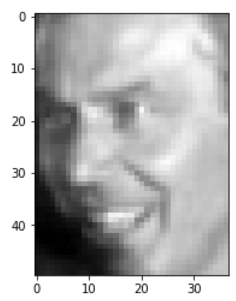
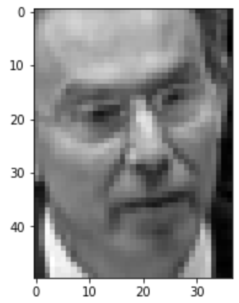
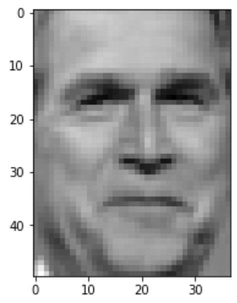
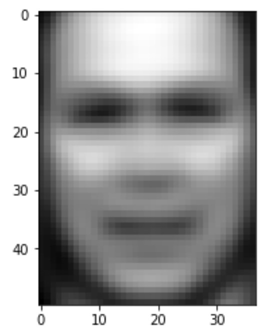
1a)

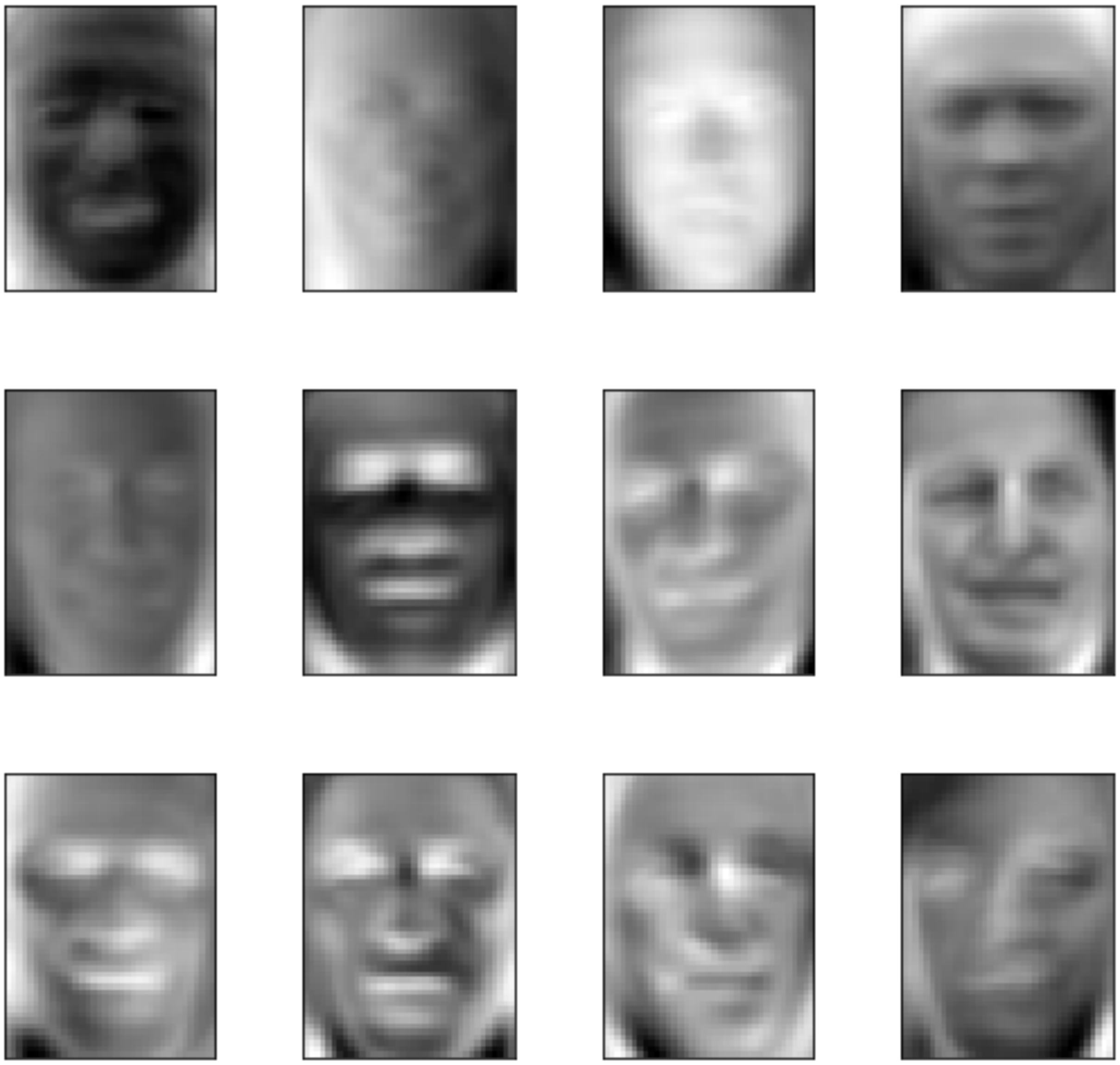
First couple of images –



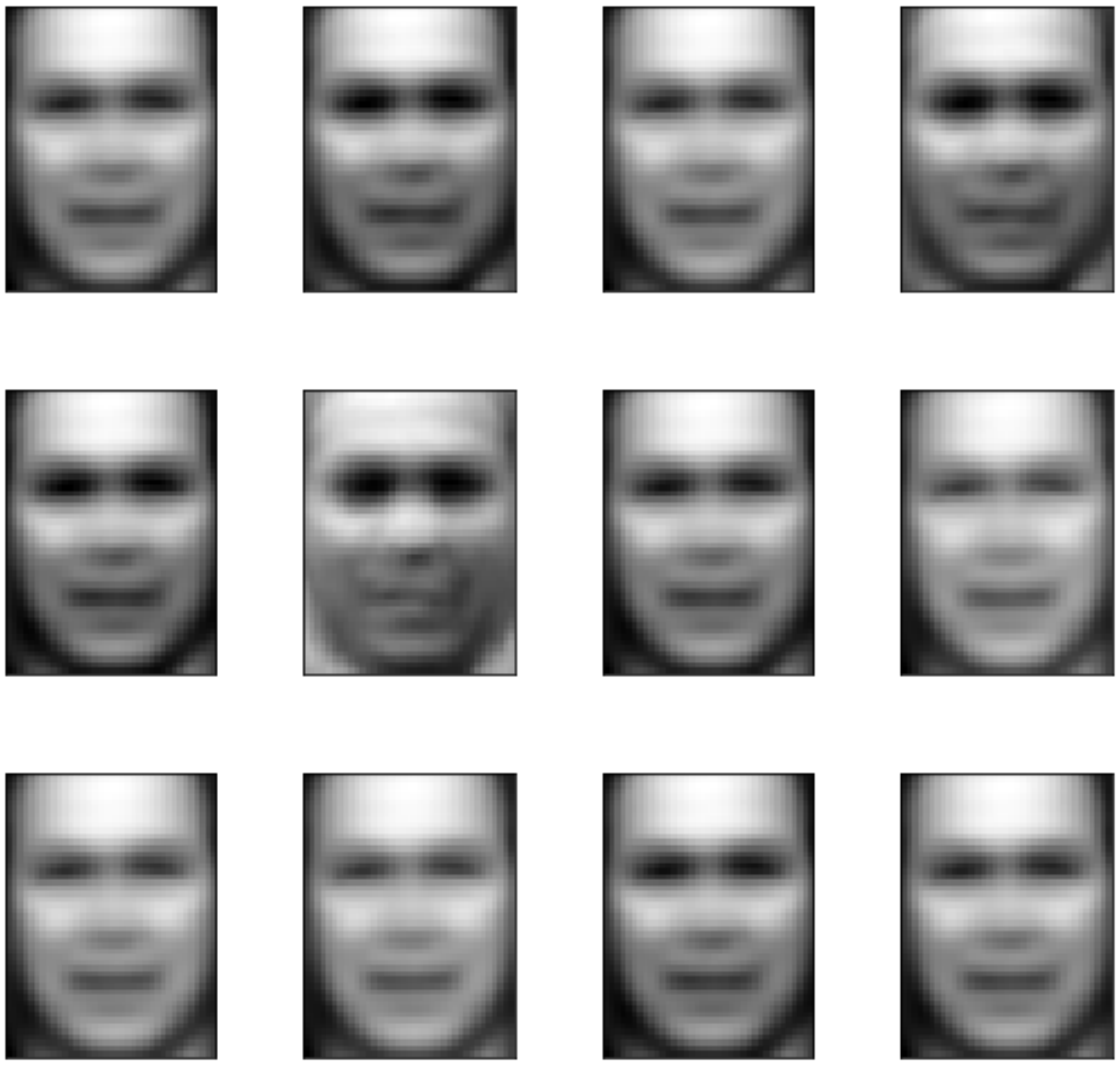
The average face seems to have the basic distinguishing