Read Me

Question 1 : Face recognition using PCA
https://www.kaggle.com/muditr97/icm2015502-assignment-6-pca?scriptVersionId=12520719

Question 2 : Face recognition using LDA
https://www.kaggle.com/muditr97/assignment-6-question-2-lda?scriptVersionId=12540307

Data Set:

1. face-pca
2. Pca-faces
3. Facesss
4. face-face

The Assignment is on Jupyter Notebook
FileName for Face recognition using PCA**:**
ICM2015502_Assignment_6_PCA
FileName for Face recognition using LDA
ICM2015502_Assignment_6_LDA

Open the file which (is public) add this to Notebook, Data set Name :”face-face” “face-pca” “faces “ and “facesss” and run the code, for question 1, question 2. This will predict the accuracy of our model for face recognition using PCA, LDA

1. NumPy
2. SciPy
3. Pandas
4. Os
5. Sklearn
6. Matplotlib
7. PIL
8. Sns
9. Random
10. Csv

11. Glob
12. Sklearn.metrics ---> confusion_matrix
13. Sklearn.metrics --->accuracy_score

Analysis for Assignment for face recognition using PCA and LDA

Train_Test Split = 70 % and 30 % randomly using sklearn train test split tts()

```
def predict(X, eigen_faces, projected_faces, Y_train):
    mean_normalized = X - np.mean(X_train)

    projected_face = np.dot(mean_normalized, eigen_faces)
    all_diff = []
    for face in projected_faces:
        diff = np.linalg.norm(face-projected_face)
        all_diff.append(diff)
    var = np.argmin(all_diff)
    predicted = Y_train[var]
    return predicted
```

Predict function for the training set

```
def PCA(X_train, k):
    mean_normalized = X_train - np.mean(X_train)
    covariance_t = np.cov(mean_normalized) # Matrix Covarience
    eigen_values, eigen_vectors = np.linalg.eig(covariance_t) #
    # select best k eigen vectors
    eigen_vectors_pd = pd.DataFrame(data=eigen_vectors)
    sorted_indices = np.argsort(eigen_values)
    sorted_indices_k = sorted_indices[:k]
    eigen_vectors_pd_k = eigen_vectors_pd[sorted_indices_k] # k
    eigen_vectors_k = eigen_vectors_pd_k.values # now considere
    eigen_faces = np.dot(mean_normalized.T,eigen_vectors_k)
    projected_faces = np.dot(mean_normalized, eigen_faces)
    return eigen_faces,projected_faces
```

The PCA function

```python
def LDA(X_train,Y_train,m):
    #seperate by class
    loop_var = set(Y_train) #creation of set
    seperated_X = []
    for i in loop_var:
        single_class_data = []
        for index, j in enumerate(Y_train): # enumerates list everything from 1 to N
            if(i == j):
                single_class_data.append(X_train[index])
        seperated_X.append(np.array(single_class_data))
    tot_cov = np.cov(seperated_X[0].T)
    # calc total covariance
    for index,i in enumerate(seperated_X):
        if(index == 0):
            continue
        tot_cov += np.cov(i.T)
    # calc diff mean
    overall_mean = X_train.mean(0)
    diff_mean = np.atleast_1d(np.square(overall_mean - seperated_X[0].mean(0)))
    for index,i in enumerate(seperated_X):
        if(index == 0):
            continue
        diff_mean += np.square(overall_mean - seperated_X[index].mean(0))
    mean_matrix = np.expand_dims(diff_mean,axis = 1)
    mean_matrix = np.dot(mean_matrix,mean_matrix.T)
    criterion = np.linalg.inv(tot_cov).dot(mean_matrix)
    # select best m component
    fisher_faces,eigen_vectors_m = solve_LDA(X_train,m,criterion)
    return fisher_faces,eigen_vectors_m
```

LDA function