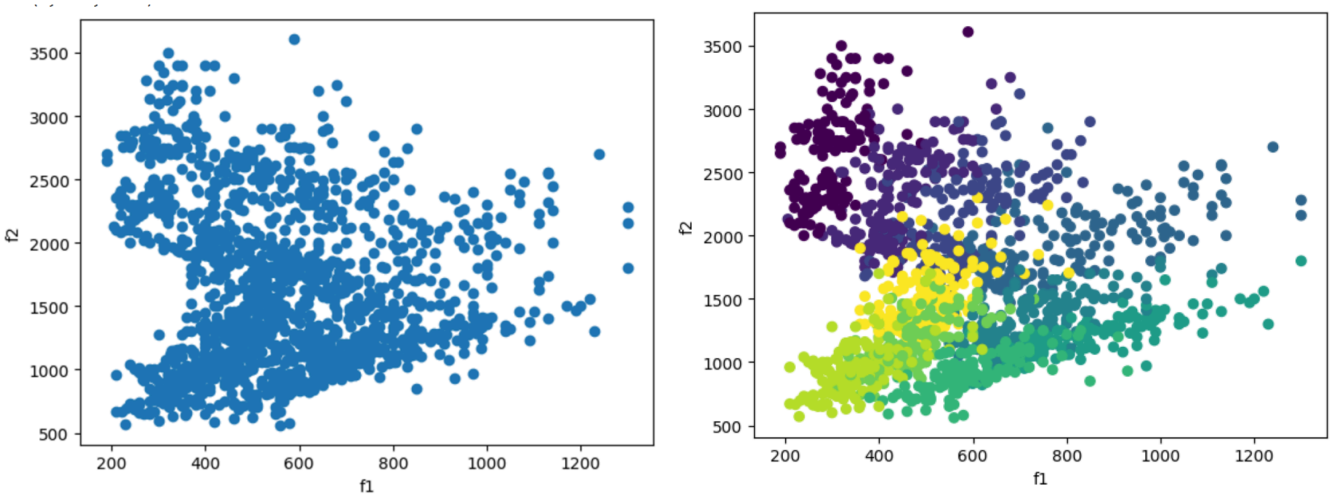
Name: Kiana Hadysadegh
Student ID: 230632224
Assignment Number 3(Density Estimation)
Module number: ECS708P

Q1. Produce a plot of F1 against F2. (You should be able to spot some clusters already in this scatter plot.). Comment on the figure and the visible clusters [2 marks]

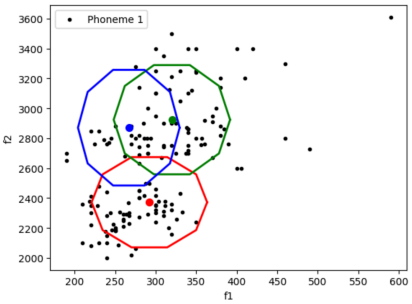


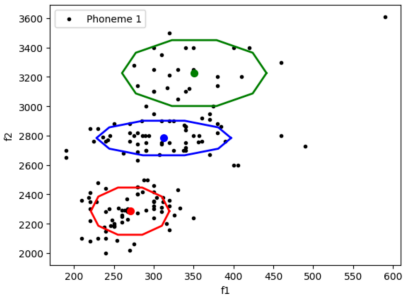
Clusters are dense regions of points in the plot, which means groups of points that are closer to each other in comparison with other areas in the plot. In the left figure, we have the plot of our data points without specified classes, and in my opinion, we can see at least 3 distinct clusters in the figure. In the right figure, we plot the data points and separate them based on phoneme_ids which show points of each class with different colors.

Q2. Run the code multiple times for K=3, what do you observe? Use figures and the printed MoG parameters to support your arguments [2 mark]

Since in the EM algorithm we randomly pick centroids from the data in the start state we can see that the start figure of each run is different. Also, since the start state is different and the EM algorithm is initialized by random centroids the algorithm can stuck in the local optima so the ending figure is different sometimes. This is because the EM algorithm is dependent on the initial parameter values.