features of a face – eyes, nose, a mouth, chin and a general shape of the face. It is also very blurry.

b)

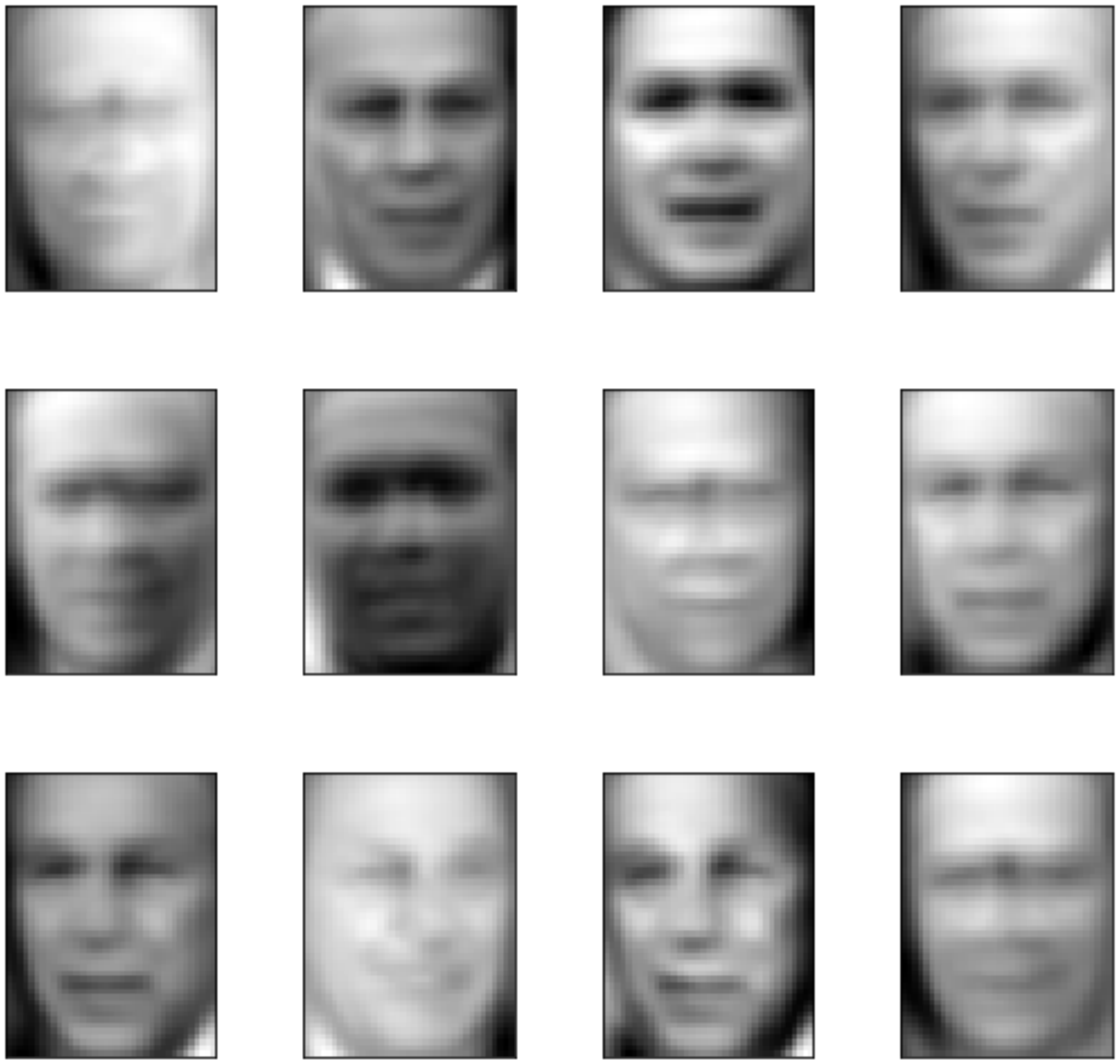


I think these were selected because they vary significantly from each other in terms of facial features and the color contrast of the picture and hence they most closely capture the dataset.

