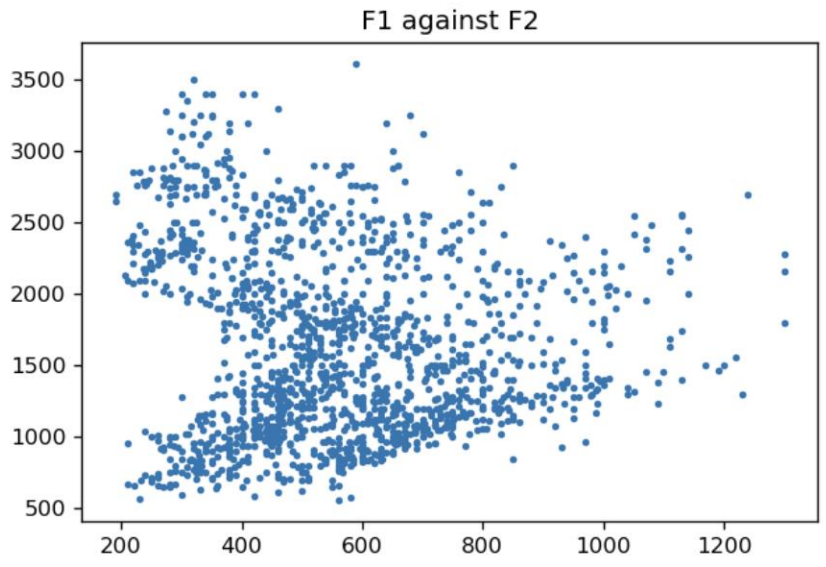
# ML Assignment 2

Unsupervised Learning

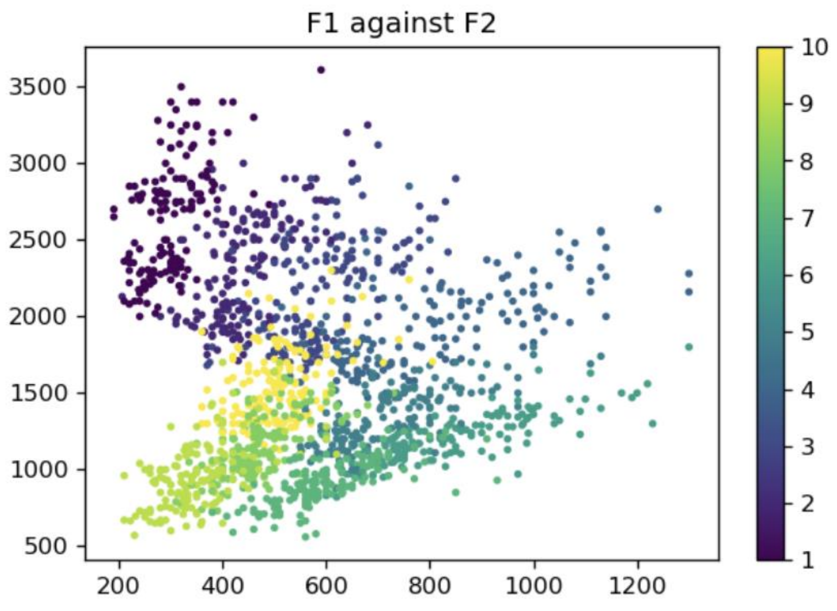
Gargeya Sharma – 220278025 – MSc Artificial Intelligence

## Ques 1:



**Observation:**

After plotting the scatter plot, we can see that the points are clustered at specific positions more densely than others. These rough clustered areas can be thought as the region where there is high correlation between the features (F1) and (F2). The figure also shows that with increasing value of the features, the points are more scattered (Density of points in decreasing). From the graph shown below we can prove the analysis above that for different phenome ids, we have different clusters.