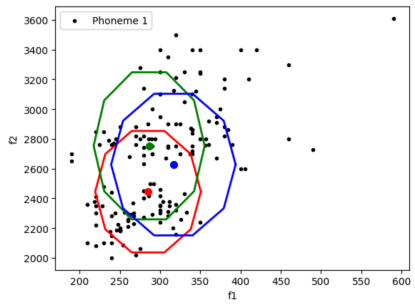
Start figure - End figure - Parameters
First run:

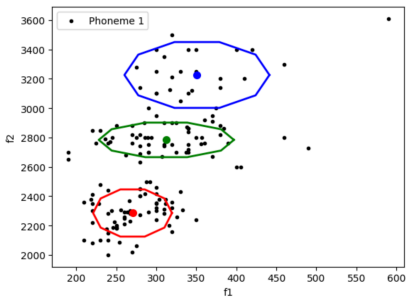





```
[[ 270.3952  2285.4653 ]
 [ 350.8445  3226.3367 ]
 [ 312.59113 2783.8975 ]] [[[ 1213.7384145      0.        ]
  [    0.         14278.42036085]]

 [[ 4102.86137306     0.        ]
  [    0.         27830.24656762]]

 [[ 3562.59931538     0.        ]
  [    0.          7657.78029577]]]
```

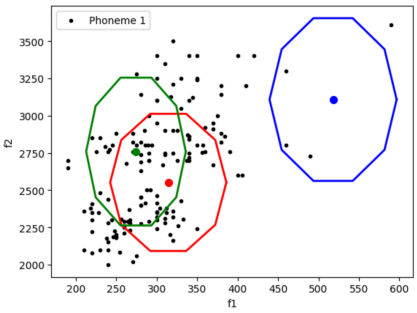
Second run:

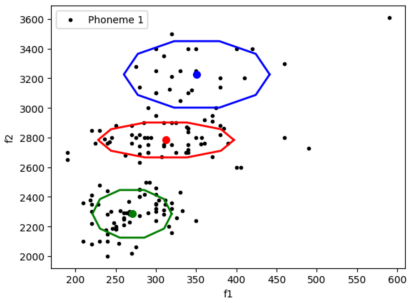





```
[[ 270.3952  2285.4653 ]
 [ 312.59125 2783.898  ]
 [ 350.8446  3226.3394 ]] [[[ 1213.7384348      0.         ]
  [    0.         14278.42035434]]

 [[ 3562.59744324     0.         ]
  [    0.          7657.84861601]]

 [[ 4102.875312       0.         ]
  [    0.         27829.54597241]]]
```

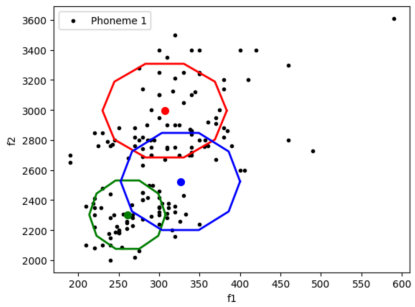
Third run:

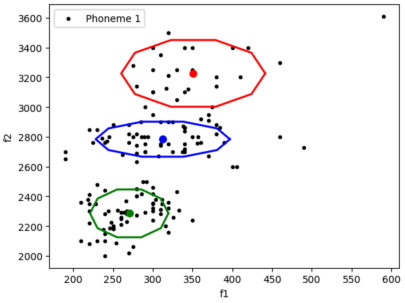





```
[[ 312.59128 2783.898  ]
 [ 270.3952  2285.4653 ]
 [ 350.84464 3226.3398 ]] [[[ 3562.59733336     0.         ]
  [    0.          7657.85758069]]

 [[ 1213.73843866     0.         ]
  [    0.         14278.4196138 ]]

 [[ 4102.87704745     0.         ]
  [    0.         27829.44268308]]]
```

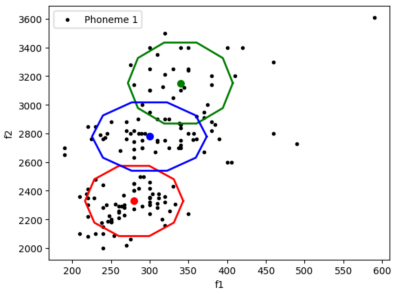
Fourth run:

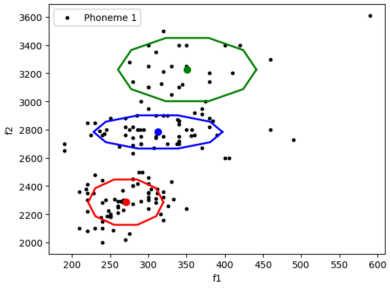





```
[[ 350.84457 3226.338  ]
 [ 270.3952  2285.4653 ]
 [ 312.5912  2783.8977 ]] [[[ 4102.86848113     0.         ]
  [    0.         27829.89779745]]

 [[ 1213.73842322     0.         ]
  [    0.         14278.42115515]]

 [[ 3562.59828559     0.         ]
  [    0.          7657.81508733]]]
```