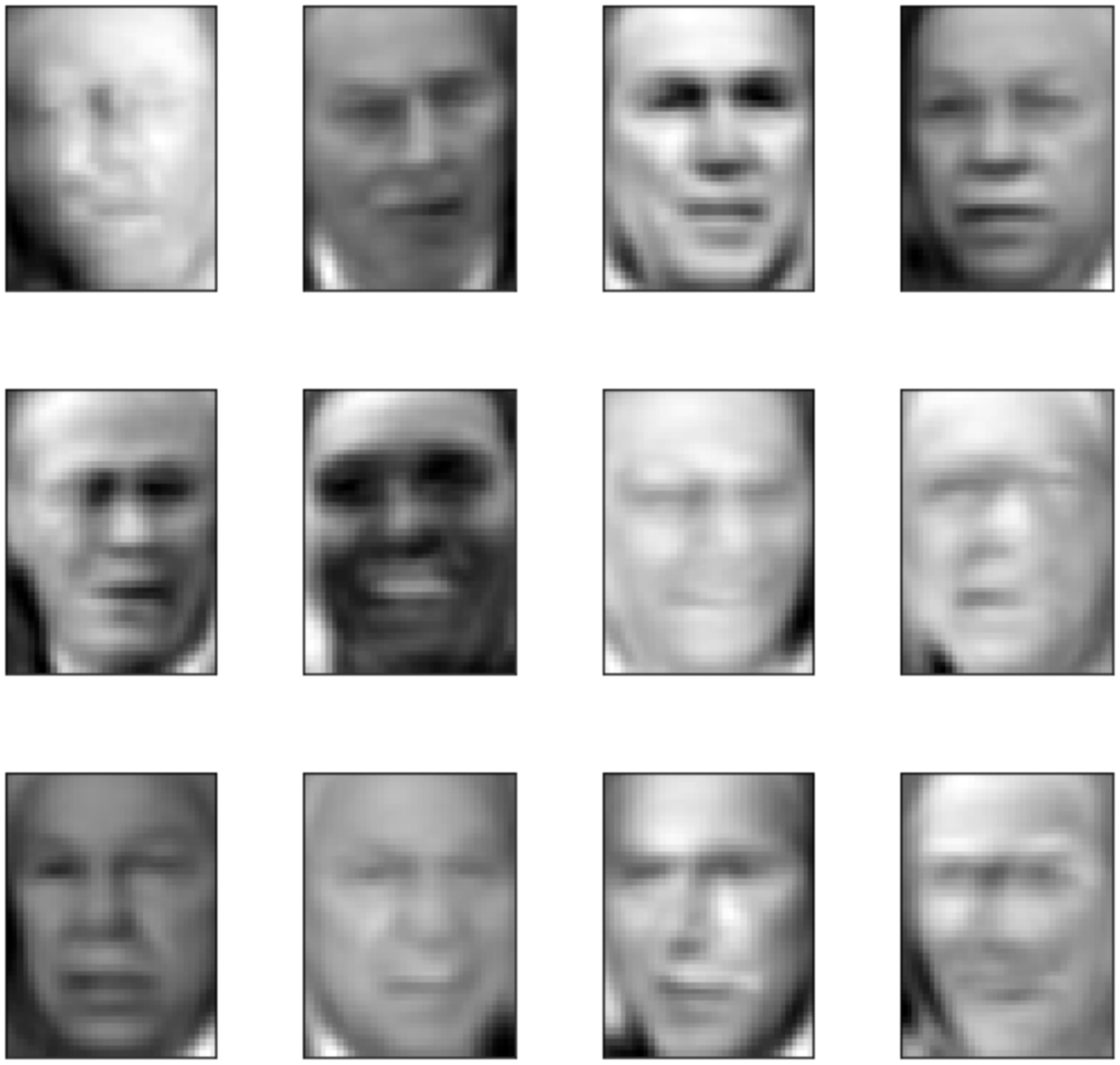
c)
l= 1



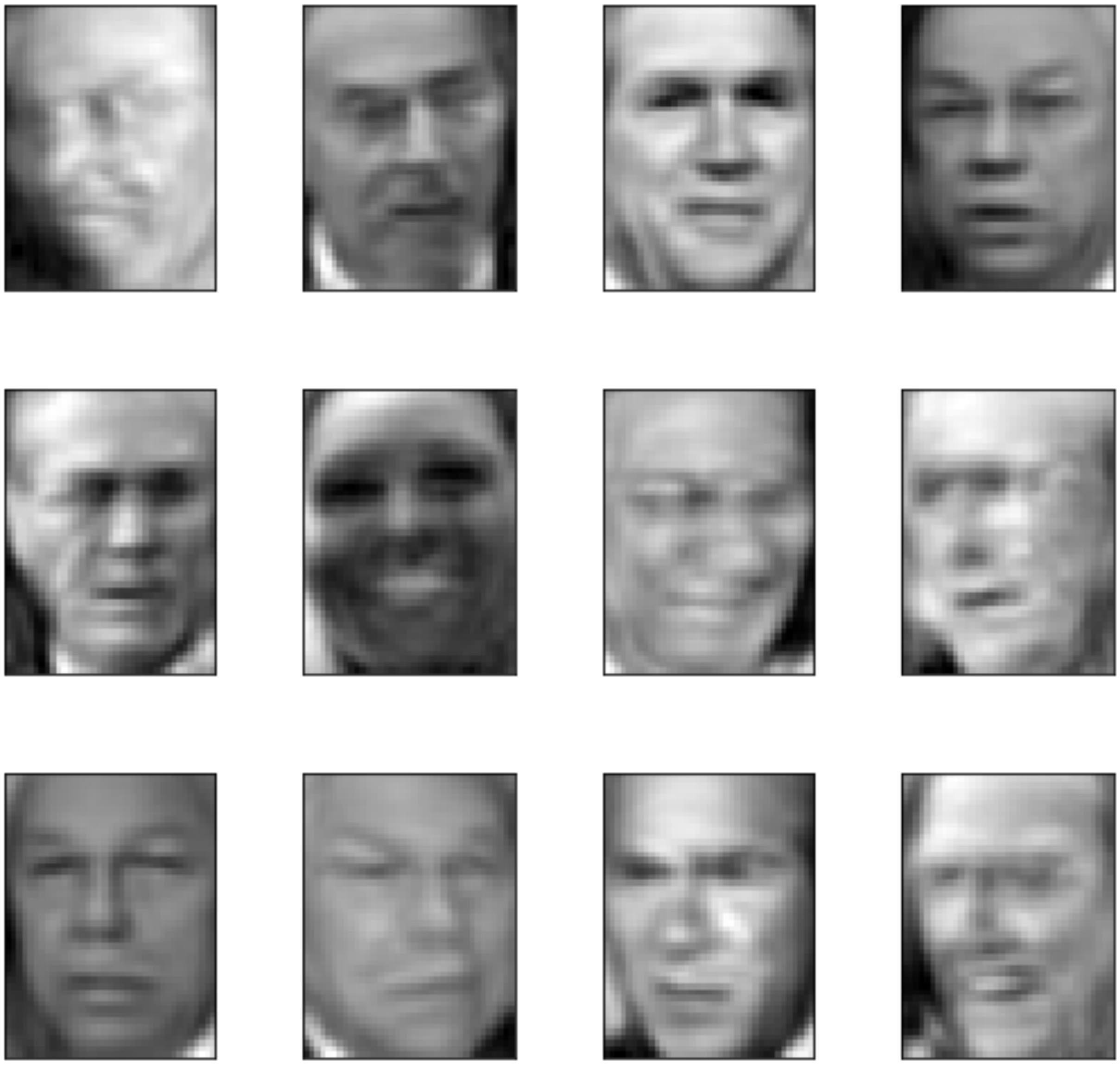
l = 10



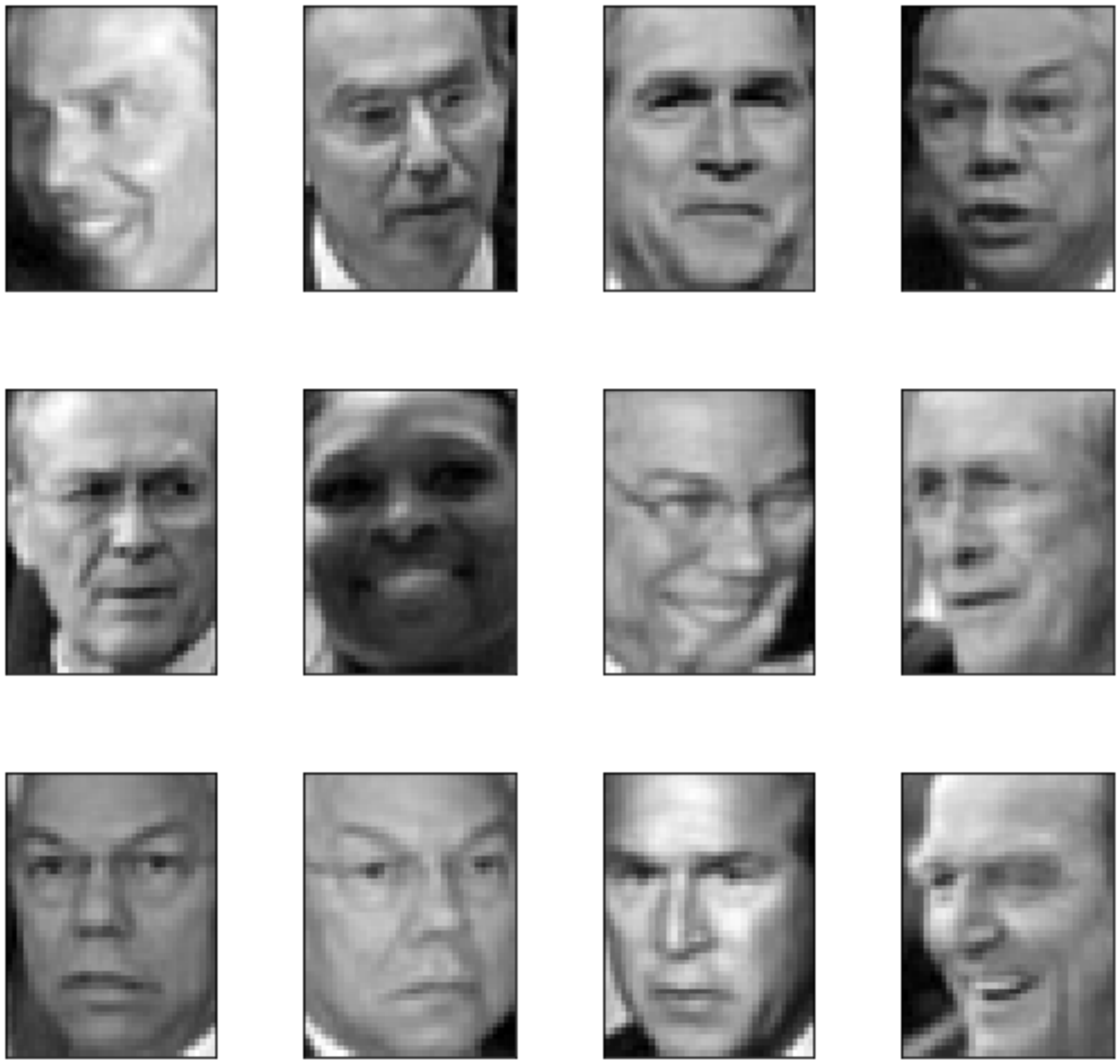
l = 50



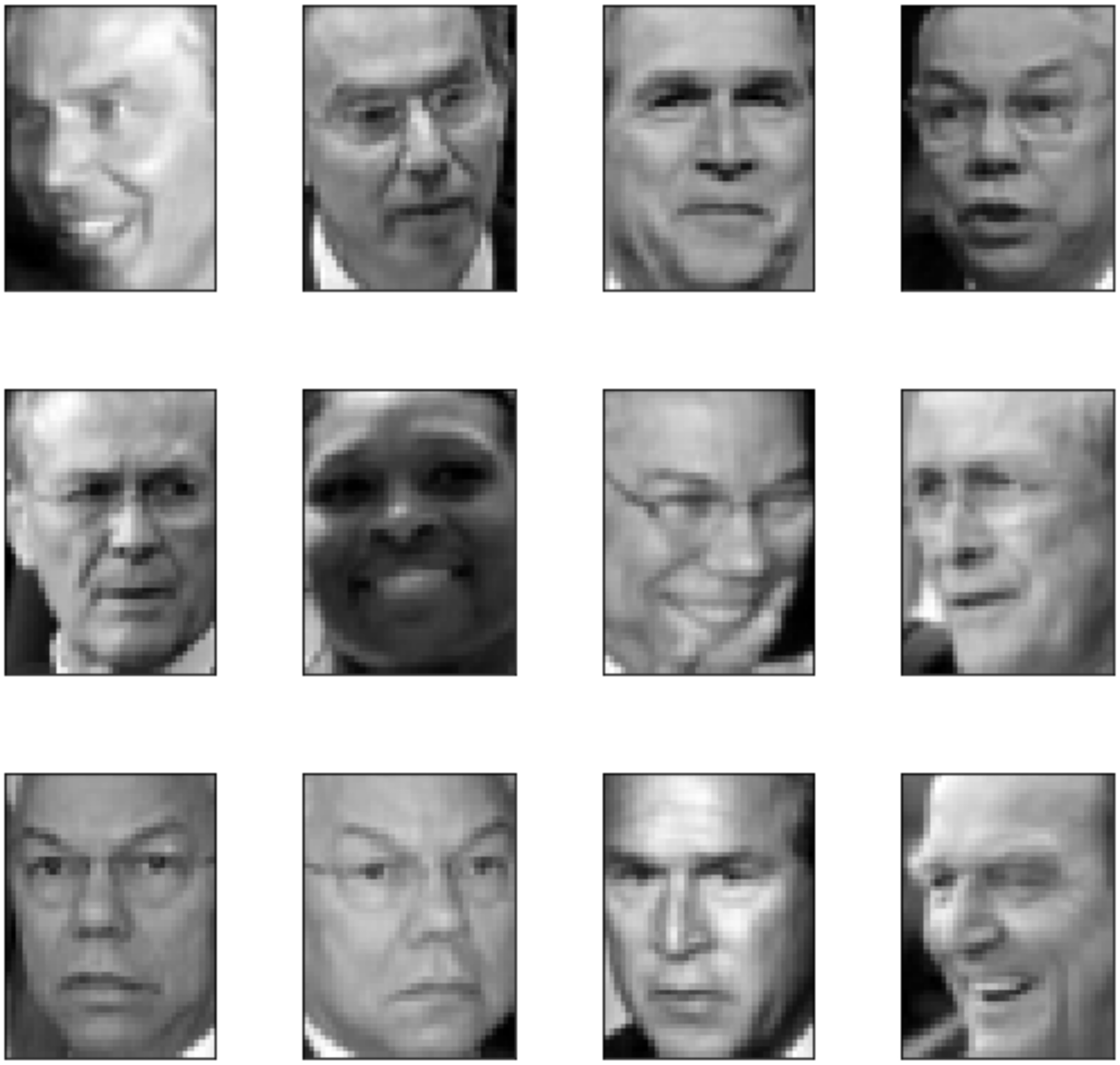
l = 100



l = 500



l = 1288



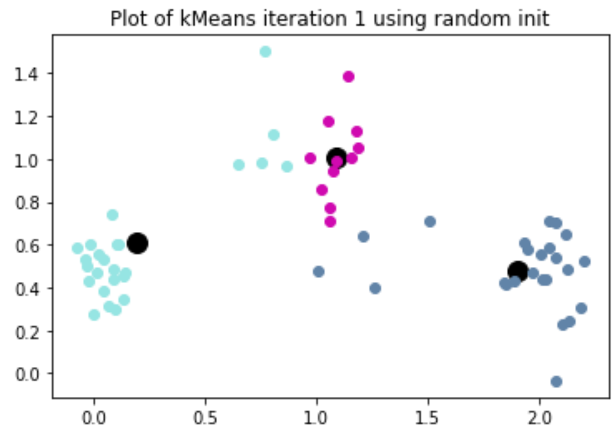
When a smaller number of components are used to represent the images, they have fewer distinguishing features from each other and facial