**Question 1**

**What we did:**

In this question, we used the Olivetti dataset like we did in the last assignment (Assignment 3). We then did the train-validation-test split, and then used PCA to reduce the dimensionality of the training set while preserving 99% of the variance. The training set was then reduced from 4096 features to 222 features. We then determined the most suitable covariance type for the training set. I set the number of components to 35, and I found that the “diag” type is best using both AIC and BIC. After that, I used the “diag” type to determine the minimum number of clusters with both AIC and BIC. The best number of clusters by AIC was 3, and it was 2 by BIC. I chose 2 for the steps after because I want the minimum number of clusters. We then plotted the results of PCA, results of finding the best covariance type, and results of finding the minimum number of clusters. Right after that, we output the hard clustering assignments and soft clustering probabilities of each instance in the reduced training set. Then, we used the model to generate 5 new faces and visualized them. We then modified the first 5 images of the training set by rotating them with random degrees, flipping them horizontally, and darkening them. We then used the score\_samples() method to check if the model is able to detect the anomalies (modified images).

**Images and/or Discussion:**

Step 3:

After applying PCA with the “n\_components” parameter set to 0.99 to preserve 99% of the variance, we get the reduced dataset. This reduced dataset has 222 features compared to the original 4096.

Step 4:

For this, I set the “n\_components” of GaussianMixture to be 20. I originally tried to use 40 because I was thinking that there should be ideally 40 clusters since there are 40 unique people. However, I got this error:



Therefore, I decided to decrease the number of clusters instead of increasing reg\_covar so that I potentially will have less clusters with very few points.

I then tried the 4 covariance types: "spherical", "diag", "tied", and "full". And here is the AIC and BIC result:

A white background with black numbers and letters

Description automatically generated

As we can see, by both criteria, “diag” seems to be the best covariance type because it has the lowest scores.

Step 5:

In this step, we determined the minimum number of clusters. I set the covariance type to “diag” because of the previous step and set the range to try as 1 to 20 to include the number of components I used in the previous step. Here is the result of both AIC and BIC:

A screenshot of a computer

Description automatically generated

As we can see, the best # of clusters by BIC are 2 while by AIC they are 3. This makes sense because as stated in the course modules, BIC prefers simpler models with fewer parameters compared to AIC.

I chose the # of clusters for the next steps to be 2, determined by BIC, because I want the minimum number of clusters.

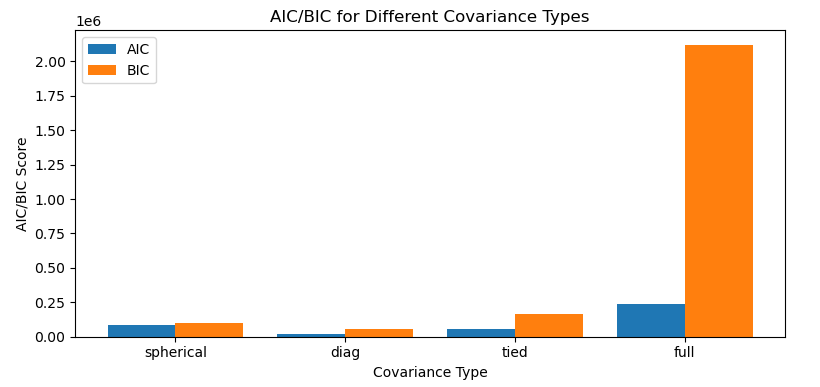
Step 6:

In this step, we plotted results from step 3, 4, and 5.

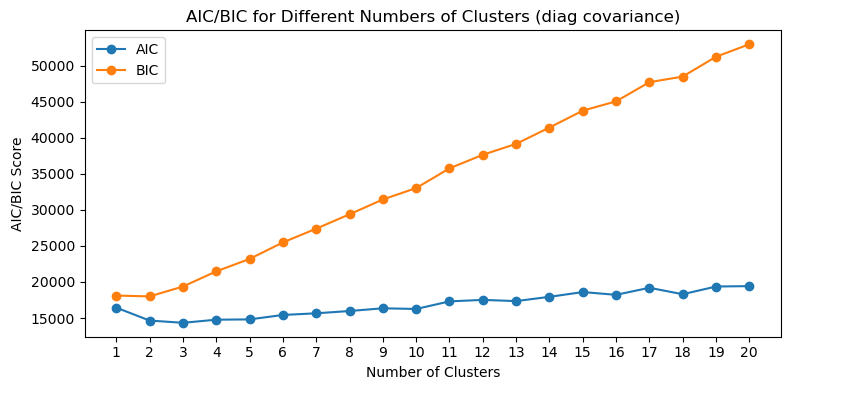
A graph with a curve

Description automatically generated

The image above is the result of step 3, which is the result of applying PCA to the training set. As we can see, to preserve 99% of the variance, there needs to be 222 features.

