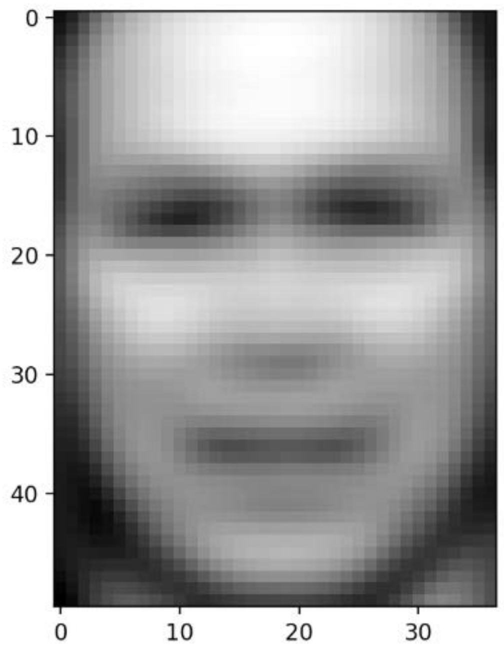
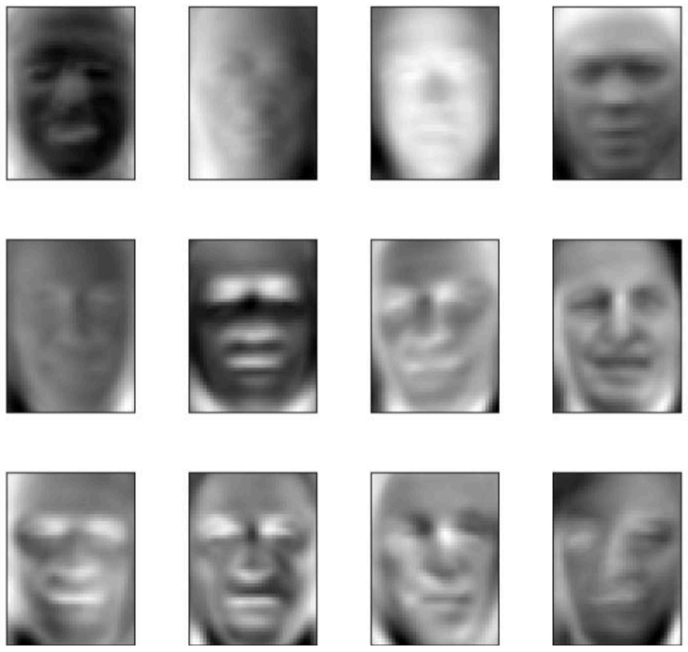
1) PCA and Image Reconstruction

a)



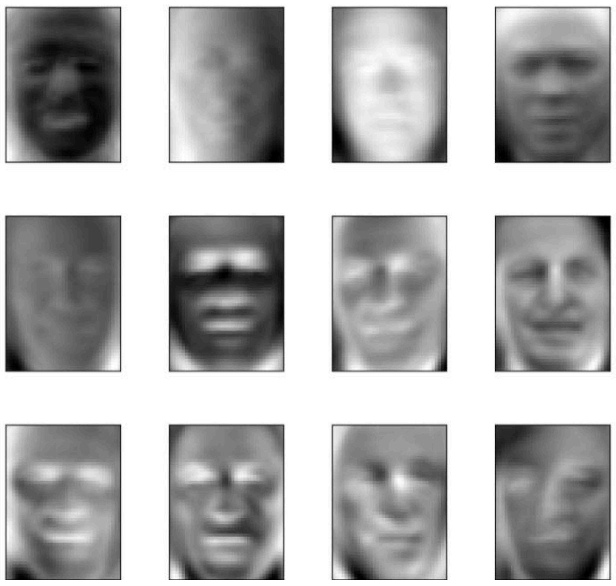
This “average” face is blurry to the point where one cannot identify who the face represents, but it is still identifiable to be a face. We cannot see specific features such as eyes, nose, and mouth, but we are able to see a general outline of these facial features.

b)

