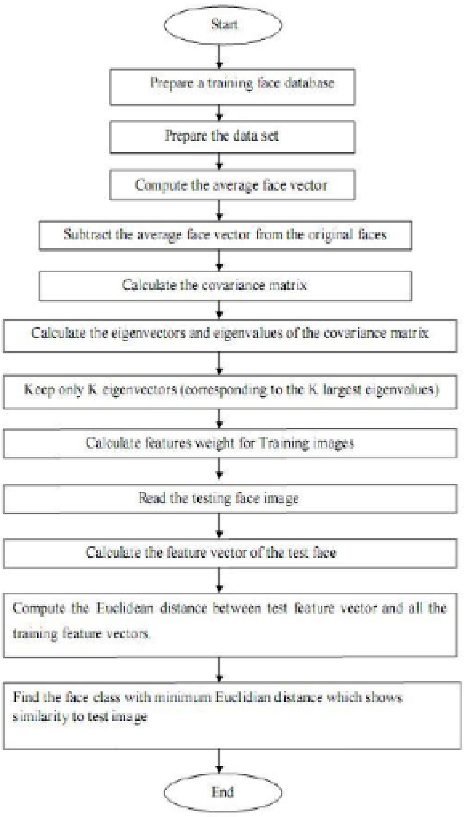
DATA:23/04/2022 **EXP No: 10**	**Title of the Lab** **Implementation of Deep Learning algorithms for an application**	**Name: Avinash Reddy Vasipalli** **Registration Number: RA1911027010007** **Section: N1 Lab Batch: 1** **Day Order: 2**

Aim: Building a ML model for face recognition by applying PCA

Description of concept or problem:

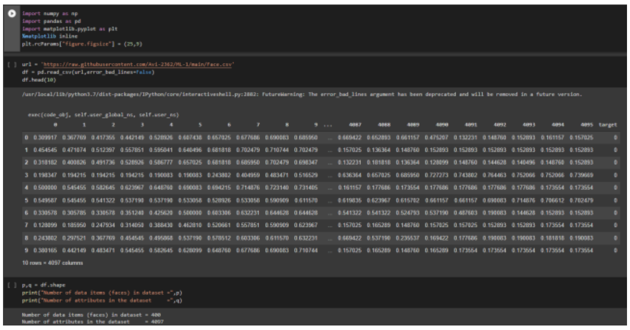
A face recognition using PCA is a statistical approach used for reducing the number of variables in face recognition. In PCA every image in training set is represented as liner combination of weighted called eigenfaces. These Eigen vectors are obtained from covariance matrix of training image set.


Start
Prepare a training face database
Prepare the data set
Compute the average face vector
Subtract the average face vector from the original faces
Calculate the covariance matrix
Calculate the eigenvectors and eigenvalues of the covariance matrix
Keep only K eigenvectors (corresponding to the K largest eigenvalues)
Calculate features weight for Training images
Read the testing face image
Calculate the feature vector of the test face
Compute the Euclidean distance between test feature vector and all the training feature vectors
Find the face class with minimum Euclidian distance which shows similarity to test image
End

The process has 2 steps i.e Initialization and recognition. Initialization process involves acquiring the initial set of face images called training set. Calculate the Eigenfaces from the training set, keeping only high eigen values. M images define the face space. As new faces are experienced the eigenfaces can be updated or recalculated.

Calculate distribution in this M-dimensional space for each known person by projecting his or her face in images onto these faces- spaces