recognition is very generalized. Increasing the number of components makes facial recognition more specific and the images more varied from one another as there are more features that are taken into account in the construction of each image. However, beyond a certain number, as seen from the difference in images between l = 500 and l = 1288, increasing the number of components does not add significant value or variation to these images.

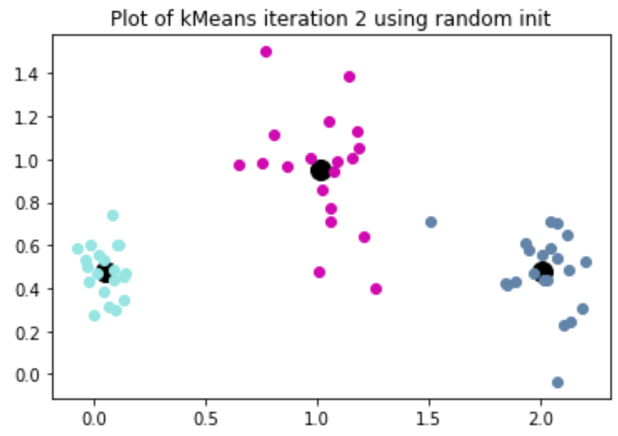
2)

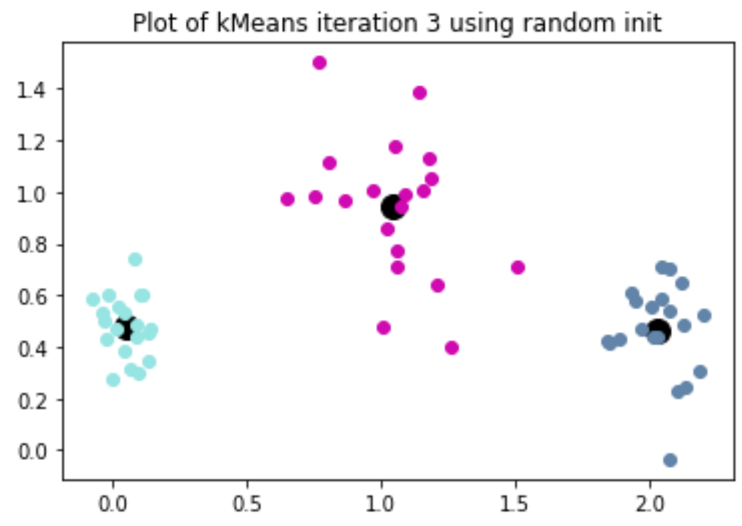
a) The minimum possible value would be 0 which would happen if the cluster center is the point itself (each cluster has one data point, $c^{(i)} = i$ and $u_j = x_j$) and when k = n. This would be overfitting the data and will give us no clustering information about the data.

