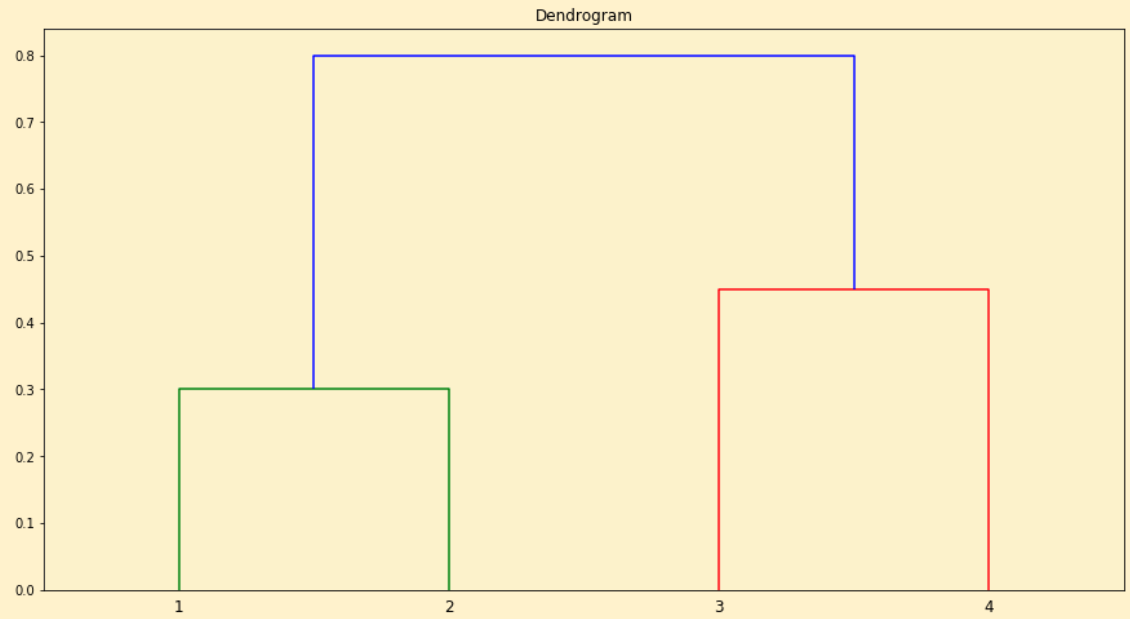
Homework 6

Gavin Monroe - ComS 474

1) Show your calculations. Recall that "single linkage" means we define the distance of two clusters C1 and C2 as the minimum distance of any pair of elements dist(C1, C2) := min a∈C1,b∈C2 dist(a, b) and that "complete linkage" means we define the distance of two clusters C1 and C2 as the maximum distance of any pair of elements dist(C1, C2) := max a∈C1,b∈C2 dist(a, b).

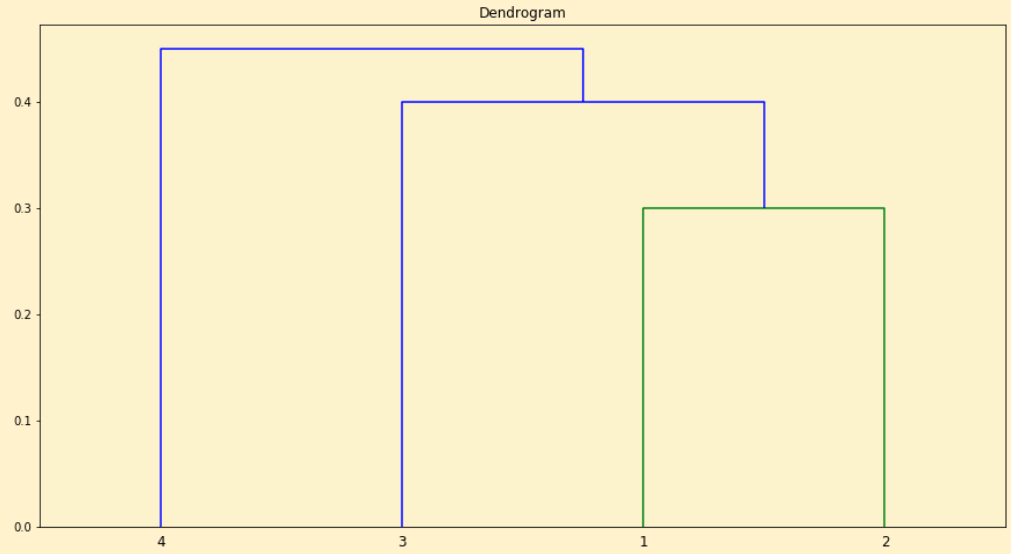
Matrix: [0.3 0.4 0.7 0.3 0.5 0.8 0.4 0.5 0.45 0.7 0.8 0.45]
i = 1; We have to see that 0.3 is the minimum dissimilarity. So we fuse the observations 1 & 2 to from a cluster (1,2) at height 0.3 we know have the new dissimilarity matrix:
[0, 0.5 , 0.8 | 0.5, 0 , 0.45 | 0.8 , 0.95, 0]
When i = 3, we now see that the minimum dissimilarity is 0.45, so then we can fuse observations 3 & 4 to form cluster (3, 4) at height 0.45. We now then produce the new dissimilarity matrix: [0, 0.8 | 0.8, 0]

```
dis_mat = np.array([[0.0, 0.3, 0.4, 0.7], [0.3, 0.0, 0.5, 0.8], [0.4, 0.5, 0.0, 0.45], [0.7, 0.8, 0.45, 0.0]])
dists = squareform(dis_mat)
linkage_matrix = linkage(dists, "complete")
fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111)
dendrogram(linkage_matrix, labels=["1", "2", "3", "4"])
plt.title("Dendrogram")
plt.show()
```



