Project Report 1

Name: HARI KRISHNAN

Course: AI and ML

Problem Statement: Face Feature Extraction Using PCA

Prerequisites:

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url
https://www.python.org/downloads/ can be referred to download python. Once you have python
downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this:
https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-externalcommand/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.
Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url
https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

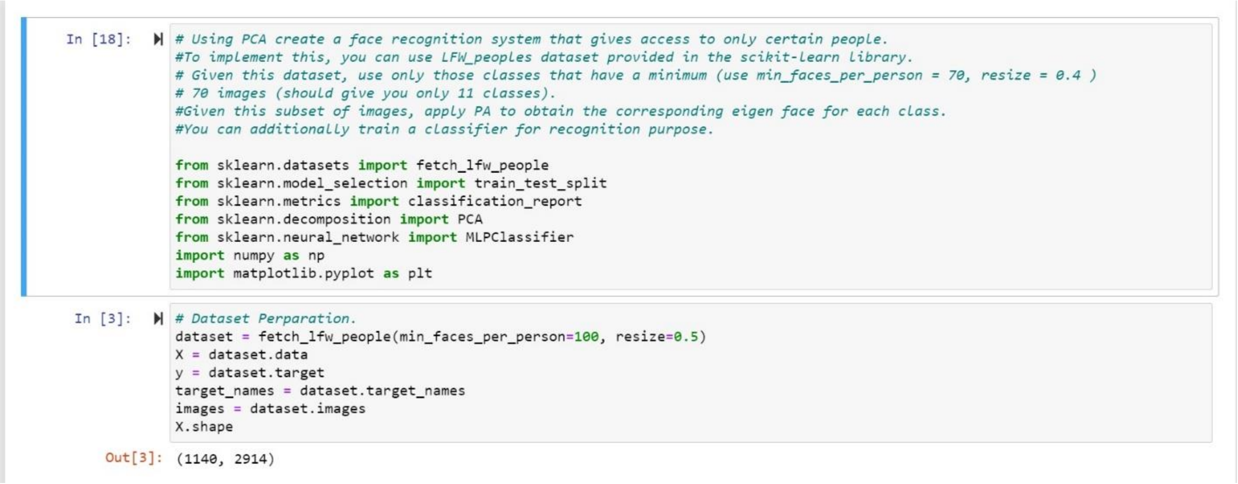
Dataset Used:

The Data source used for this project has been generated using sklearn library.