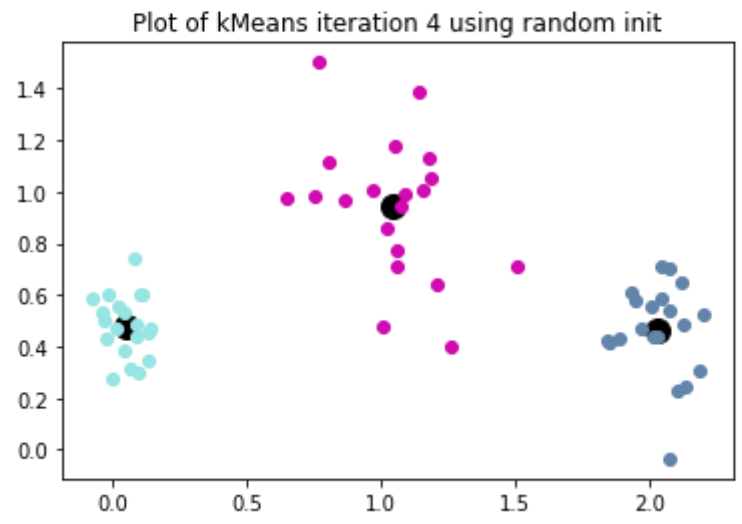
Note: For parts d, e and f, the last 2 iterations generate the same graph.

d) k-means with random-init



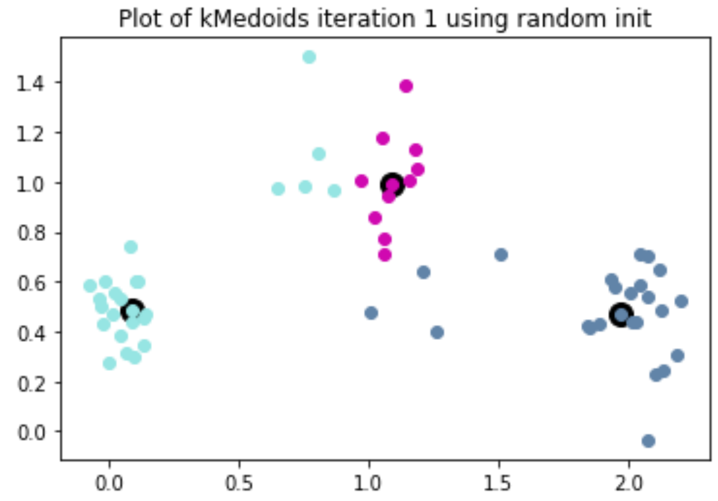


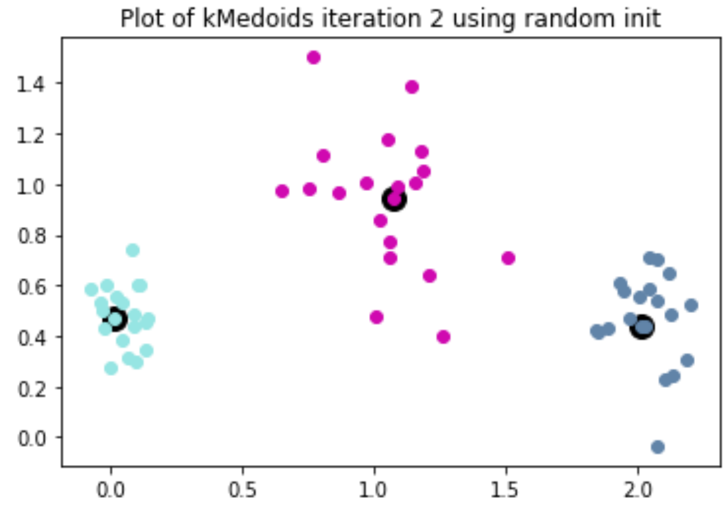


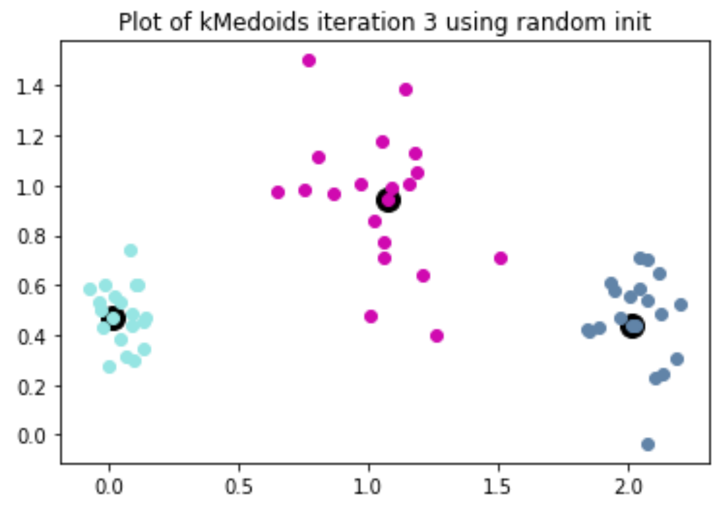


Hence it takes 3 iterations.