b) Single linkage clustering:

```
linkage_matrix = linkage(dists, "single")
fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111)
dendrogram(linkage_matrix, labels=["1", "2", "3", "4"])
plt.title("Dendrogram")
plt.show()
```



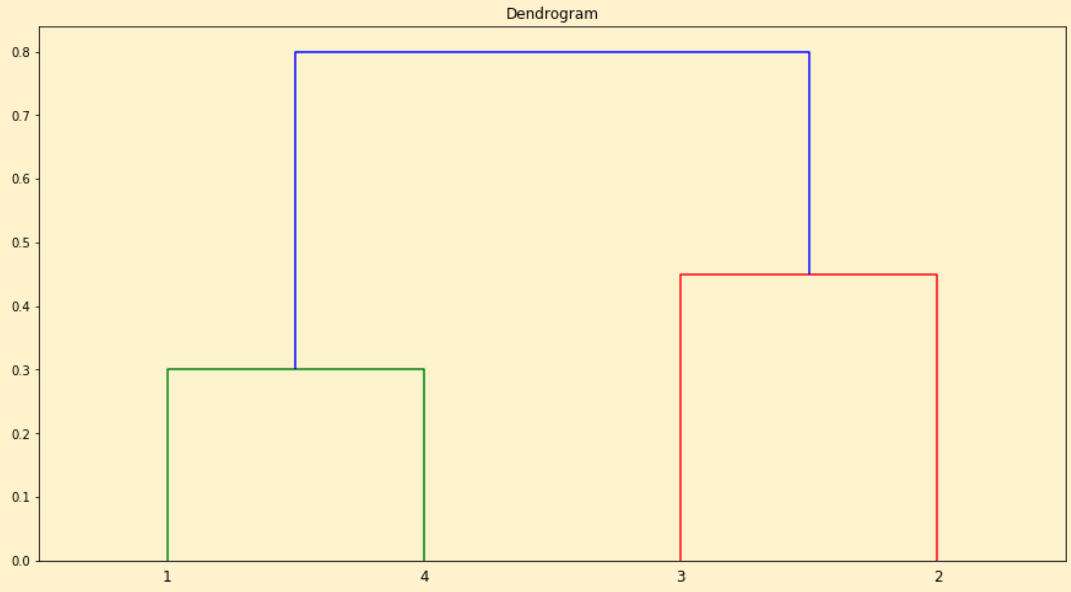
This time using the simple linkage clustering we will again use the algorithm to explain the differences in the steps, that will lead to the den drogram.