Method used for Detection

PCA

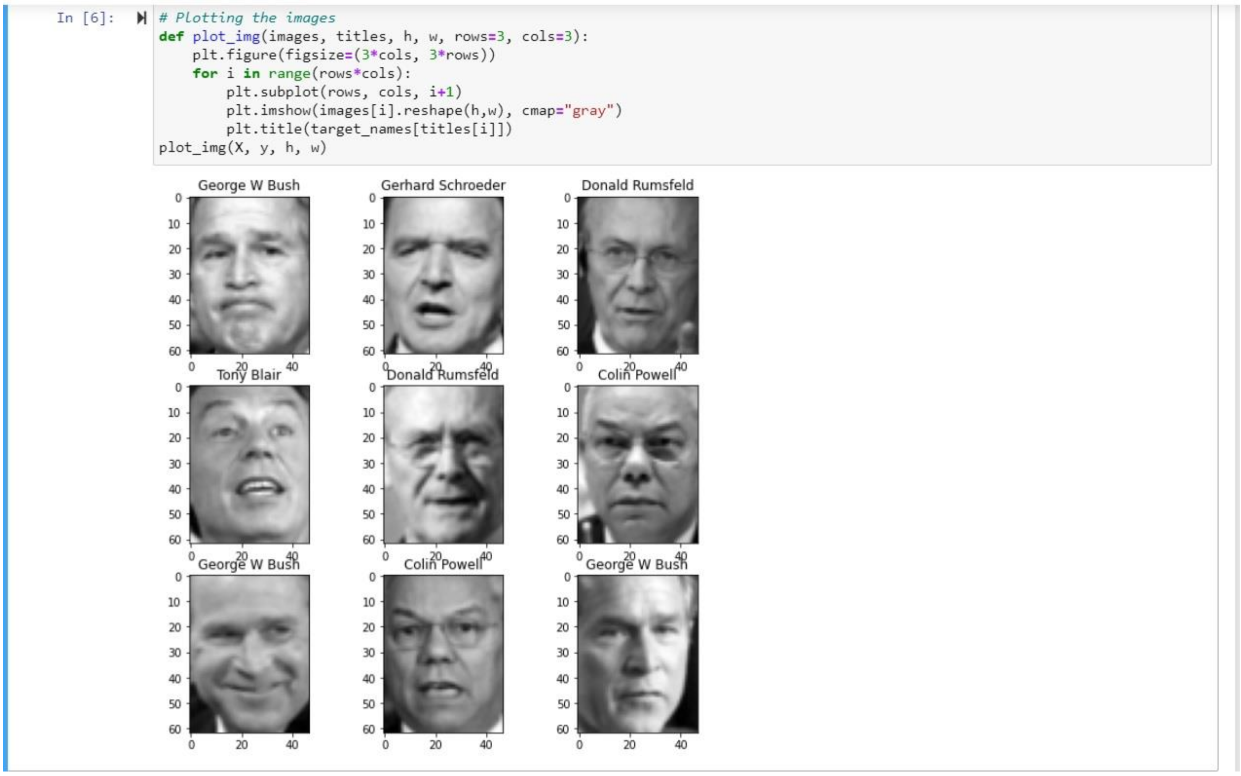
Importing the libraries and capturing images:



```
In [18]:  # Using PCA create a face recognition system that gives access to only certain people.
          #To implement this, you can use LFW_peoples dataset provided in the scikit-learn library.
          # Given this dataset, use only those classes that have a minimum (use min_faces_per_person = 70, resize = 0.4 )
          # 70 images (should give you only 11 classes).
          #Given this subset of images, apply PA to obtain the corresponding eigen face for each class.
          #You can additionally train a classifier for recognition purpose.

          from sklearn.datasets import fetch_lfw_people
          from sklearn.model_selection import train_test_split
          from sklearn.metrics import classification_report
          from sklearn.decomposition import PCA
          from sklearn.neural_network import MLPClassifier
          import numpy as np
          import matplotlib.pyplot as plt
```

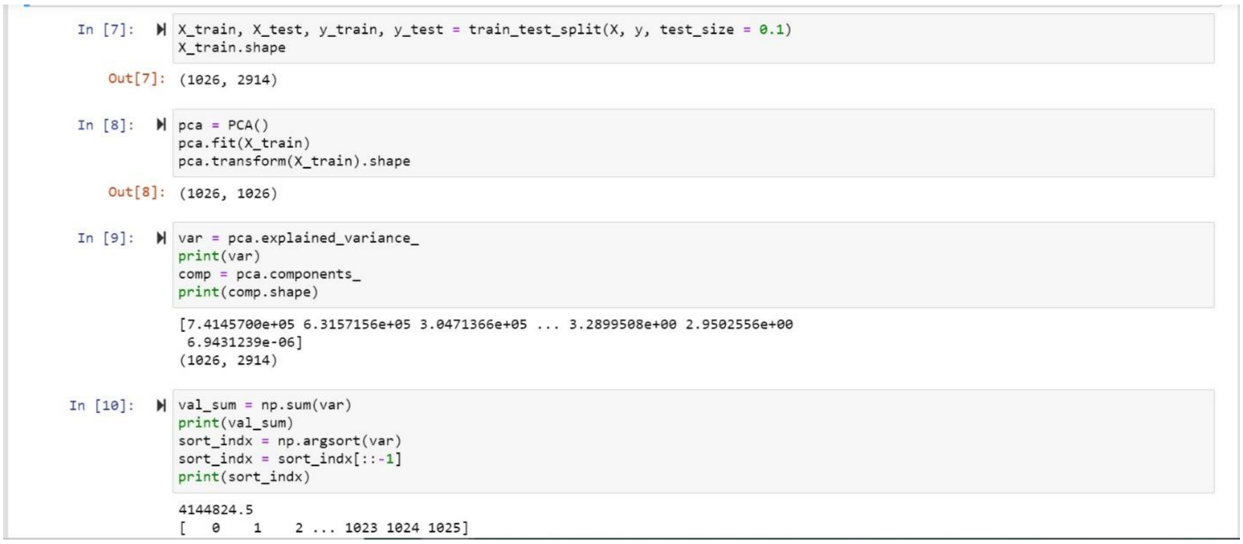
```
In [3]:   # Dataset Perparation.
          dataset = fetch_lfw_people(min_faces_per_person=100, resize=0.5)
          X = dataset.data
          y = dataset.target
          target_names = dataset.target_names
          images = dataset.images
          X.shape

Out[3]:   (1140, 2914)
```