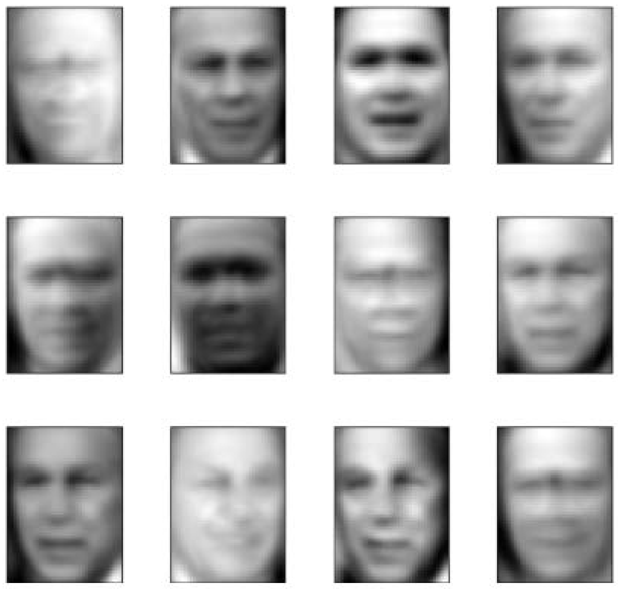


The top 12 eigenfaces are all blurry and they all are indistinguishable in terms of identity. They vary in minor ways and you can tell that they are different faces, yet they all are similar in the sense that their eyes, noses, and mouths are not defined. They each have different shades of light on their face, some with more shadows and brighter eye sockets, whereas others have whiter highlights on their face and their eye sockets are darker. I think that these photos were selected as the top eigenfaces because they are the best representation of the training data. Because each of these faces vary in one way or another, they represent the diversity of the many training examples. In terms of basis vectors, these 12 eigenfaces have the top 12 highest variances.

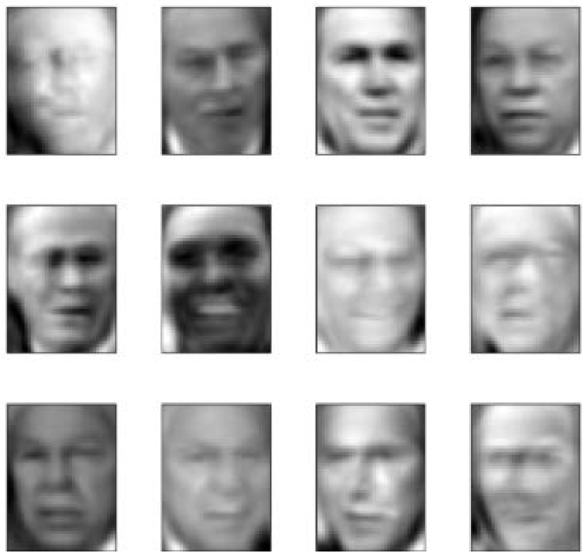
c)