Matrix: [0.3 0.4 0.7 0.3 0.5 0.8 0.4 0.5 0.45 0.7 0.8 0.45]

i=4; with the cluster of 1,2, and then i3 = the cluster of ((1,2),3), and then finally the cluster with the i=4 remaining with the form of (((1,2),3), 4)

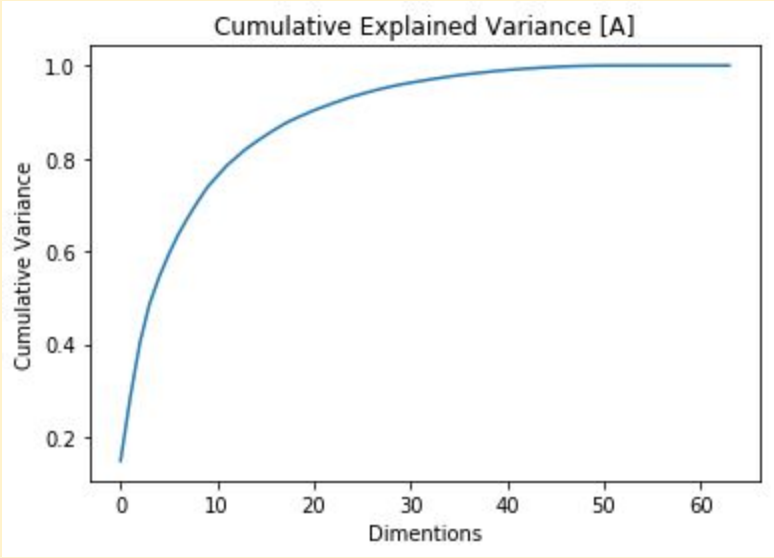
c) Finally we combine the clusters once again for the following problem to get this:

```
dis_mat = np.array([[0.0, 0.3, 0.4, 0.7], [0.3, 0.0, 0.5, 0.8], [0.4, 0.5, 0.0, 0.45], [0.7, 0.8, 0.45, 0.0]])
dists = squareform(dis_mat)
linkage_matrix = linkage(dists, "complete")
fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111)
dendrogram(linkage_matrix, labels=["1", "4", "3", "2"])
plt.title("Dendrogram")
plt.show()
```



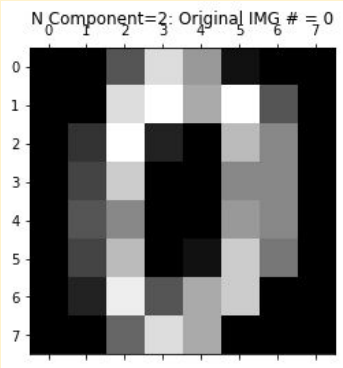
2)

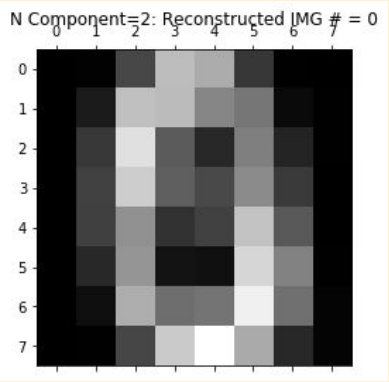
A):

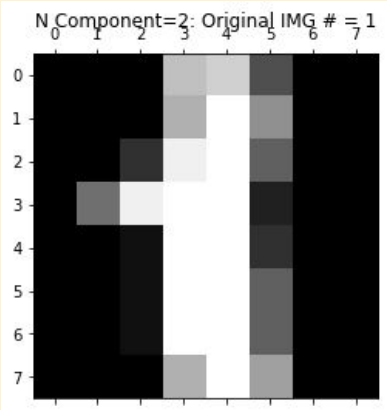


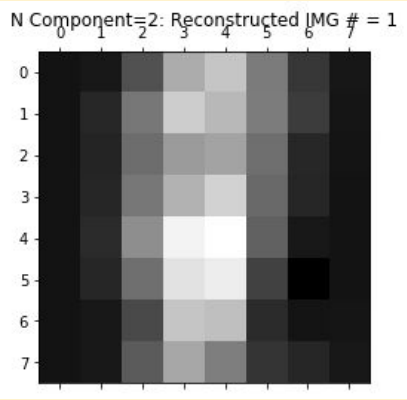
B): 0.2850936482369929

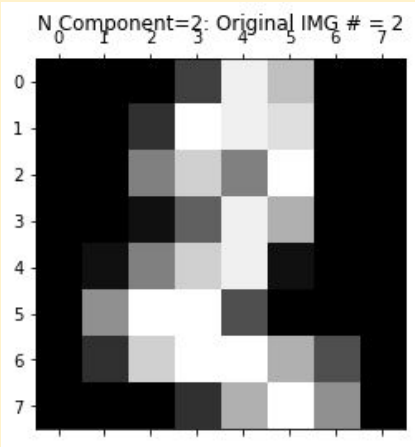
C):