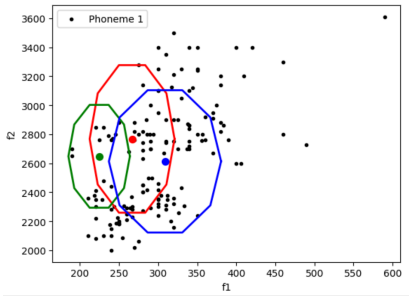
Fifth run:

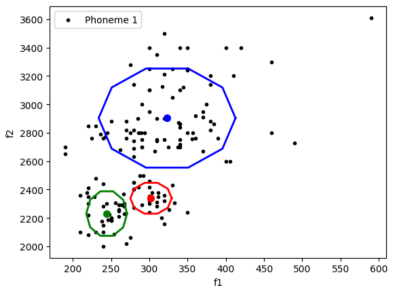





```
[[ 270.3952  2285.4653 ]
 [ 350.8446  3226.3394 ]
 [ 312.59125 2783.898  ]] [[[ 1213.73843494     0.         ]
  [    0.         14278.4202995 ]]

 [[ 4102.875375       0.         ]
  [    0.         27829.54221422]]

 [[ 3562.59743765     0.         ]
  [    0.          7657.84897245]]]
```

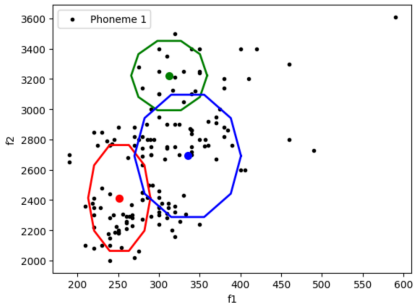
Sixth Run:

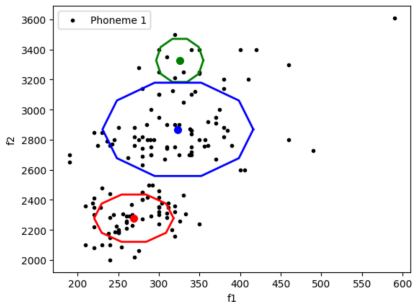





```
Finished.

[[ 302.26315 2338.855  ]
 [ 244.30171 2231.018  ]
 [ 323.24017 2903.5637 ]] [[[  365.77190917     0.        ]
  [    0.          6568.98351071]]

 [[  356.12422529     0.        ]
  [    0.         13577.9242552 ]]

 [[ 4002.5369687      0.        ]
  [    0.         67340.65021549]]]
```

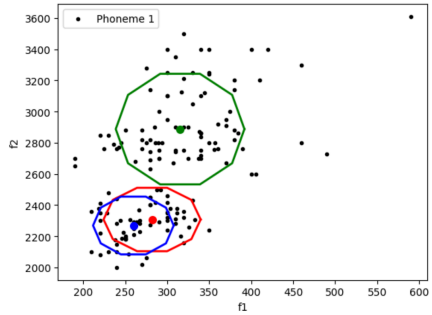
Seventh run:

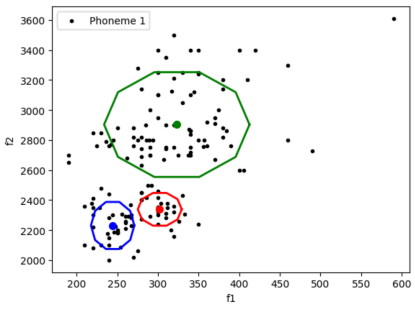





```
Finished.

[[ 269.23795 2278.1    ]
 [ 325.6923  3327.6377 ]
 [ 323.42883 2869.0532 ]] [[[ 1195.80769911     0.        ]
  [    0.         13681.55363036]]

 [[  421.28182807     0.        ]
  [    0.         11380.78396347]]

 [[ 4305.86895774     0.        ]
  [    0.         53222.38544626]]]
```

Eighth run:






```
Finished.

[[ 302.26315 2338.855  ]
 [ 323.24017 2903.5637 ]
 [ 244.30171 2231.018  ]] [[[  365.77190917     0.        ]
  [    0.          6568.98351071]]

 [[ 4002.5369687      0.        ]
  [    0.         67340.65021549]]

 [[  356.12422529     0.        ]
  [    0.         13577.9242552 ]]]
```

Q5. Use the 2 MoGs (K=3) learnt in tasks 2 & 3 to build a classifier to discriminate between phonemes 1 and 2, and explain the process in the report [4 marks]

We first reload our trained models from saved parameters (mean, covariance, and weights for phoneme1 and phoneme2 trained models). For each phoneme model:

1- we create a Z matrix (shape NxK) to store its predictions

2- get model predictions using the get_predicitons function and stores it in Z

3- We sum the elements in the matrix over the second axis (K axis) for maximum likelihood estimation. This means for each sample xi, we sum predictions of the sample for each GMM (we have K GMMS).This new array is our prediction for that phoneme.