Source Code:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams["figure.figsize"] = (25,9)
```

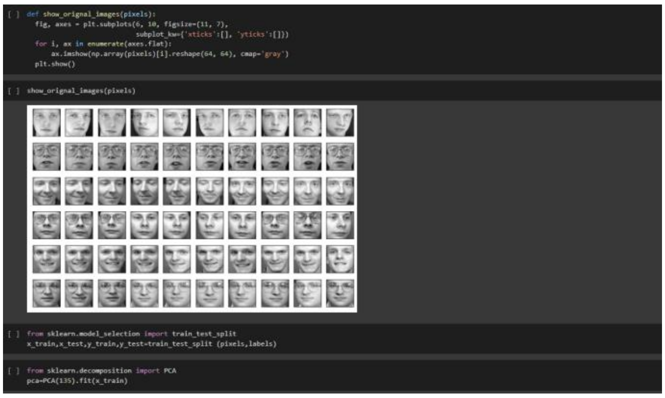
```
url = 'https://raw.githubusercontent.com/Avi-2362/ML-1/main/Face.csv'
df = pd.read_csv(url,error_bad_lines=False)
df.head(10)
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: FutureWarning: The error_bad_lines argument has been deprecated and will be removed in a future version.

  exec(code_obj, self.user_global_ns, self.user_ns)
          0         1         2         3         4         5         6         7         8         9  ...      4087      4088      4089      4090      4091      4092      4093      4094      4095  target
0  0.309917  0.367769  0.417355  0.442149  0.528926  0.607438  0.657025  0.677686  0.690083  0.685950  ...  0.669422  0.652893  0.661157  0.475207  0.132231  0.148760  0.152893  0.161157  0.157025       0
1  0.454545  0.471074  0.512397  0.557851  0.595041  0.640496  0.681818  0.702479  0.710744  0.702479  ...  0.157025  0.136364  0.148760  0.152893  0.152893  0.152893  0.152893  0.152893  0.152893       0
2  0.318182  0.400826  0.491736  0.528926  0.586777  0.657025  0.681818  0.685950  0.702479  0.698347  ...  0.132231  0.181818  0.136364  0.128099  0.148760  0.144628  0.140496  0.148760  0.152893       0
3  0.198347  0.194215  0.194215  0.194215  0.190083  0.190083  0.243802  0.404959  0.483471  0.516529  ...  0.636364  0.657025  0.685950  0.727273  0.743802  0.764463  0.752066  0.752066  0.739669       0
4  0.500000  0.545455  0.582645  0.623967  0.648760  0.690083  0.694215  0.714876  0.723140  0.731405  ...  0.161157  0.177686  0.173554  0.177686  0.177686  0.177686  0.177686  0.173554  0.173554       0
5  0.549587  0.545455  0.541322  0.537190  0.537190  0.533058  0.528926  0.533058  0.590909  0.611570  ...  0.619835  0.623967  0.615702  0.661157  0.661157  0.690083  0.714876  0.706612  0.702479       0
6  0.330578  0.305785  0.330578  0.351240  0.425620  0.500000  0.603306  0.632231  0.644628  0.644628  ...  0.541322  0.541322  0.524793  0.537190  0.487603  0.190083  0.144628  0.152893  0.152893       0
7  0.128099  0.185950  0.247934  0.314050  0.388430  0.462810  0.520661  0.557851  0.590909  0.623967  ...  0.157025  0.165289  0.148760  0.157025  0.157025  0.152893  0.152893  0.173554  0.173554       0
8  0.243802  0.297521  0.367769  0.454545  0.495868  0.537190  0.578512  0.603306  0.611570  0.632231  ...  0.669422  0.537190  0.235537  0.169422  0.177686  0.190083  0.190083  0.181818  0.190083       0
9  0.380165  0.442149  0.483471  0.545455  0.582645  0.628099  0.648760  0.677686  0.690083  0.710744  ...  0.157025  0.165289  0.148760  0.165289  0.173554  0.173554  0.173554  0.173554  0.173554       0
10 rows × 4097 columns
```

```
p,q = df.shape
print("Number of data items (faces) in dataset =",p)
print("Number of attributes in the dataset     =",q)
```