Plot the images:

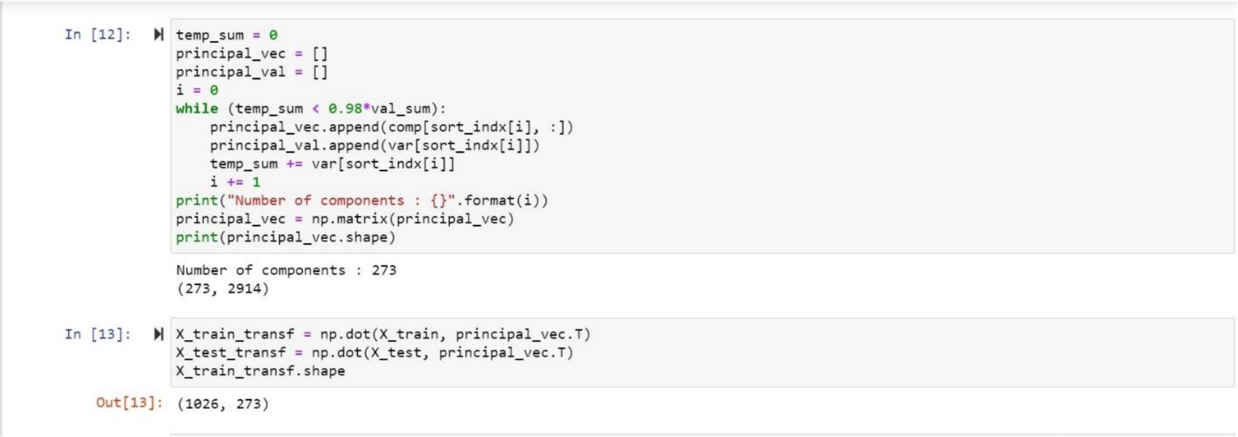


```
In [6]:   # Plotting the images
          def plot_img(images, titles, h, w, rows=3, cols=3):
              plt.figure(figsize=(3*cols, 3*rows))
              for i in range(rows*cols):
                  plt.subplot(rows, cols, i+1)
                  plt.imshow(images[i].reshape(h,w), cmap="gray")
                  plt.title(target_names[titles[i]])
          plot_img(X, y, h, w)
```