4- To classify the data we compare predictions (= Z matrix) of phoneme_1 and phoneme_2 for each sample xi. The sample xi belongs to the phoneme which has a higher prediction value.

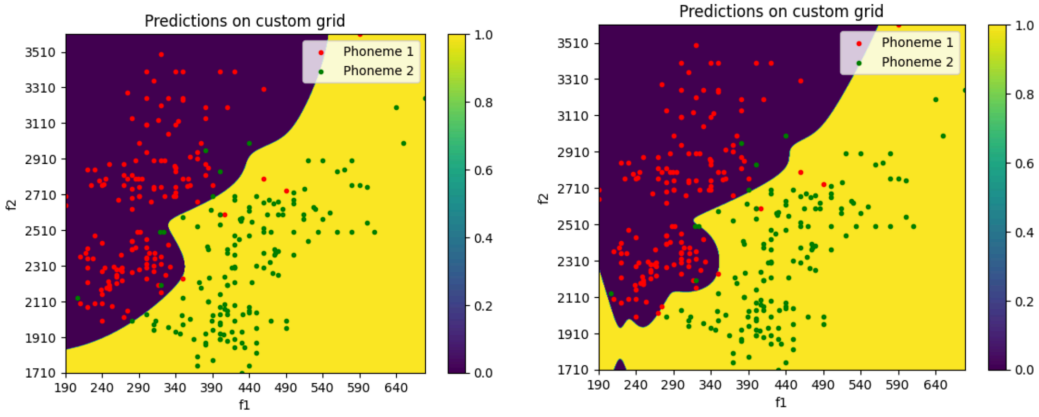
5. To do this we stack the predictions of two phonemes vertically and apply argmax. So if the prediction of phoneme 1 is higher than phoneme to for sample xi, we store 0 in index i of the classification result array, otherwise, we store 1.

6. Because we only use phoneme ids 1 and 2 we should remove other classes from the phoneme_id array (which has our ground truth labels) to match our classification data in terms of size.
7. Since we want to use phonem_id array as ground truth labels, we add 1 to the classification result array to match with labels (datapoint which belongs to phoneme1 with 1 as prediction value and datapoint belongs to phoneme2 with 2 as prediction value)
8. To calculate miss classification error we calculate how many times our prediction was wrong and divide it by the total number of predictions
9. To calculate accuracy we calculate how many times our prediction was correct and divide it by the total number of predictions

Q6. Repeat for K=6 and compare the results in terms of accuracy. [2 mark]
K = 3 -> Accuracy = 0.9506578947368421
K = 6 -> Accuracy = 0.9605263157894737
We observe that we have almost the same Accuracy for both Ks. This can occur because of the following reasons:
1- our data naturally does not have more than 3 distinct clusters
Or
2- adding more clusters does not capture meaningful variations from the data.
Or
3- Overfitting can be another reason when our model fits the noises instead of the patterns of the data. So, by adding more clusters it has the same performance as before.
Or
4- since the EM algorithm is initialized by random centroids, it can stuck in a local optima. So both models can converge into the same local and have the same accuracies.

Q7. Display a "classification matrix" assigning labels to a grid of all combinations of the F1 and F2 features for the K=3 classifiers from above. Next, repeat this step for K=6 and compare the two. [3 marks]

Left figure K=3, Right figure K=6



We can observe although the boundary changes a little bit, almost no changes are observed in the class that the data points belong to. This is because changing k from 3 to 6 doesn't affect our accuracy so much, as we have seen the same situation in the previous question. So, It can show that maybe:

1- our data naturally does not have more than 3 distinct clusters or 2- adding more clusters does not capture meaningful variations from the data. 3- Overfitting can be another reason when our model fits the noises instead of the patterns of the data. So, by adding more clusters it has the same performance as before. 4- since the EM algorithm is initialized by random centroids, it can stuck in a local optima. So both models can converge into the same local and have the same accuracies.