```
Number of data items (faces) in dataset = 400
Number of attributes in the dataset     = 4097
```

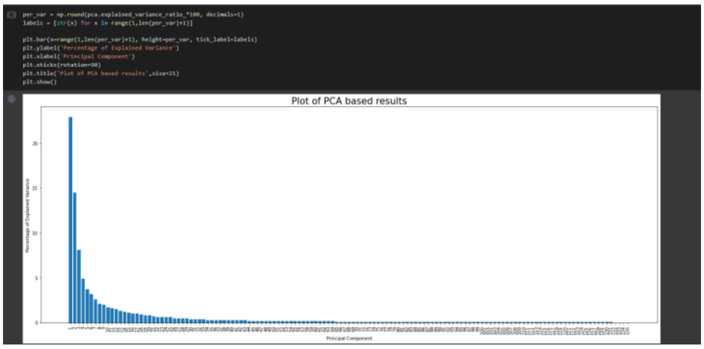


```
def show_orignal_images(pixels):
  fig, axes = plt.subplots(6, 10, figsize=(11, 7),
                           subplot_kw={'xticks':[], 'yticks':[]})
  for i, ax in enumerate(axes.flat):
      ax.imshow(np.array(pixels)[i].reshape(64, 64), cmap='gray')
  plt.show()
```

```
show_orignal_images(pixels)
```

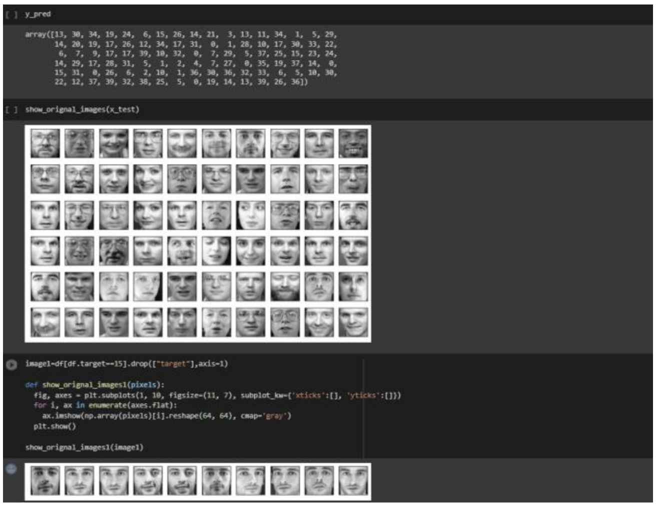
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split (pixels,labels)
```

```
from sklearn.decomposition import PCA
pca=PCA(135).fit(x_train)
```



```
per_var = np.round(pca.explained_variance_ratio_*100, decimals=1)
labels = [str(x) for x in range(1,len(per_var)+1)]

plt.bar(x=range(1,len(per_var)+1), height=per_var, tick_label=labels)
plt.ylabel('Percentage of Explained Variance')
plt.xlabel('Principal Component')
plt.xticks(rotation=90)
plt.title('Plot of PCA based results',size=21)
plt.show()
```



```
y_pred
```

```
array([13, 30, 34, 19, 24,  6, 15, 26, 14, 21,  3, 13, 11, 34,  1,  5, 29,
       14, 20, 19, 17, 26, 12, 34, 17, 31,  0,  1, 28, 10, 17, 30, 33, 22,
        6,  7,  9, 17, 17, 39, 10, 32,  0,  7, 29,  5, 37, 25, 15, 23, 24,
       14, 29, 17, 28, 31,  5,  1,  2,  4,  7, 27,  0, 35, 19, 37, 14,  0,
       15, 31,  0, 26,  6,  2, 10,  1, 36, 30, 36, 32, 33,  6,  5, 10, 30,
       22, 12, 37, 39, 32, 38, 25,  5,  0, 19, 14, 13, 39, 26, 36])
```

```
show_orignal_images(x_test)
```

```
image1=df[df.target==15].drop(["target"],axis=1)

def show_orignal_images1(pixels):
  fig, axes = plt.subplots(1, 10, figsize=(11, 7), subplot_kw={'xticks':[], 'yticks':[]})
  for i, ax in enumerate(axes.flat):
    ax.imshow(np.array(pixels)[i].reshape(64, 64), cmap='gray')
  plt.show()

show_orignal_images1(image1)
```

Conclusion:

The Set images are tested and trained After calculating the eigen values and SVM classification. The final images or to say original images are shown in the end.