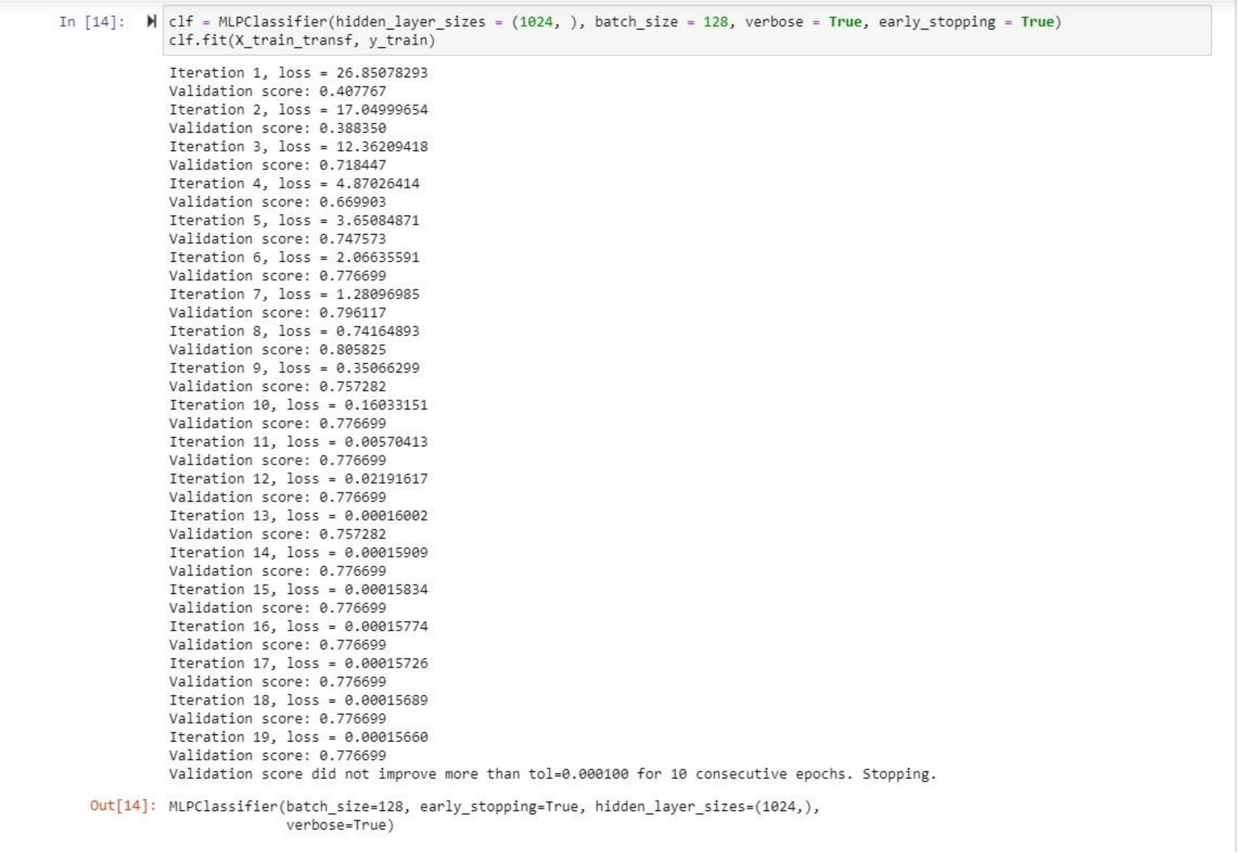
Train the data:



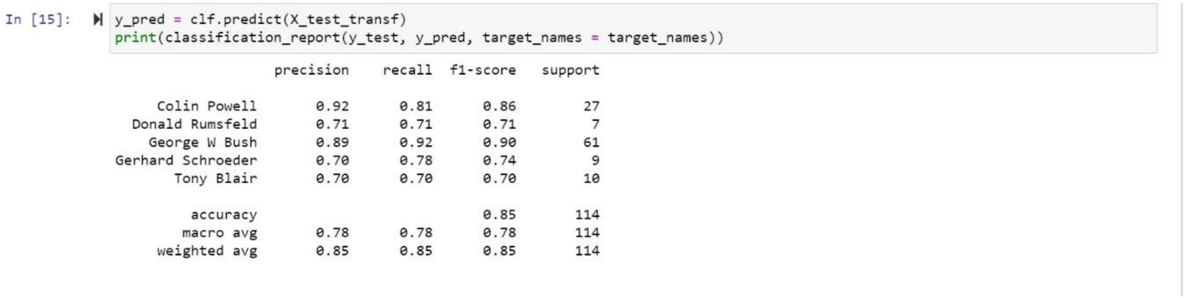
```
In [7]:  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)
         X_train.shape

Out[7]: (1026, 2914)

In [8]:  pca = PCA()
         pca.fit(X_train)
         pca.transform(X_train).shape

Out[8]: (1026, 1026)

In [9]:  var = pca.explained_variance_
         print(var)
         comp = pca.components_
         print(comp.shape)

         [7.4145700e+05 6.3157156e+05 3.0471366e+05 ... 3.2899508e+00 2.9502556e+00
          6.9431239e-06]
         (1026, 2914)

In [10]: val_sum = np.sum(var)
         print(val_sum)
         sort_indx = np.argsort(var)
         sort_indx = sort_indx[::-1]
         print(sort_indx)

         4144824.5
         [   0    1    2 ... 1023 1024 1025]
```



```
In [12]: temp_sum = 0
         principal_vec = []
         principal_val = []
         i = 0
         while (temp_sum < 0.98*val_sum):
             principal_vec.append(comp[sort_indx[i], :])
             principal_val.append(var[sort_indx[i]])
             temp_sum += var[sort_indx[i]]
             i += 1
         print("Number of components : {}".format(i))
         principal_vec = np.matrix(principal_vec)
         print(principal_vec.shape)

         Number of components : 273
         (273, 2914)

In [13]: X_train_transf = np.dot(X_train, principal_vec.T)
         X_test_transf = np.dot(X_test, principal_vec.T)
         X_train_transf.shape

Out[13]: (1026, 273)
```