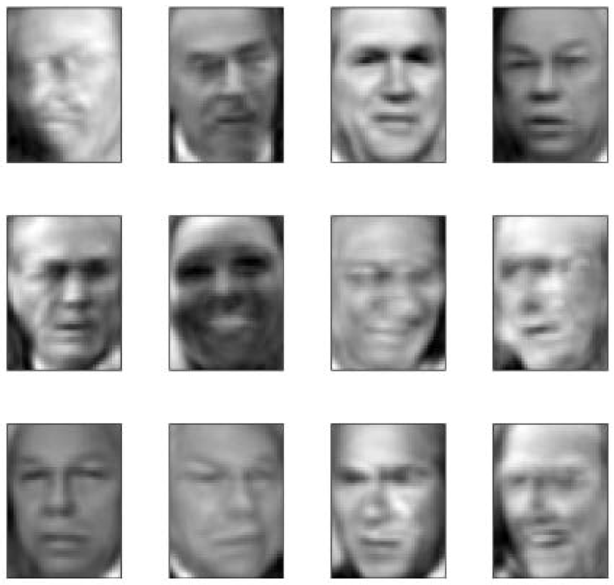
L = 1



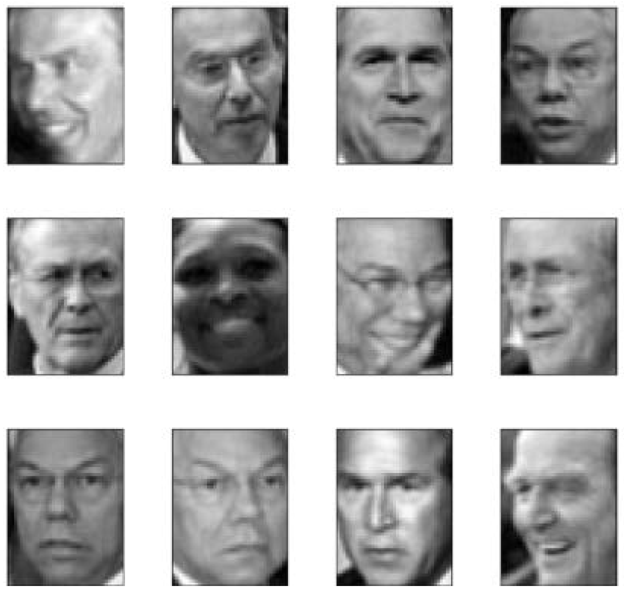
L = 10



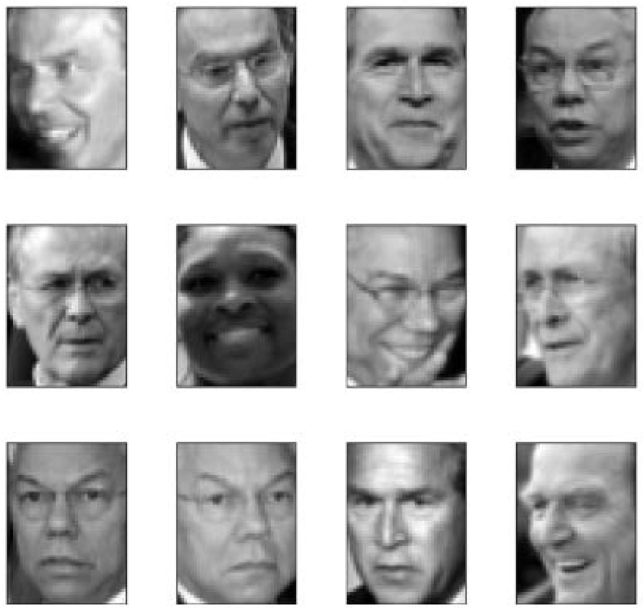
L = 50



L = 100



L = 500



L = 1288

Increasing the value of l gradually makes the images more clear and distinguishable. At around L = 50, the facial features become more identifiable and you can start seeing greater differences between the 12 eigenfaces. As we get to larger L values > 100, the faces are extremely clear and we can tell who each person is. We notice that much larger jumps in L values must be used in order to get the slightest transformations in clarity.

② k-Means and k-Medoids

A) $\mu_j \in \mathbb{R}^d$, $j \in \{1, \dots, k\}$

$c^{(i)} \in \{1, \dots, k\}$, $i \in \{1, \dots, n\}$

$$J(c, \mu) = \sum_{i=1}^{n} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

If we minimize the objective function over μ, c, and k, then that allows us to set k, the # of clusters, equal to n, the # of cluster assignments. If $\underline{k = n}$, then each data point will be determined as its own cluster and thus the distance between each data point and the nearest cluster will be $\emptyset$. This can be computationally expressed by:

$$\mu_j = x_j \quad \forall \; j \in \{1, \dots, n\}$$

mean of clusters

$$c^{(i)} = i \quad \forall \; i \in \{1, \dots, n\}$$

cluster assignments

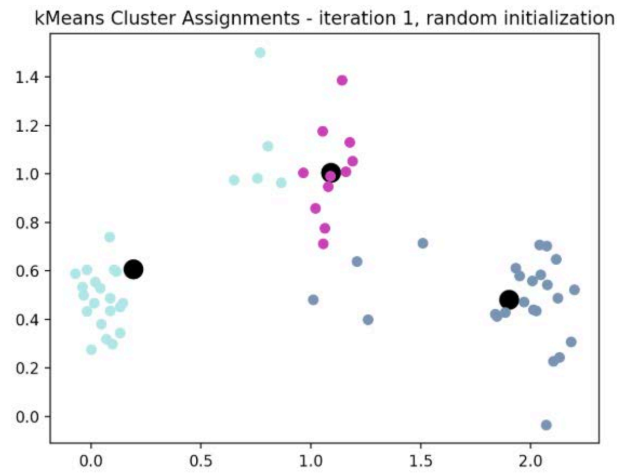
$$(x^{(i)} - \mu_{c^{(i)}})^2$$
$$= (x^{(i)} - \mu_{(i)})^2$$
$$= (x^{(i)} - x^{(i)})^2$$
$$= 0^2$$
$$= 0$$