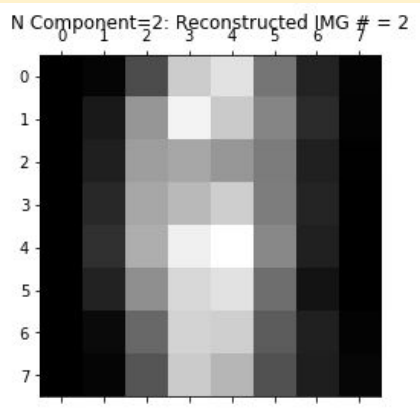


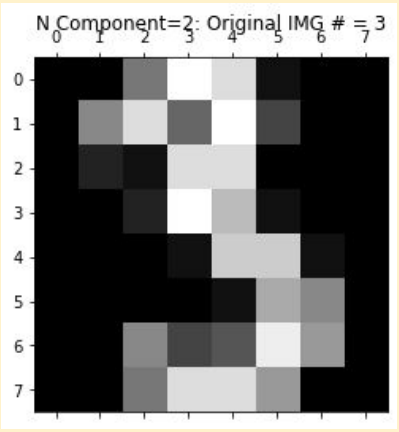


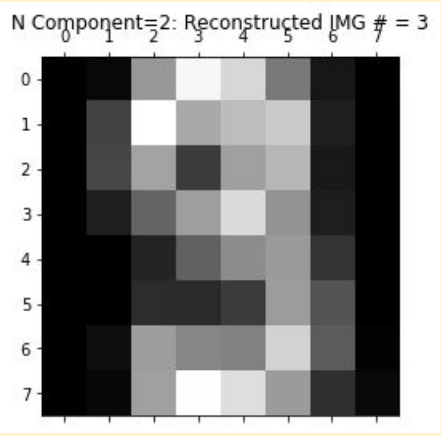


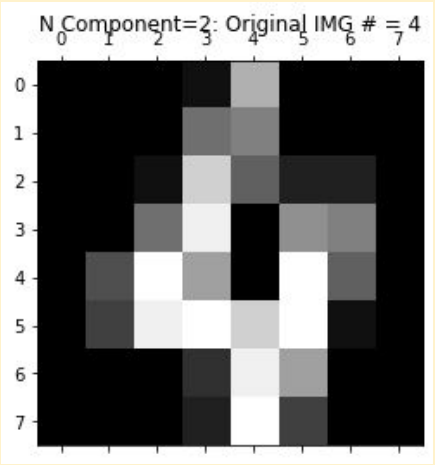


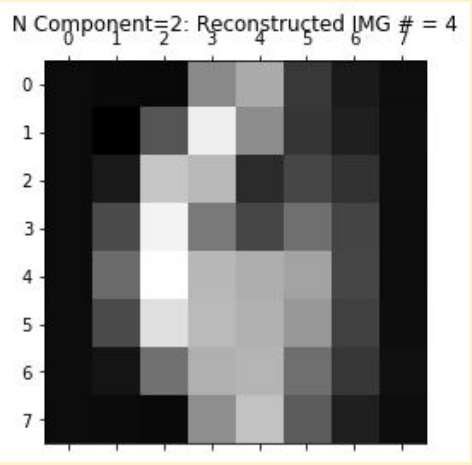






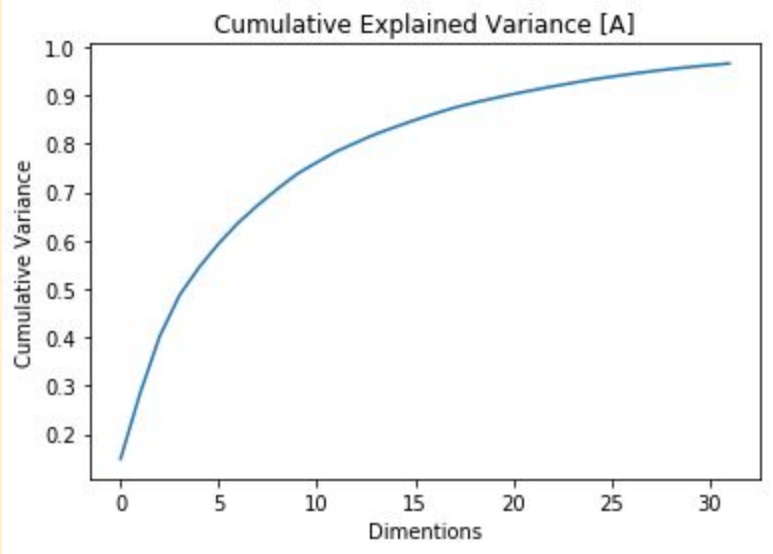






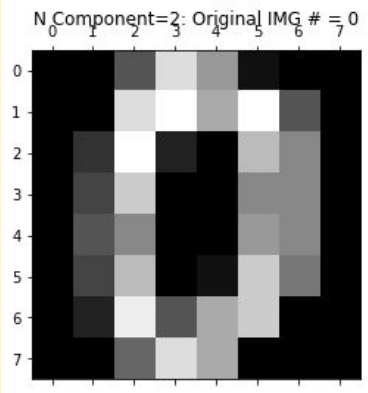
D): As you can see the images to the left(the original) are way less blurry. As for the images on the right you can see they in some way look like they are blurry when compared to their counterparts. This is because of the AI lack of training data for the most part.