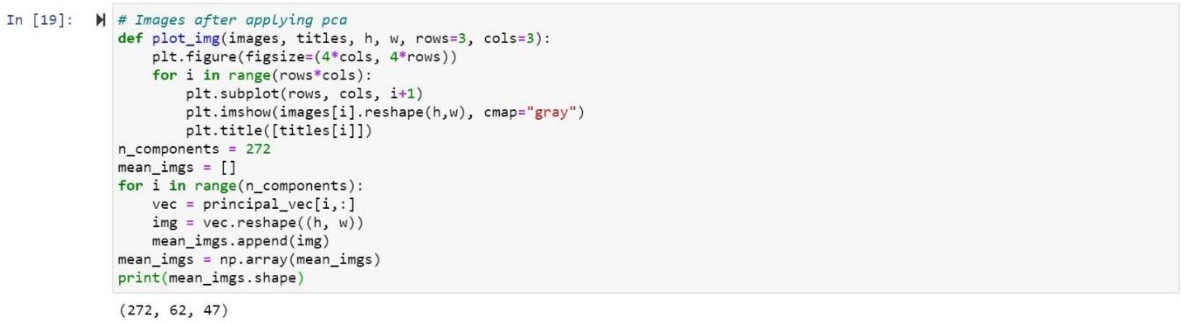


```
In [14]: clf = MLPClassifier(hidden_layer_sizes = (1024, ), batch_size = 128, verbose = True, early_stopping = True)
         clf.fit(X_train_transf, y_train)

         Iteration 1, loss = 26.85078293
         Validation score: 0.407767
         Iteration 2, loss = 17.04999654
         Validation score: 0.388350
         Iteration 3, loss = 12.36209418
         Validation score: 0.718447
         Iteration 4, loss = 4.87026414
         Validation score: 0.669903
         Iteration 5, loss = 3.65084871
         Validation score: 0.747573
         Iteration 6, loss = 2.06635591
         Validation score: 0.776699
         Iteration 7, loss = 1.28096985
         Validation score: 0.796117
         Iteration 8, loss = 0.74164893
         Validation score: 0.805825
         Iteration 9, loss = 0.35066299
         Validation score: 0.757282
         Iteration 10, loss = 0.16033151
         Validation score: 0.776699
         Iteration 11, loss = 0.00570413
         Validation score: 0.776699
         Iteration 12, loss = 0.02191617
         Validation score: 0.776699
         Iteration 13, loss = 0.00016002
         Validation score: 0.757282
         Iteration 14, loss = 0.00015909
         Validation score: 0.776699
         Iteration 15, loss = 0.00015834
         Validation score: 0.776699
         Iteration 16, loss = 0.00015774
         Validation score: 0.776699
         Iteration 17, loss = 0.00015726
         Validation score: 0.776699
         Iteration 18, loss = 0.00015689
         Validation score: 0.776699
         Iteration 19, loss = 0.00015660
         Validation score: 0.776699
         Validation score did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

Out[14]: MLPClassifier(batch_size=128, early_stopping=True, hidden_layer_sizes=(1024,),
                       verbose=True)
```