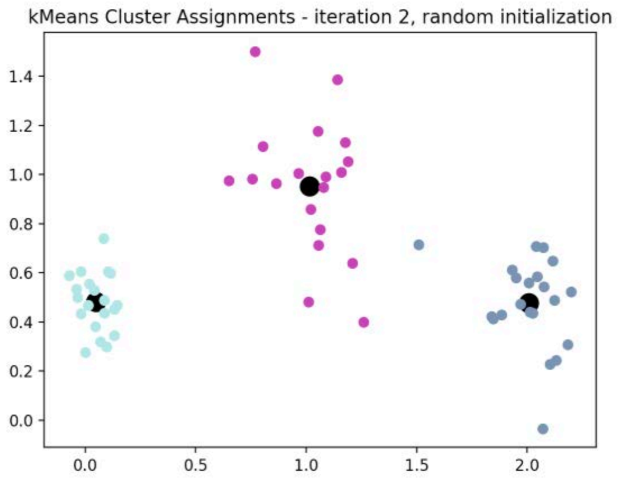
$$\Rightarrow J(c, \mu) = \sum_{i=1}^{n} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$
$$= \sum_{i=1}^{n} (0)$$

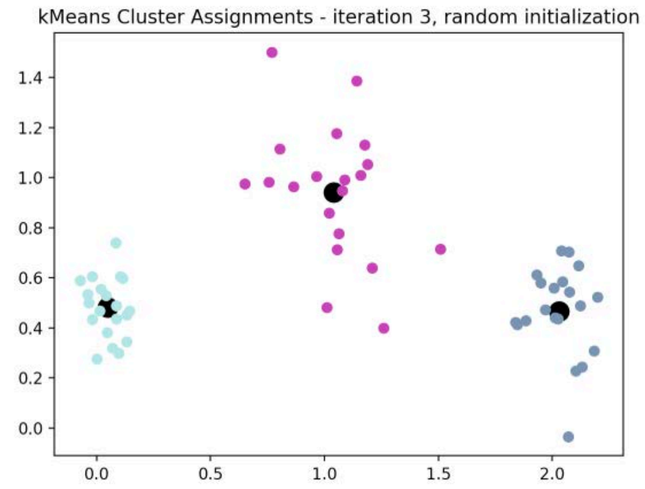
min possible value $\rightarrow$ $\boxed{J(c, \mu, k) = 0}$

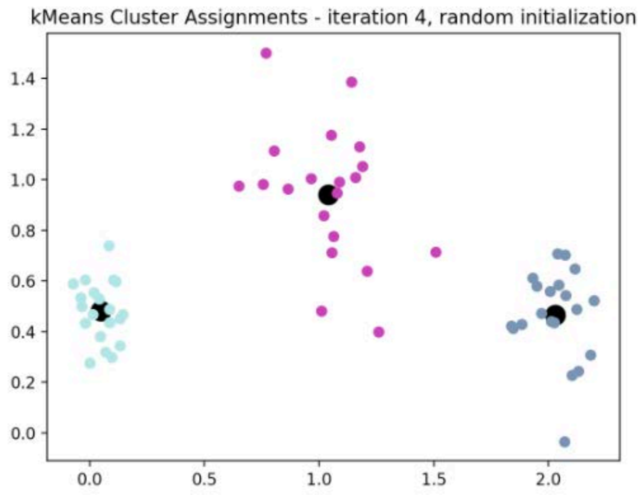
2) K-Means and K-Medoids

d)