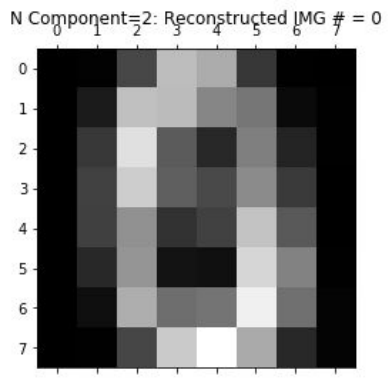
E:)

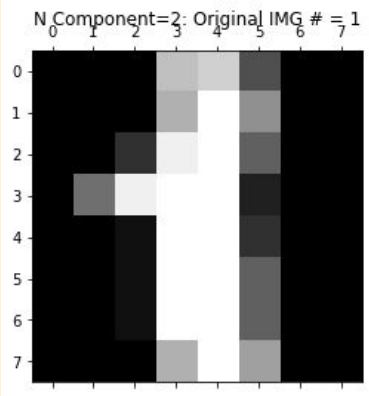


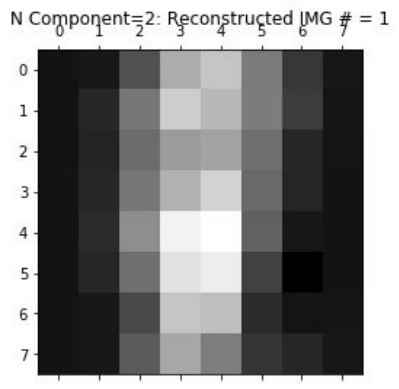
0.9663515990375467

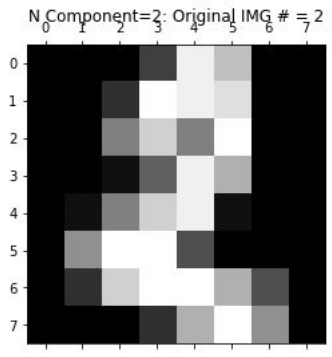
When you up the variance you can see that the machine learning performed a tad bit better in this case. When upping said variance comparing them side by side to the ones in C you can see the differences in both due to that conditional change that we did.