Q8. Try to fit a MoG model to the new data. What is the problem that you observe? Explain why it occurs [2 marks]

Problem:
The array contains Nans and infinity values because of the singularity problem.
In the case that the data dimensionality is higher than the data samples the covariance matrix is singular.
Non invertible, zero determinant.

Explanation: Covariance is a symmetric matrix. The dimensionality of the data means the number of features and when it is higher than the number of samples the covariance matrix becomes singular. So, its determinant is zero and it is not invertible (because in the calculation process division by zero will happen).

Q9. Suggest ways of overcoming the singularity problem and implement one of them. Show any training outputs in the report and discuss. [3 marks]
1. Put constraints in the form of the covariance matrix:

$$\Sigma = \sigma I \quad \sigma^2 = \frac{1}{N}\sum_n (x_n - \hat{\mu})^T (x_n - \hat{\mu})$$

In this method, 1- we calculate the variance of our data using the above formula 2-repeat its value on the diagonals of the identity matrix, and 3- consider the resulting matrix as our covariance matrix.

2. Put constraints in the form of the covariance matrix:

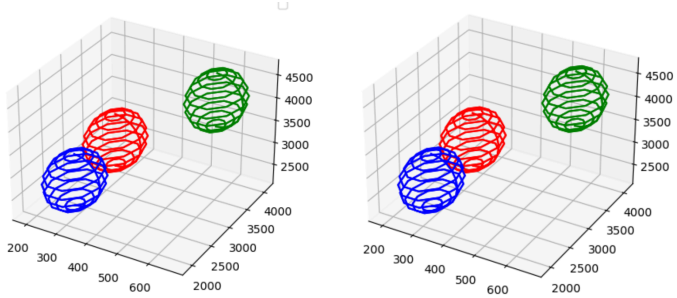
$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & \sigma_d \end{bmatrix} I \quad \sigma_j{}^2 = \frac{1}{N}\sum_n \left(x_n(j) - \hat{\mu}(j)\right)^2$$

In this method, 1- we calculate the variance using the above formula for each d feature of our data matrix. So in the end we have d variance values for d features of data 2- we put each variance_i (variance of feature i values) on the diagonals of the identity matrix, and 3- consider the resulting matrix as our covariance matrix.

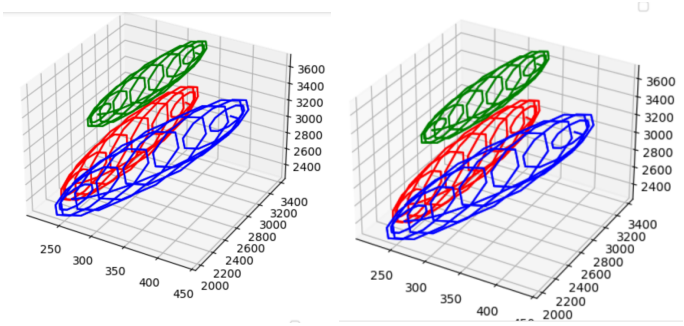
3. Regularise the covariance matrix by adding a diagonal matrix:

$$\Sigma_0 = \sigma_0{}^2 I \quad \hat{\Sigma} = \frac{1}{N}\sum_n (x_n - \hat{\mu})(x_n - \hat{\mu})^T + \sigma_0{}^2 I$$

In this method, We add a regularisation term ($\sigma_0{}^2$) to values to the elements on the diagonal of the covariance matrix and use the new matrix as covariance.
We pick the second method to fix the errors:
Outputs after using the second method:



Output before (with Nans and Infinity values in the Z prediction matrix - this result is generated from Q8 with the condition that we break the loop when we see Nans and infinity values in the Z prediction matrix):



We can observe that by using the second method our clusters become more spherical rather than oval. Also by using this method our covariance matrix is square and there is no more singularity and zero determinant problem.

```
Covariance matrix:
 [[[  3555.33789062      0.              0.        ]
  [     0.          139972.296875        0.        ]
  [     0.              0.          167592.6875    ]]

 [[  3555.33789062      0.              0.        ]
  [     0.          139972.296875        0.        ]
  [     0.              0.          167592.6875    ]]

 [[  3555.33789062      0.              0.        ]
  [     0.          139972.296875        0.        ]
  [     0.              0.          167592.6875    ]]]

Mean:
 [[ 337.32358 2962.824   3300.1475 ]
 [ 587.4282  3603.9956  4191.424  ]
 [ 272.85916 2418.2932  2691.1523 ]]
```