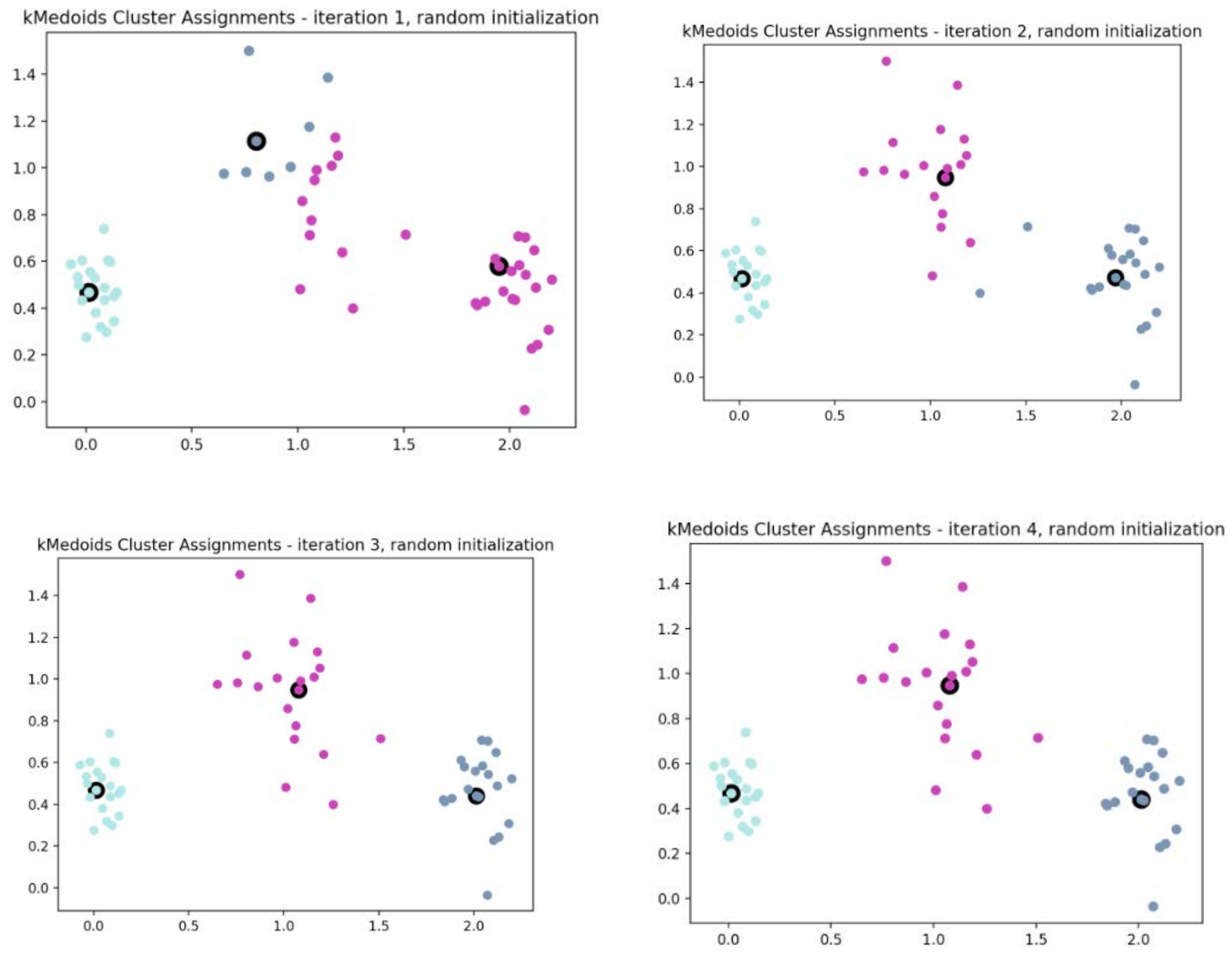




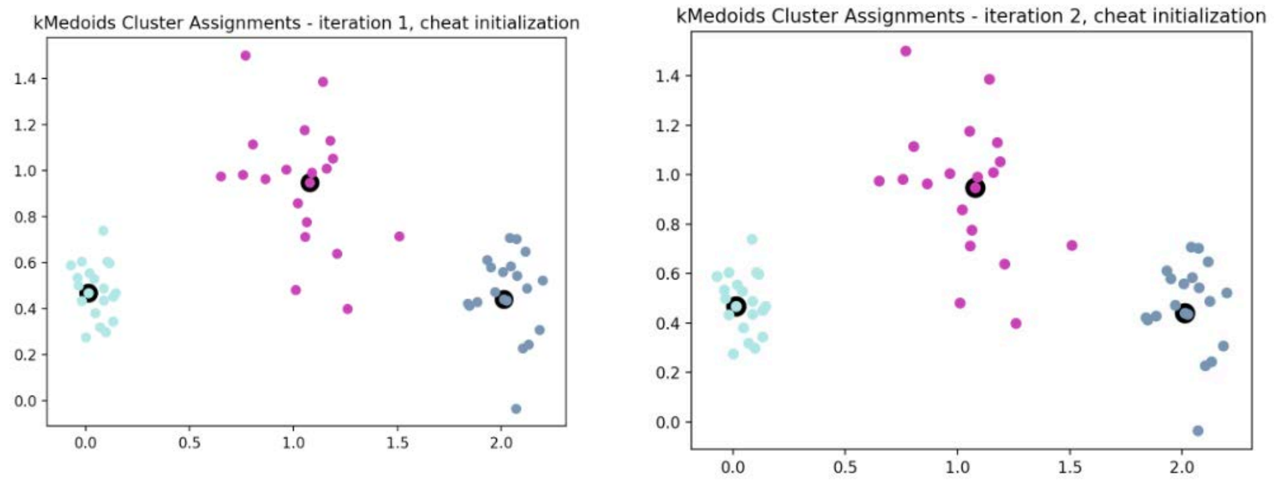




e)


kMedoids Cluster Assignments - iteration 1, random initialization
kMedoids Cluster Assignments - iteration 2, random initialization
kMedoids Cluster Assignments - iteration 3, random initialization
kMedoids Cluster Assignments - iteration 4, random initialization