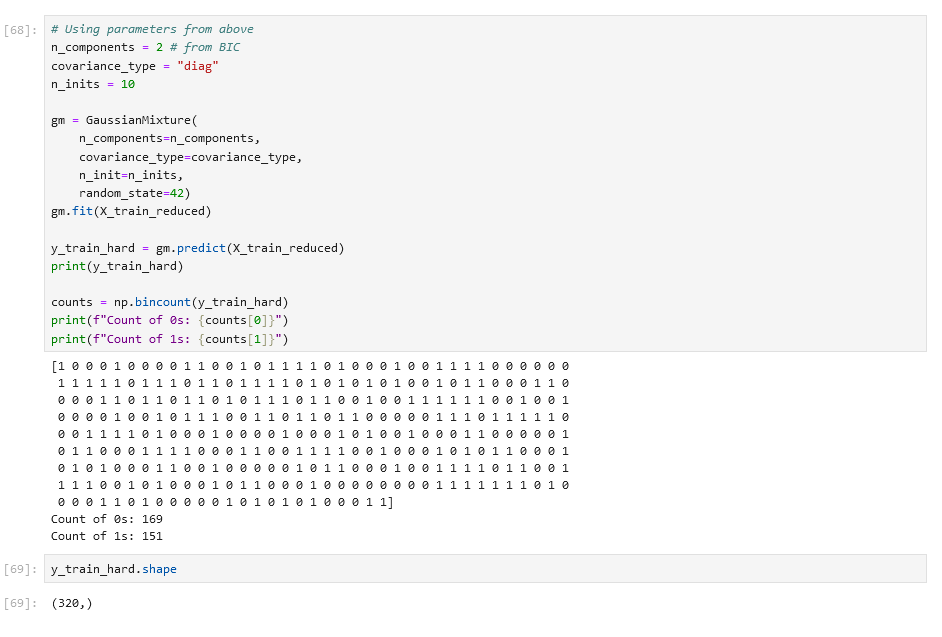


The image above is the result of step 4, which is when we’re trying to determine which covariance type we should use. We can see that both AIC and BIC are lowest when covariance type is “diag”.



The image above is the result from step 5, in which we were trying to determine the minimum number of clusters using either AIC or BIC. From the graph, we can see that for AIC, the best # of clusters are 3, and for BIC, they are 2.

Step 7:



For this step, we used the best parameters that we determined in previous steps, which is to use 2 components and use the “diag” covariance type. We then printed out the result of .predict(). And we can count the number of points that belongs to each of the 2 clusters. We can see that 169 points belong to cluster 0, and 151 points belong to cluster 1.

Step 8:

A screenshot of a computer

Description automatically generated

In this step, we printed out the probabilities of each instance, and it matches the results from step 7. Here, we can see that the first instance has a basically 100% likelihood of being in cluster 1, which matches that the first element in the list in step 7 is indeed 1.

Step 9:

A screenshot of a computer screen

Description automatically generated

In this step, we generated 5 new images and labels. We also visualized them. As we can see it’s kind of like a lot of faces being merged, which is quite creepy.

Step 10:

A collage of a person's face

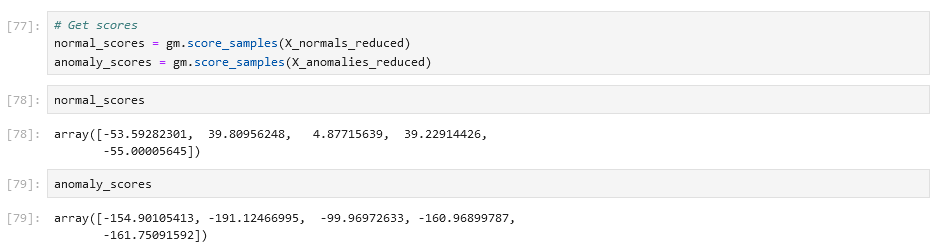
Description automatically generated

In this step, I modified the first 5 images in the training set. Here are the steps of modification:

1. Rotate image with a random angle between -180 and 180
2. Flip image horizontally
3. Darken the image

Step 11:

In this step we use the modified image from the previous step and see if the model is able to detect anomalies (the modified images).



The image above shows the result of unmodified and modified images. Judging from the numbers, I believe the model is able to detect anomalies. This is because the scores are really far apart. For example, for the second image, the normal image’s score is ~39.8 while the modified image’s score is ~-191.1.

**Lesson learned:**

* I learned that for the “n\_components” parameter in PCA, if we put a number between 0 and 1, it will become the variance percentage to preserve rather than the actual number of components.
* For GaussianMixture, I learned more about the 4 covariance types, and how a large number of components can stretch the shapes of the clusters and result in “ill-defined empirical covariance”, which is the error I got.
* For GaussianMixture, I also learned that the “reg\_covar” parameter defines the minimum size of the “bubbles” around the clusters. So, if I want to use a high number of components, I also have to increase this parameter so that the bubbles/boundaries don’t become too small.
* I learned more about how GaussianMixture can help us detect anomalies with the .score\_samples() method.