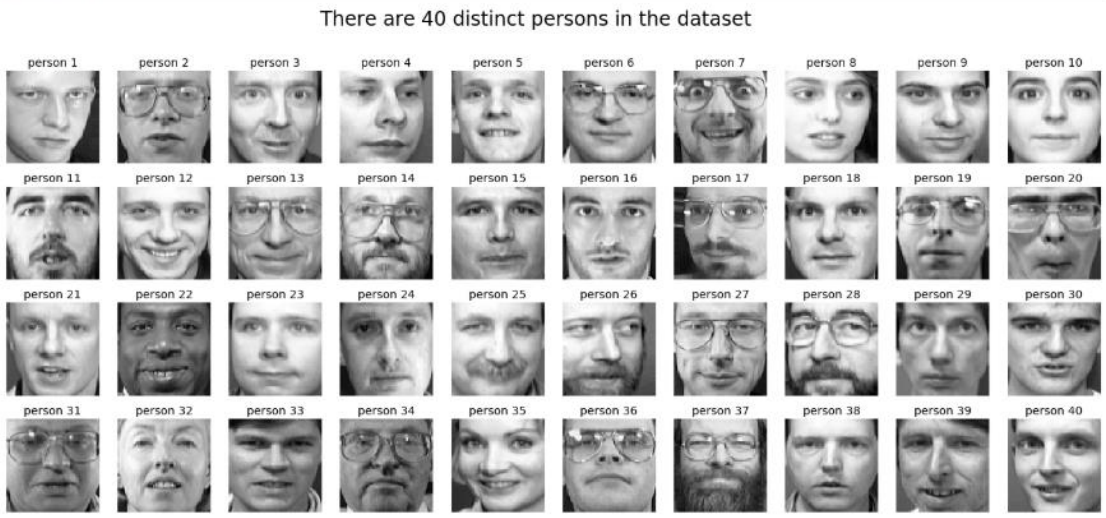
Step 2: Visualize the data

To see the dataset and model idea for the dataset.

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

```
# All unique faces in the sample
fig = plt.figure(figsize=(24, 10))
columns = 10
rows = 4
for i in range(1, columns*rows +1):
    img = pics[(10*i-1),:,:]
    fig.add_subplot(rows, columns, i)
    plt.imshow(img, cmap = plt.get_cmap('gray'))
    plt.title("person {}".format(i), fontsize=14)
    plt.axis('off')

plt.suptitle("There are 40 distinct persons in the dataset", fontsize=24)
plt.show()
```



Step 3: Reshape data

In [6]:
```
#Machine learning models can work on vectors. Since the image data is in the matrix form, it must be converted to a vector.

Y = labels.reshape(-1,1) # store labels in Y
X=pics.reshape(pics.shape[0], pics.shape[1]*pics.shape[2]) # reshape and store images in X

print("X shape:",X.shape)
print("Y shape:",Y.shape)
```

```
X shape: (400, 4096)
Y shape: (400, 1)
```

The data set contains 10 face images for each subject. Of the face images, 80 percent will be used for training, 20 percent for testing. Uses stratify feature to have equal number of training and test images for each subject. Thus, there will be 8 training images and 2 test images for each subject. You can play with training and test rates.

In [7]:
```
#Split data for train and test purposes

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state=46)

print("x_train: ",x_train.shape)
print("x_test: ",x_test.shape)
print("y_train: ",y_train.shape)
print("y_test: ",y_test.shape)
```

```
x_train:  (280, 4096)
x_test:  (120, 4096)
y_train:  (280, 1)
y_test:  (120, 1)
```

Step 4: Use Scikit Learn to execute Naïve Bayes, SVM, KNN, Decision Tree, Random forest

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

1. Use train_test_split from sklearn.cross_validation to shuffle and split the features and prices data into training and testing sets. Split the data into 90%-80%-80% training and 10% -20%-30% testing.
2. Assign the train and testing splits to X_train, X_test, y_train, and y_test.
3. Run the model Naïve Bayes, SVM, KNN, Decision Tree, Random forest
4. Calculate Accuracy and confusion matrix

```
rf = RandomForestClassifier(n_estimators = 400, random_state = 1)
rf.fit(x_train, y_train)
RF_accuracy = round(rf.score(x_test, y_test)*100,2)

y_pred = rf.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print("confusion matrix:")
print(cm)

print("RF_accuracy is %", RF_accuracy)

list_names.append("Random Forest")
list_accuracy.append(RF_accuracy)
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  after removing the cwd from sys.path.
```

```
confusion matrix:
[[3 0 0 ... 0 0 0]
 [0 6 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 2 0 0]
 [0 0 0 ... 0 3 0]
 [1 0 0 ... 0 0 2]]
RF_accuracy is % 93.33
```

Step 5: PCA

Introduce dimensionality reduction using PCA.

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

Our dataset is 64x64 image.Each pixel is considered a feature/dimension which ends up with 496

```
pca = PCA(.95)
X_train_pca = pca.fit_transform(x_train)
X_test_pca = pca.transform(x_test)


print('Original dataset:',x_train.shape)
print('Dataset after applying PCA:',X_train_pca.shape)
print('No of PCs/Eigen Faces:',len(pca.components_))
print('Eigen Face Dimension:',pca.components_.shape)
print('Variance Captured:',np.sum(pca.explained_variance_ratio_))
```

```
Original dataset: (280, 4096)
Dataset after applying PCA: (280, 103)
No of PCs/Eigen Faces: 103
Eigen Face Dimension: (103, 4096)
Variance Captured: 0.95040184
```

dimensions. With variance of 95% we could reduce to 103 principle components or dimensions.