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.5
1.5
2.0

f)


kMedoids Cluster Assignments - iteration 1, cheat initialization
kMedoids Cluster Assignments - iteration 2, cheat initialization
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.5
1.5
2.0

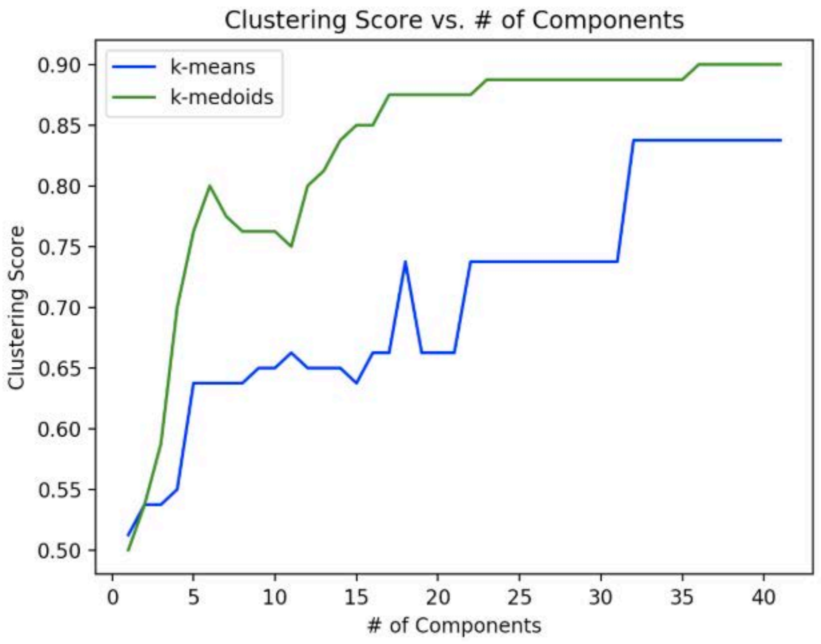
3) Clustering Faces

a)