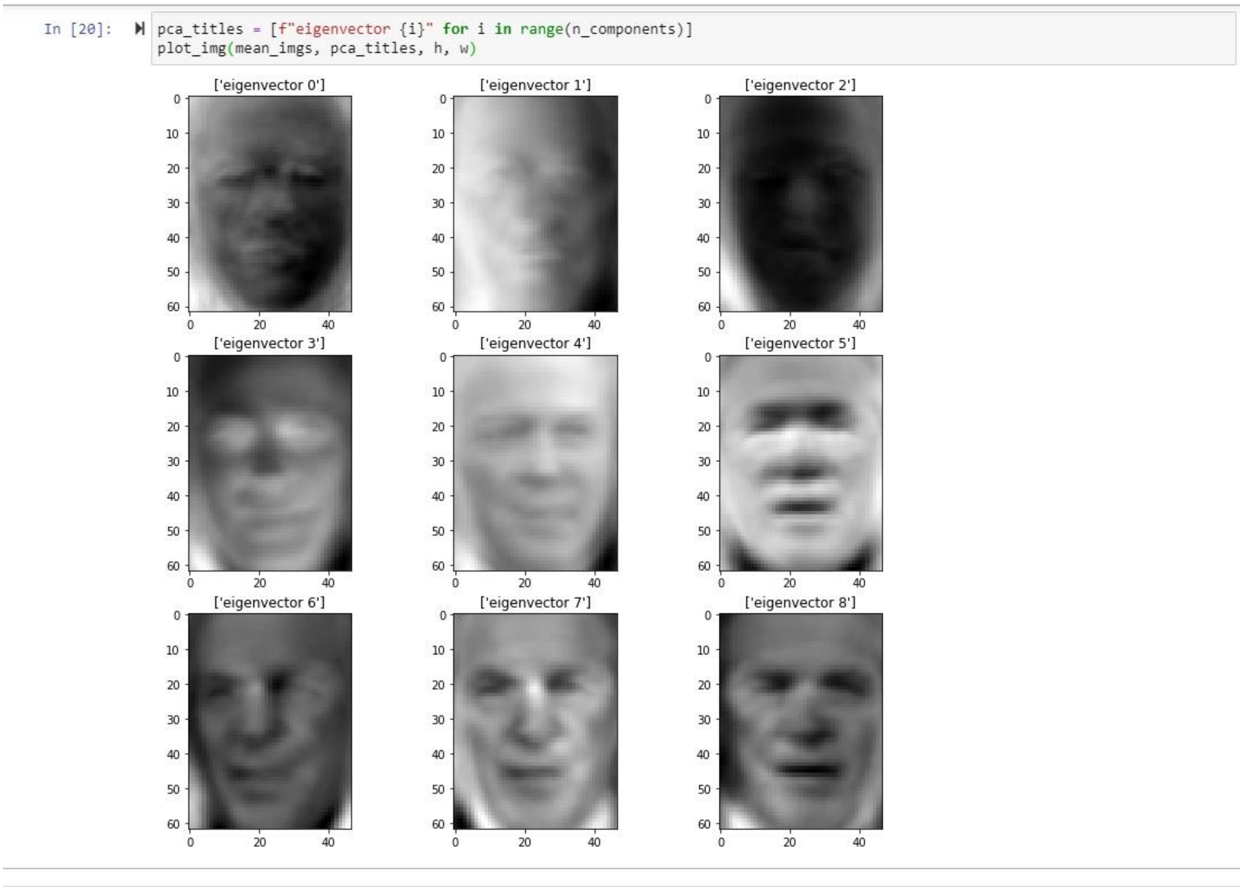
```
In [15]:  y_pred = clf.predict(X_test_transf)
           print(classification_report(y_test, y_pred, target_names = target_names))

                           precision    recall  f1-score   support

                Colin Powell       0.92      0.81      0.86        27
             Donald Rumsfeld       0.71      0.71      0.71         7
               George W Bush       0.89      0.92      0.90        61
           Gerhard Schroeder       0.70      0.78      0.74         9
                  Tony Blair       0.70      0.70      0.70        10

                    accuracy                           0.85       114
                   macro avg       0.78      0.78      0.78       114
                weighted avg       0.85      0.85      0.85       114
```

Plot the images after applying PCA:



```
In [19]:  # Images after applying pca
           def plot_img(images, titles, h, w, rows=3, cols=3):
               plt.figure(figsize=(4*cols, 4*rows))
               for i in range(rows*cols):
                   plt.subplot(rows, cols, i+1)
                   plt.imshow(images[i].reshape(h,w), cmap="gray")
                   plt.title([titles[i]])
           n_components = 272
           mean_imgs = []
           for i in range(n_components):
               vec = principal_vec[i,:]
               img = vec.reshape((h, w))
               mean_imgs.append(img)
           mean_imgs = np.array(mean_imgs)
           print(mean_imgs.shape)

           (272, 62, 47)
```

Output:



```
In [20]:  pca_titles = [f"eigenvector {i}" for i in range(n_components)]
           plot_img(mean_imgs, pca_titles, h, w)
```