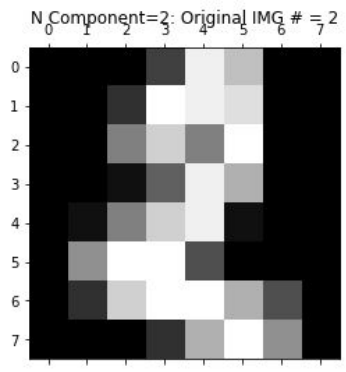


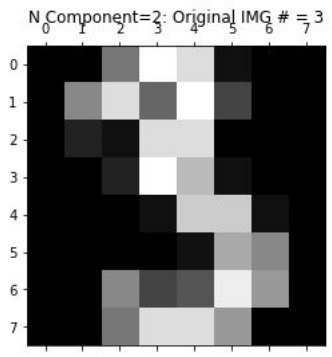


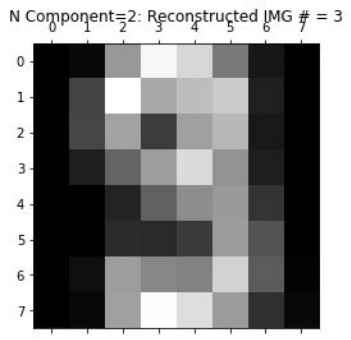


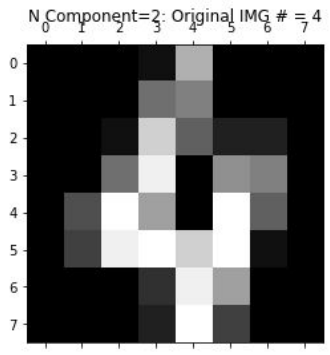


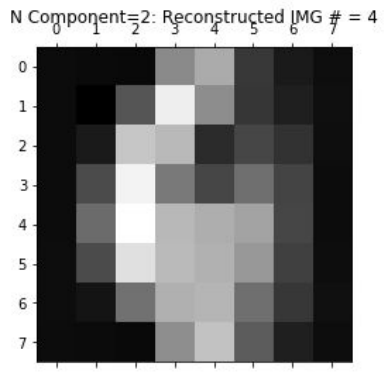

N Component=2: Original IMG # = 2


N Component=2: Original IMG # = 2


N Component=2: Original IMG # = 3

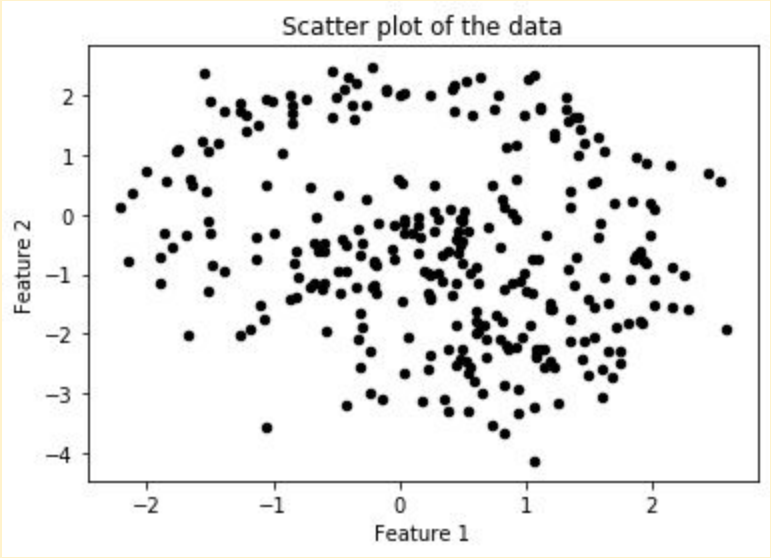

N Component=2: Reconstructed IMG # = 3


N Component=2: Original IMG # = 4

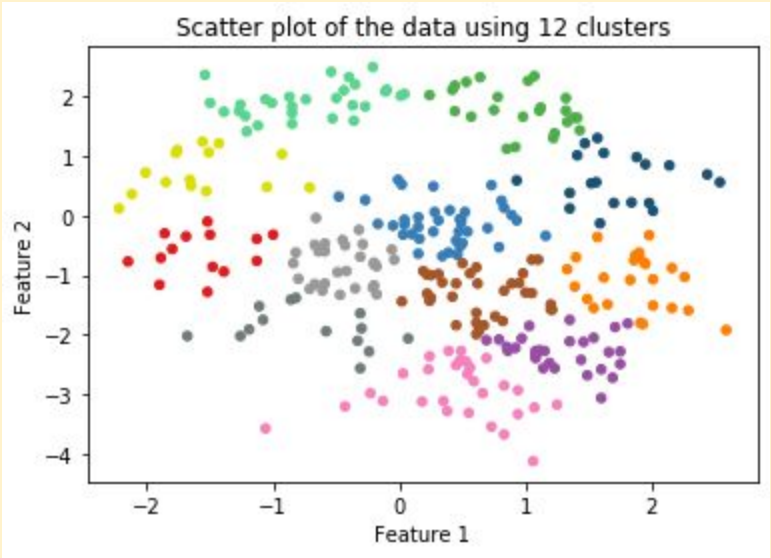

N Component=2: Reconstructed IMG # = 4

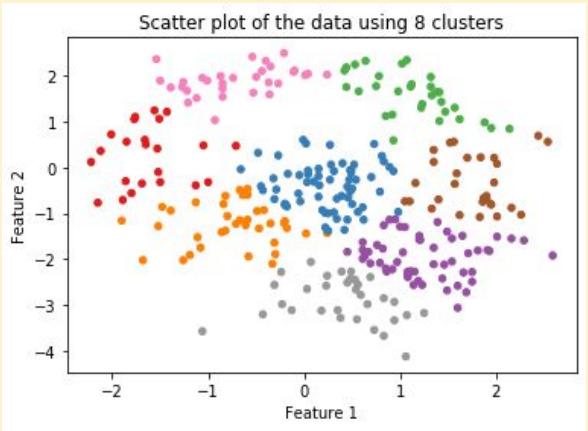
3)