	average	min	max
k-means	0.6175	0.55	0.775
k-medoids	0.6325	0.575	0.725

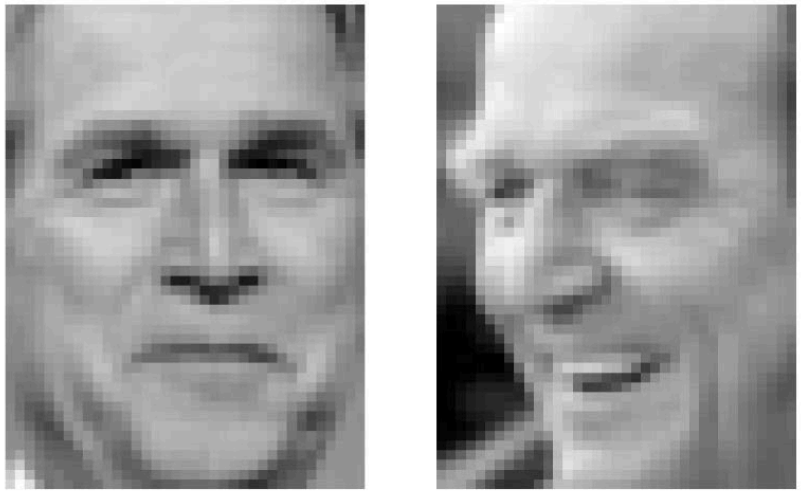
The k-Means algorithm had better runtime at around 0.15564 seconds compared to k-Means who had a runtime around 0.256 seconds. k-Medoids had a higher average performance, but k-Means had a higher maximum.

b)



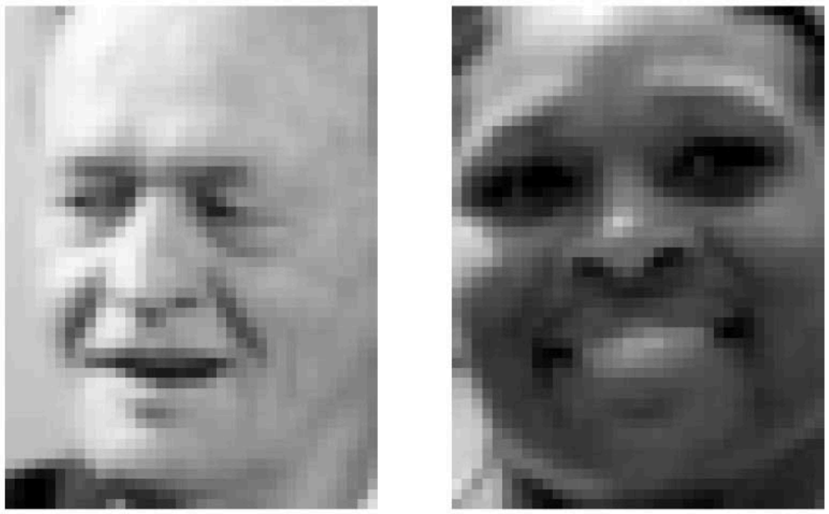
For both k-Means and k-Medoids, the cluster score gradually increases as the number of components increases. The increase for both is, however, not perfectly linearly since there are a lot of jumps and falls, showing that a larger number of components does not necessarily always correlate to a higher clustering score. But the general trend shows that more components will allow the algorithm to distinguish the data points into separate clusters better, thus producing a higher clustering score. Additionally, the plots begin to level out around 17 components for the k-Medoids and around 32 for the k-Means. Overall, the k-Medoids algorithm has better performance than the k-Means algorithm, which makes sense because medoids are restricted to be actual members of the data set whereas means are an average of the data points in a cluster. K-Medoids is more robust in the sense that it is not as sensitive to outliers as k-means is, thus potentially improving the performance of the algorithm.

c) Pair that clustering found difficult to discriminate



Labels 4 & 5
Clustering score = 0.5125

Pair that clustering found easy to discriminate



Labels 9 & 16
Clustering score = 0.9875

In order to find a pair that clustering can discriminate very well and one that finds it very difficult, I used nested for loops to iteratively compare all 40 photos that are not the same photo. If the two photos are different, I would use the build_face_image_ponts in order to translate these images to labeled points. Using these labeled points, I would then run the k-Medoids algorithms with k = 2 clusters. Using the score() function in cluster.py, I would then compute the cluster score of the medoid clusters. While still iterating, I would get the minimum and maximum cluster scores along with the associated labels of both photos and store them in a variable that can be used outside the nested for loops. Once I had these maximum and minimum scores, I used the plot_representative_images function to plot the images given the labels.