Step 6:

Calculate Eigen Faces and Mean Face

When PCA is called a mean of the dataset is created. The mean is subtracted from the original dataset to create a covariance matrix. From the covariance matrix there is decomposition of Eigen face and Eigen values. This is an interesting dataset which can showcase the mean and Eigen face/Eigen Vector.

Mean Face is the average of all images in the dataset.

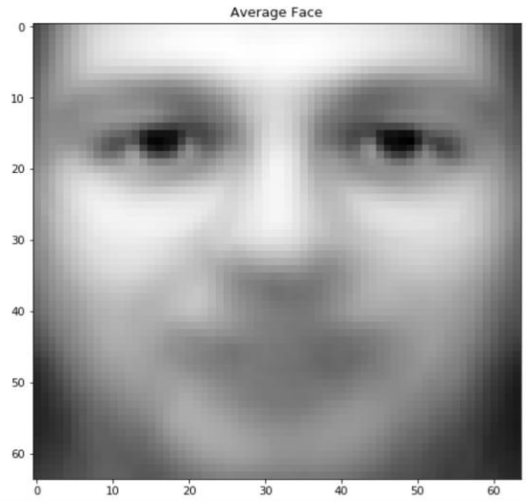
Eigen faces / principle components captures variations in the images. Variations can be different lighting, facial angle, hair length etc. There is one Eigen face for each principle component.

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

```
# Average face of the samples

plt.subplots(1,1,figsize=(8,8))
plt.imshow(pca.mean_.reshape((64,64)), cmap="gray")
plt.title('Average Face')
```

```
Text(0.5, 1.0, 'Average Face')
```



Step 7: Program output

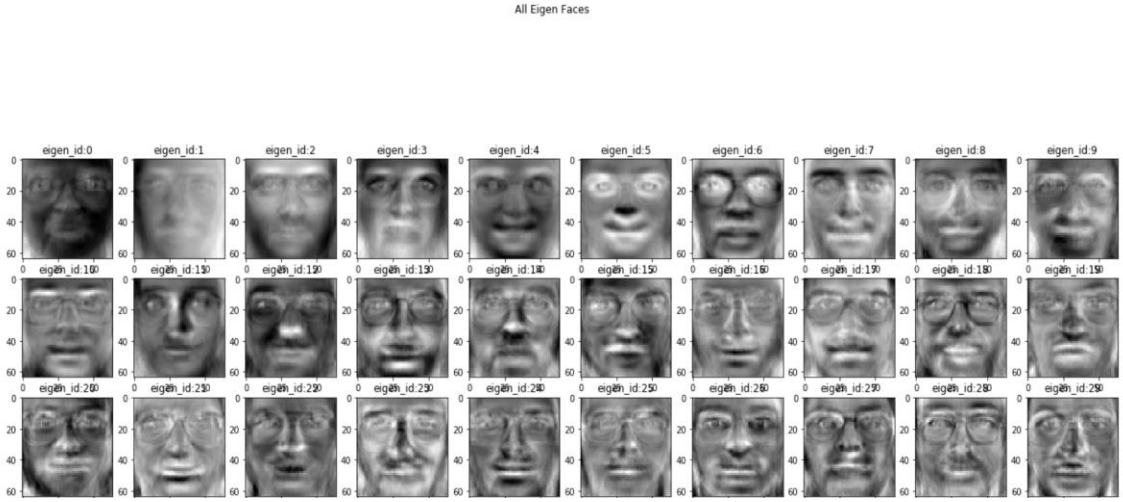
Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

```
number_of_eigenfaces=len(pca.components_)
eigen_faces=pca.components_.reshape((number_of_eigenfaces, pics.shape[1], pics.shape[2]))

columns=10
rows=int(number_of_eigenfaces/columns)
fig, axarr=plt.subplots(nrows=rows, ncols=columns, figsize=(24,24))
axarr=axarr.flatten()
for i in range(number_of_eigenfaces):
    axarr[i].imshow(eigen_faces[i],cmap="gray")

    axarr[i].set_title("eigen_id:{}".format(i))
plt.suptitle("All Eigen Faces".format(10*"=", 10*"="))
```

```
Text(0.5, 0.98, 'All Eigen Faces')
```



Accuracy Metrics Comparison with PCA

METHOD	90:10		80:20		70:30	
	Without PCA	With PCA	Without PCA	With PCA	Without PCA	With PCA
Naive Bayes	92.5	92.5	87.50	90.0	73.33	77.50
KNN	90.0	90.0	91.25	92.5	90.00	90.83
Random Forest	90.0	100.0	93.75	95.0	93.33	92.50
Logistic Regression	95.0	97.5	96.25	97.5	97.50	98.33
SVM	95.0	95.0	97.5	97.5	96.67	96.67

With PCA the SVM model have the same results, but other model's accuracy increases with use of PCA. There is accuracy of 100 for random forest. This could be because our data is small, and data is clean. With 90% data in the training it could have caused the algorithm to overfit.

Oanh Doan
Tess Mundadan
Ramandeep Kaur Bagri

Conclusion: PCA worked well with image dataset. We got a accuracy of 100 for Random forest with data split of 70:30 and PCA =103.This could be a result of overfitting with 90% data in training. Enhancement for this project would be to identify female - male learning/ Smile or not smile face.

References:

https://www.kaggle.com/serkanpeldek/face-recognition-on-olivetti-dataset
https://www.youtube.com/watch?v=PitcORQSjNM
https://www.youtube.com/watch?v=_VTtrSDHPwU&t=86s
https://www.kaggle.com/elikplim/eergy-efficiency-dataset/kernels

Class notes: Introduction to data mining and machine learning