e) k-medoids with random-init



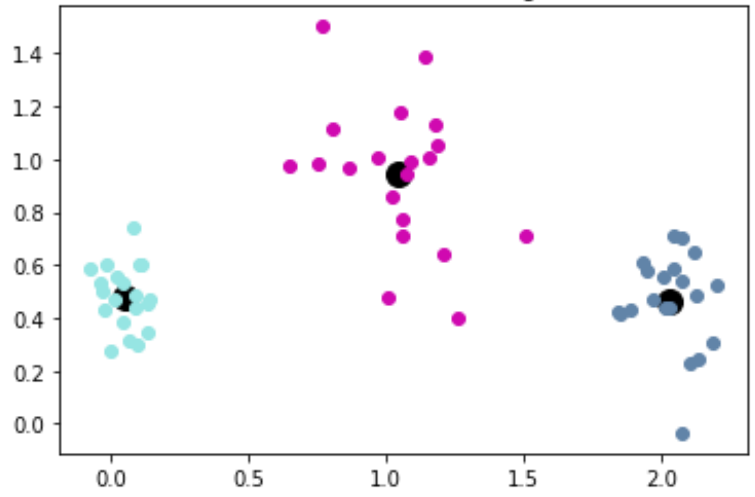




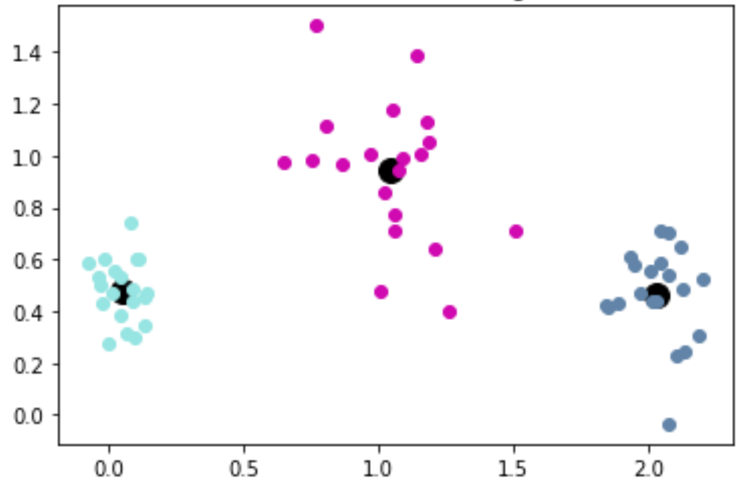
hence it takes 2 iterations.

f) k-means with cheat init

Plot of kMeans iteration 1 using cheat init

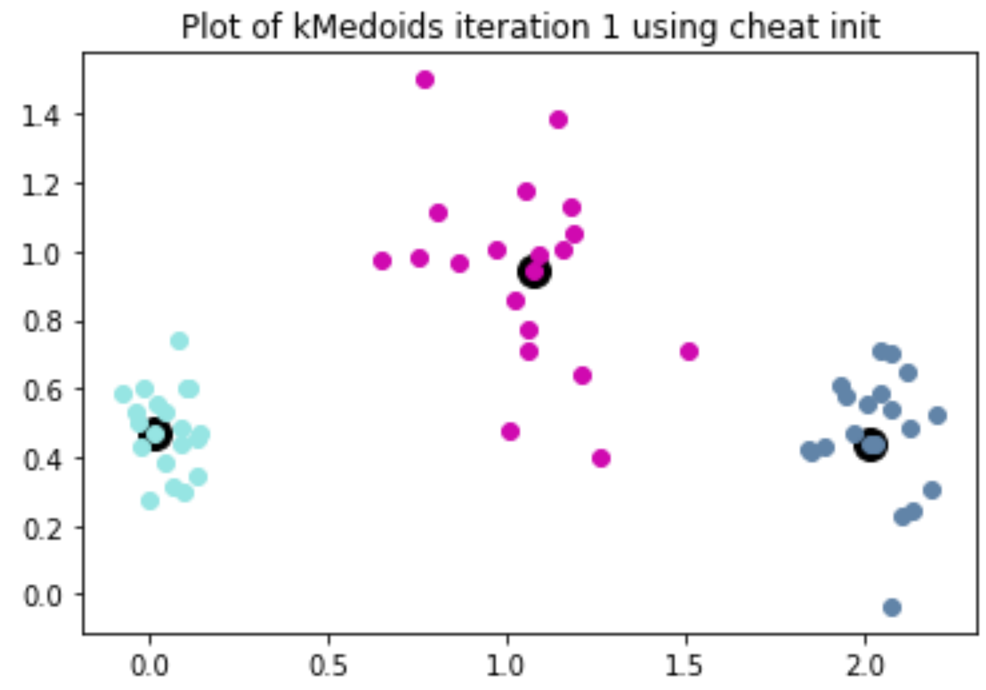


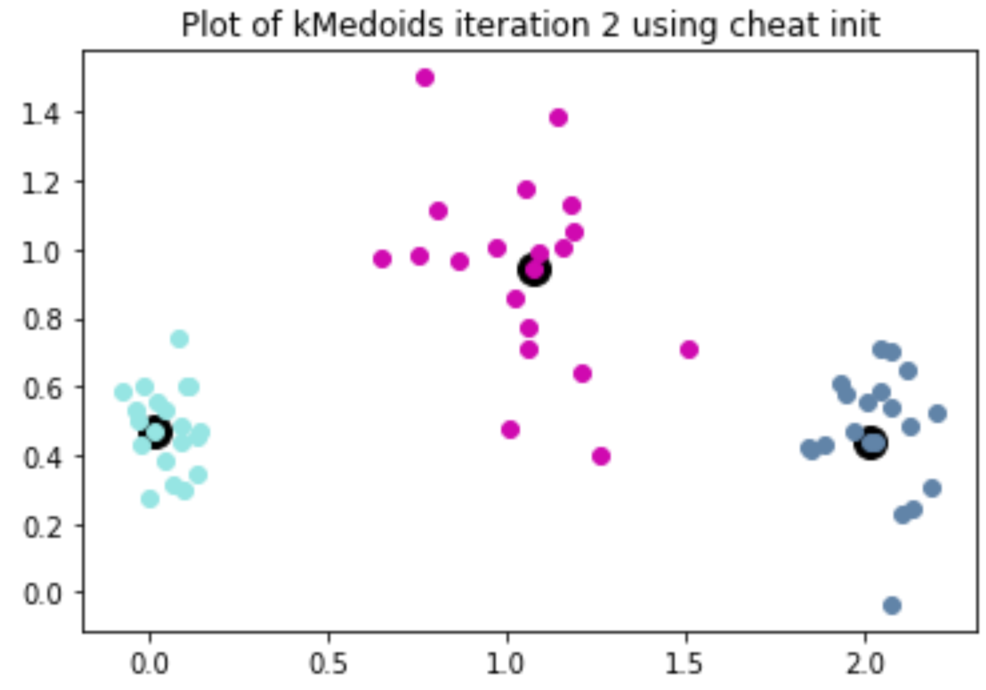
Plot of kMeans iteration 2 using cheat init



hence it takes 1 iteration.

k-medoids with cheat init





hence it takes 1 iteration.