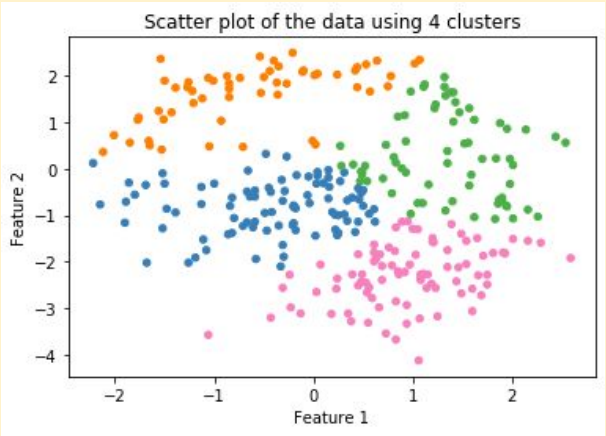
A):


Scatter plot of the data
Feature 2
Feature 1

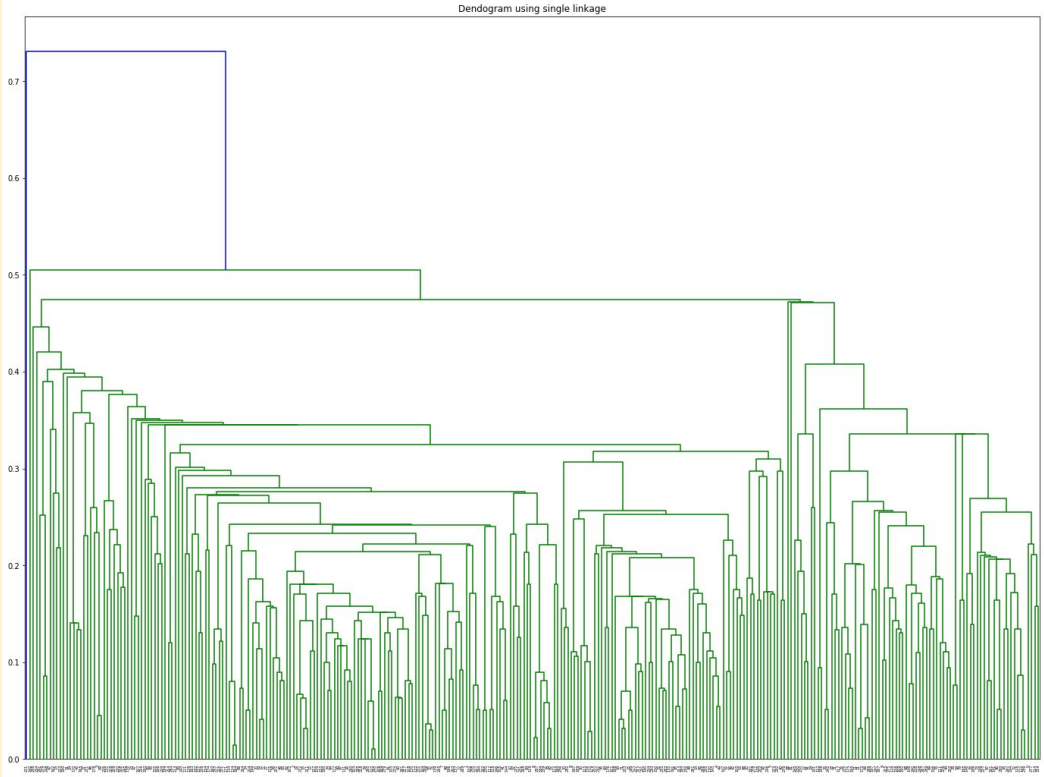
B):