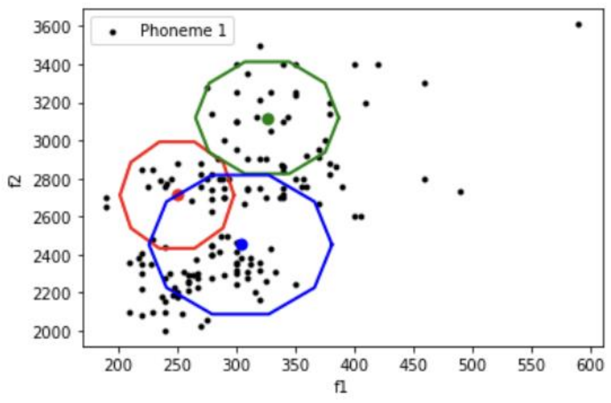


Ques 2:

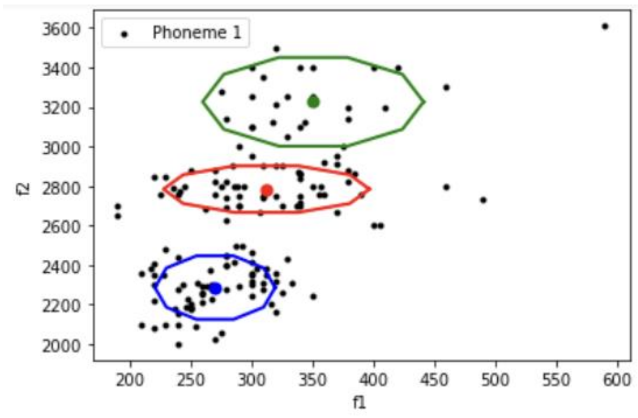
I have performed three trials for the clustering with K=3. My take on their comparison is that we are aware that the initial parameter for these clusters is random, but as we are following the EM algorithm, the parameters (mean, covariance matrix and GMM weights) are trained to fit the dataset to maximize the log likelihood for a data point to lie in a particular cluster. After executing the algorithm, you can notice that the output looks very similar, because those trained values are approximately the same for each run (reaching the same local minima ). My argument is supported by plots and metrics shown below for 3 of my trials.

**Trial 1:**

Starting Point:



Final Point:

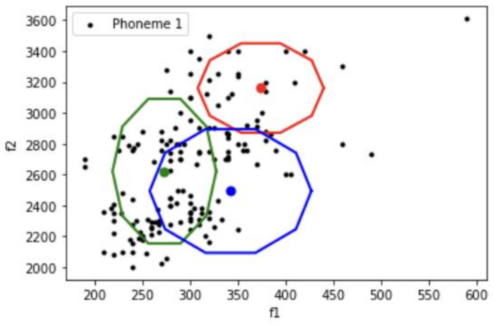


MoG Parameters:

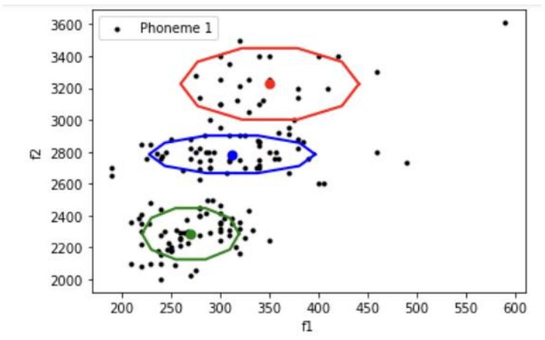
```
Mu:  [[ 312.5817      2783.8562 ]
      [ 350.83713    3226.1108 ]
      [ 270.39526    2285.4658 ]]

S:  [[[ 3562.74335101     0.        ]
     [    0.        7652.1949851 ]]

    [[ 4101.73123675     0.        ]
     [    0.       27888.44414467]]

    [[ 1213.73659901     0.        ]
     [    0.       14278.48433465]]]
```

**Trial 2:**

Starting Point:



Final Point:

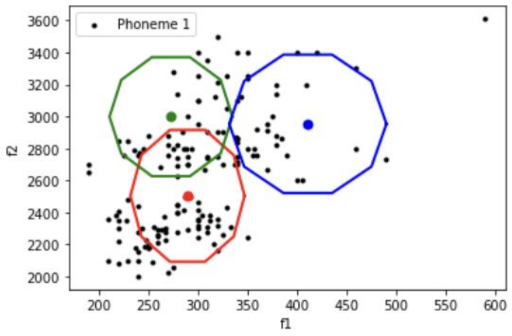


MoG Parameters:

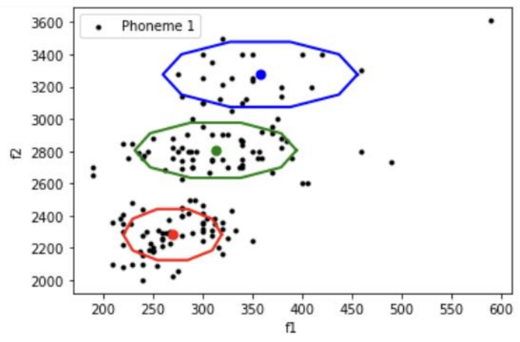
```
Mu:     [[ 350.8446      3226.3394 ]
          [ 270.3952     2285.4653 ]
          [ 312.59125   2783.898 ]]


S:      [[[ 4102.87537499      0.        ]
           [    0.         27829.54221494]]

          [[ 1213.73843494      0.        ]
           [    0.         14278.42029951]]

          [[ 3562.59743765      0.        ]
           [    0.          7657.84897238]]]
```

**Trial 3:**

Starting Point:



Final Point:



MoG Parameters:

```
Mu:   [[ 270.1105        2282.3682 ]
        [ 313.47006    2805.3816 ]
        [ 358.194        3275.9473 ]]


S:   [[[ 1216.39544413     0.       ]
       [    0.         13863.71328176]]

      [[ 3296.20862381     0.       ]
       [    0.         16225.21316834]]

      [[ 4772.58892353     0.       ]
       [    0.         22584.94581971]]]
       [    0.         14278.48433465]]]
```

Ques 5.

We need to get predictions on your data using both phenome id's Mixture of Gaussians' (MoGs) parameters. The output will give us the probability of how likely a point is to lie in one out of 'k' clusters. Without normalizing these probabilities, I have taken maximum value from both of these predictions' matrices separately along the row axis in the dataset and then concatenated these maximum values; column wise; to then further compare them and determine the maximum of maximum probability that will give us which phenom id is highly probable to include that instance in itself. We are using the 'argmax' to get the index of which column had the highest probability. The indexes of columns stacked before comparison are 0 and 1 for phenome id 1 and 2. So to map them with same values, I just added 1 in the resultant of argmax to get the outputs as either 1 or 2.

This approach is broken down to find which cluster is highly likely to represent that data point but because we are classifying based on class (phenome id), we are interested in which phenome id's cluster showed the highest probability.

To determine the accuracy using predictions and ground truth, I am creating a new array for ground truth which is represented by how our 'X' dataset is created using first all the instances of phenome id: 1 and then rest of phenome id: 2. Then simply comparing the values of predictions and ground truth to get Boolean values for match as True and False for mismatch. Summing the True(1) values and dividing it by the total number of instances will give us the accuracy of the predictions.

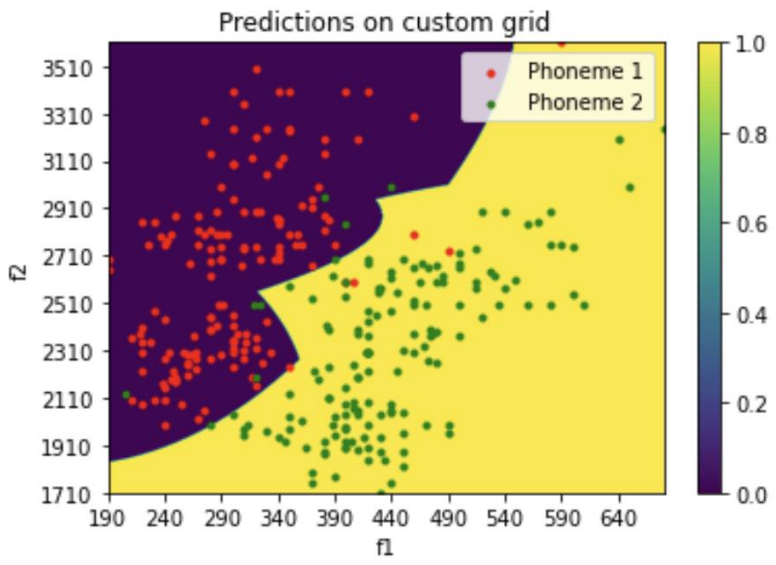
We got the accuracy of **95.3947%**

Ques 6.