3a)

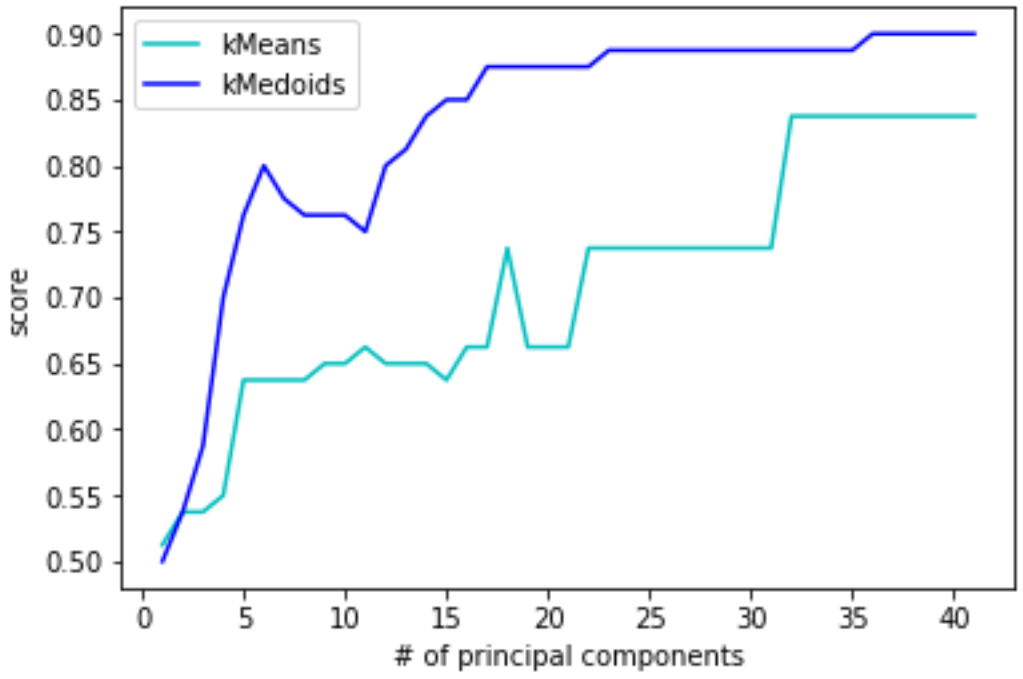
k-means:
average – 0.66125
min – 0.5875

max – 0.775
average time – 0.2106

k-medoids:
average – 0.6175
min – 0.58125
max – 0.625
average time – 0.2131

K-means performs better, it has a higher score for all 3 score metrics and has a slightly lower runtime.

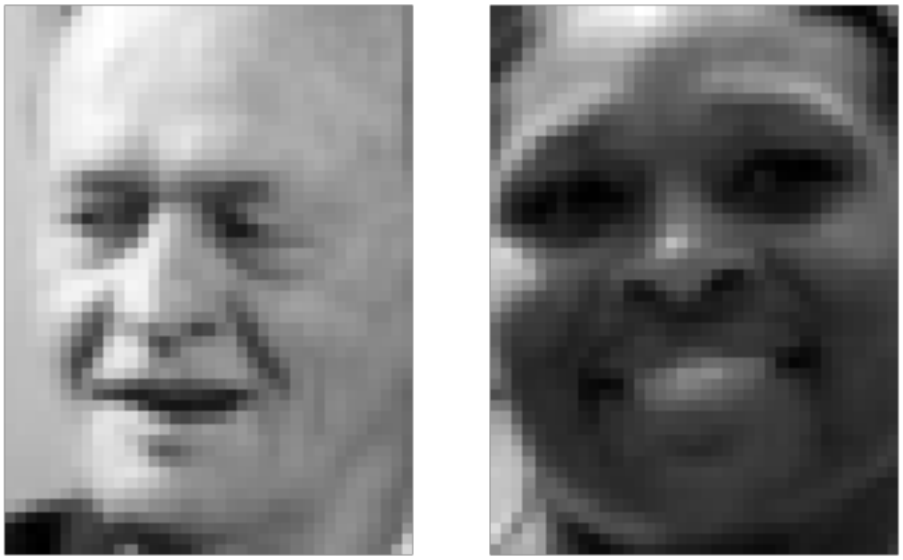
b)



As the number of principal components increases, scores for both algorithms increases. Additionally, kMedoids performs consistently better than kMeans. This conflicts with my earlier findings where kMeans performed better, but makes sense as kMedoids is a better algorithm and should return higher scores as it is more robust to noise since it minimizes pairwise difference instead of minimizing squared error. Increasing the number of principal components increases the score as more features can be used to distinguish the points, however this plateaus when the number is large as beyond that, the additional feature does not contribute significantly to being able to cluster the points.

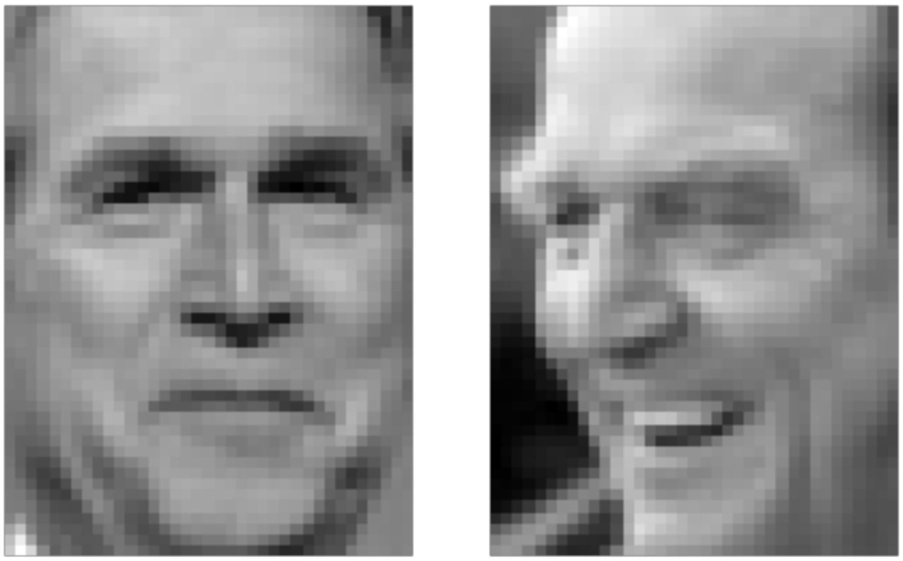
c) I iterated through all possible pairs and computed the best minimum and maximum score, using a model with kMedoids and init='cheat' as it yields the best performance.

The pair that discriminated best was (9,16) with a score of 0.9875:



Both images have distinguishing facial features such as eyes and nose, as well as skin colors that explains the high score.

The pair that discriminated worst was (4,5) with a score of 0.5125 :



In contrast to the previous pair, both images here have similar facial features, such as face wrinkles, and skin color. Hence, they have a low score.