Scatter plot of the data using 12 clusters
Feature 2
Feature 1

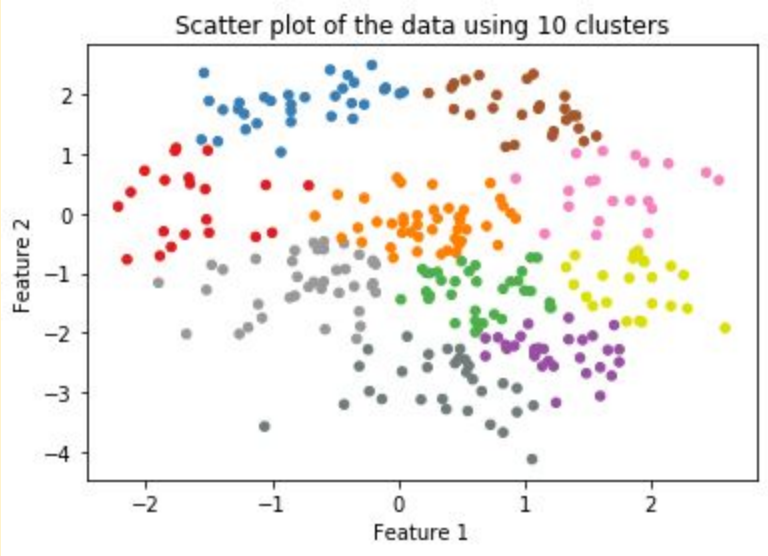

Scatter plot of the data using 8 clusters
Feature 2
Feature 1


Scatter plot of the data using 4 clusters
Feature 2
Feature 1

C):

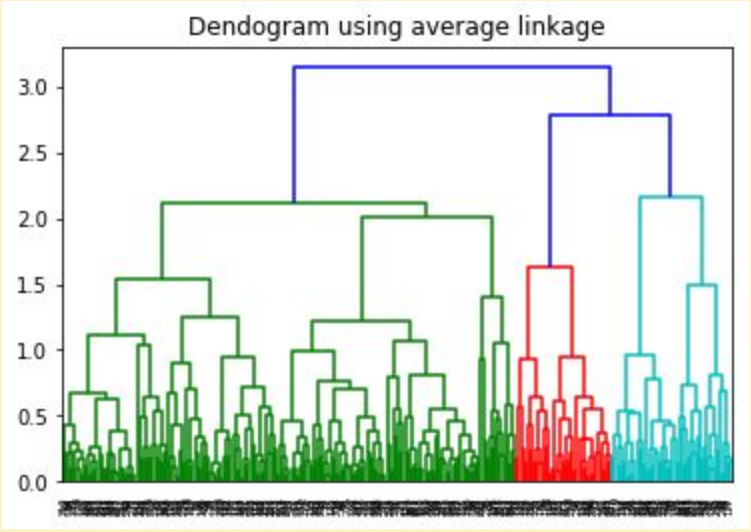


D:)



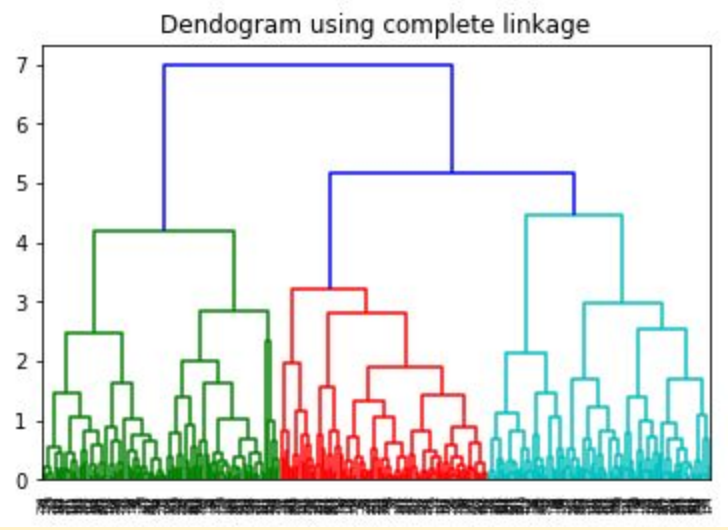
You can see that 0.5, 0.45, and 0.23 is a major distance in the gap here.

E):



The gap here can be see at the 2.3, 2.0, and 3.1

F):



You can see the gaps here at 5, 4, and 7.

G):Complete-interface grouping

The most pessimistic scenario time multifaceted nature of complete-interface grouping is all things considered O(n^2 log n). One O(n^2 log n) calculation is to process the n^2 separation metric and afterward sort the separations for every datum point (by and large time: O(n^2 log n)).
Single-connect grouping
The time unpredictability of single-connect grouping is O(n^2). We initially figure all separations in O(n^2).
IN this context I would Have to pick k- means due to its performance over the read of them.