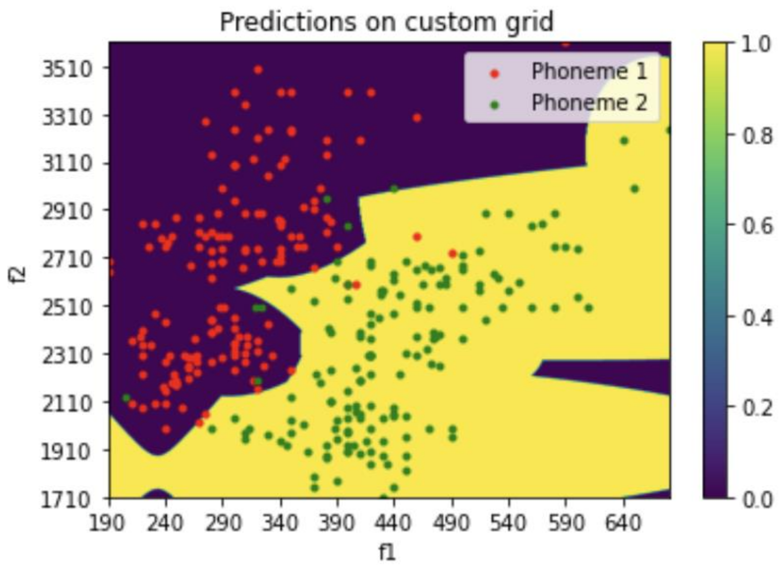
Increasing the number of clusters provides more room for data to belong to separate clusters and it increases the overall metric of log likelihood. That's why when we increased the number of clusters from k=3 to k=6, the accuracy of the predictions increases to **95.7237%**

Ques 7.

Classification matrix for k=3



Classification matrix for k=6



**Comparison:**

As we increased the number of clusters, the number of locations for a point to belong to through a cluster increases as well. This change starts fitting the model more closely, hence resulting in less generalized solutions or we can say that clusters are overfitting the data. This increase in accuracy looks promising but while we are overfitting the data, it is not what we want from the model. You can observe certain holes in the regions of the classification matrix when k=6 when compared to that of k=3. This looks like certain gaussians for k=6 in both of our different classes is trained to capture even anomalies of a distribution. I am indicating towards overfitting because the improve in accuracy is miniscule with respect to the increase in the number of clusters. The same can be proven by the visualizations as well.

## Ques 8:

After running for certain number of iterations, we are getting a value error that says:

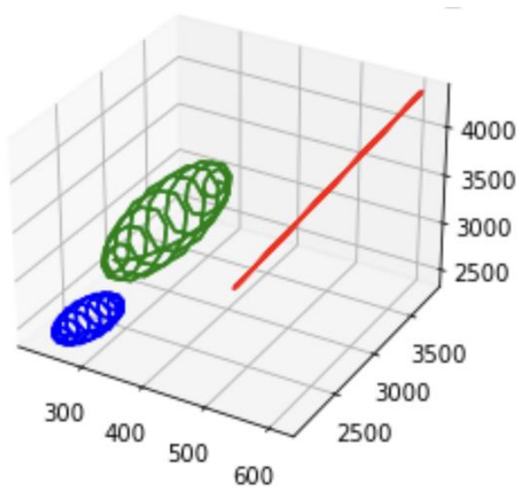
```
ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

This error occurred because the addition of 3rd column in the dataset which is the summation of feature 1 and feature 2, is a dependent value. With this we increased the data dimensionality higher than what is provided with the data samples hence causing the covariance matrix to be singular. In GMM, covariance matrix is one of our parameters hence, due to it being singular (determinant = 0), coefficient of MoG gives an infinite value as the output due to division by zero.

## Ques 9.

Ways to overcome singularity problem:

1. Adding a small value 'epsilon' to the diagonal positions of our covariance matrix at each update will remove any chance of getting a '0' as a value in it. This solves the purpose of mitigating the singularity problem and execute the code without any error.
   In my case, I have set the value of epsilon as 1e-4 (0.0001) . We can see k=3 best fitted gaussians in 3-D space comprised of 3 features in the data.



   So, I created an identity matrix with the same size of covariance matrix for each cluster and multiply epsilon with it to get a diagonal matrix with non zeros values as epsilon. Next and last thing to do is to just add this identity matrix with covariance matrix and prevent the resultant of covariance matrix to ever reach zero.
2. Another way to solve this singularity problem is to add {standard deviations of the features along all data points; multiplied with an identity matrix} to the covariance matrix in the same way mentioned above. This and many others approach like this are meant to avoid getting the determinant of the covariance matrix as '0'.