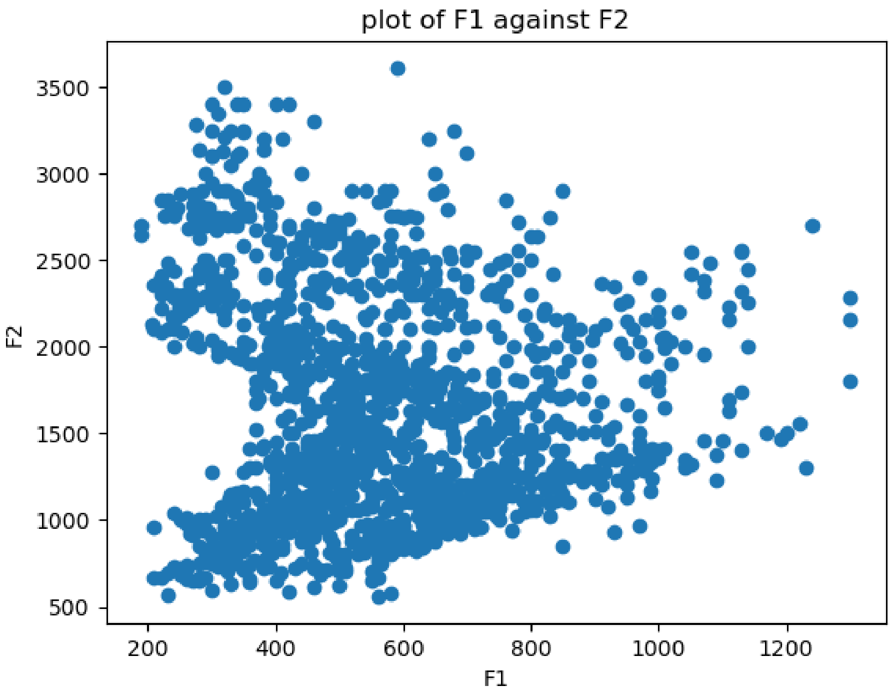
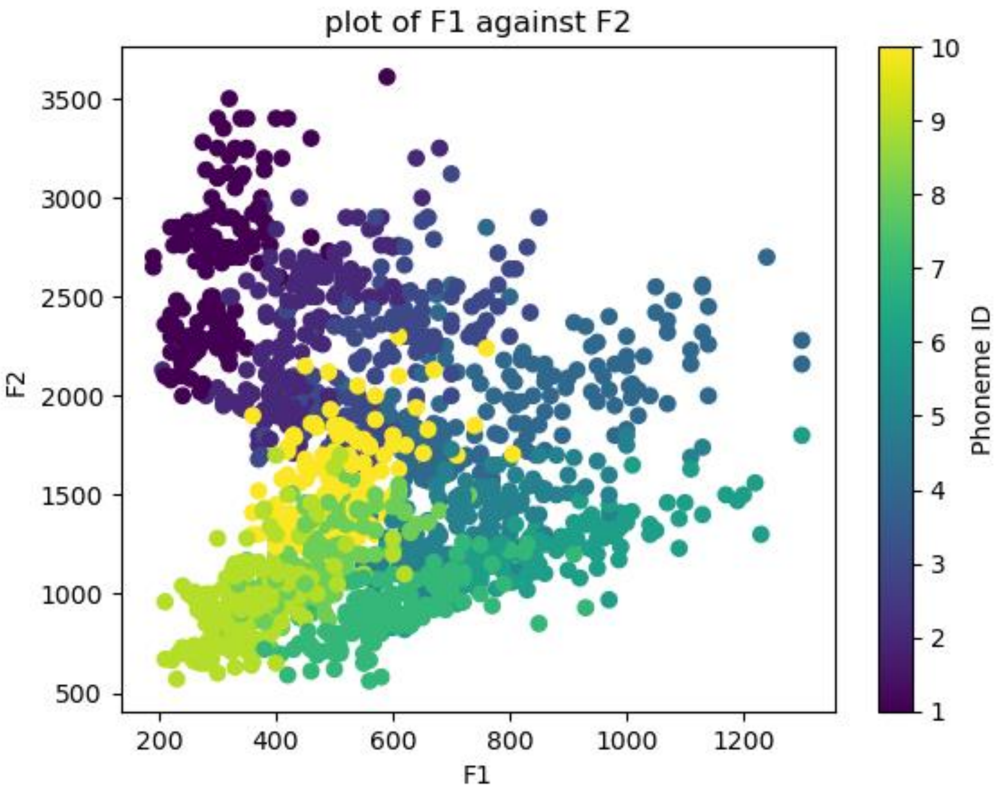
Sai Vivek Rambha (2023585403) Assignment 2 Density Estimation ESC708P

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.



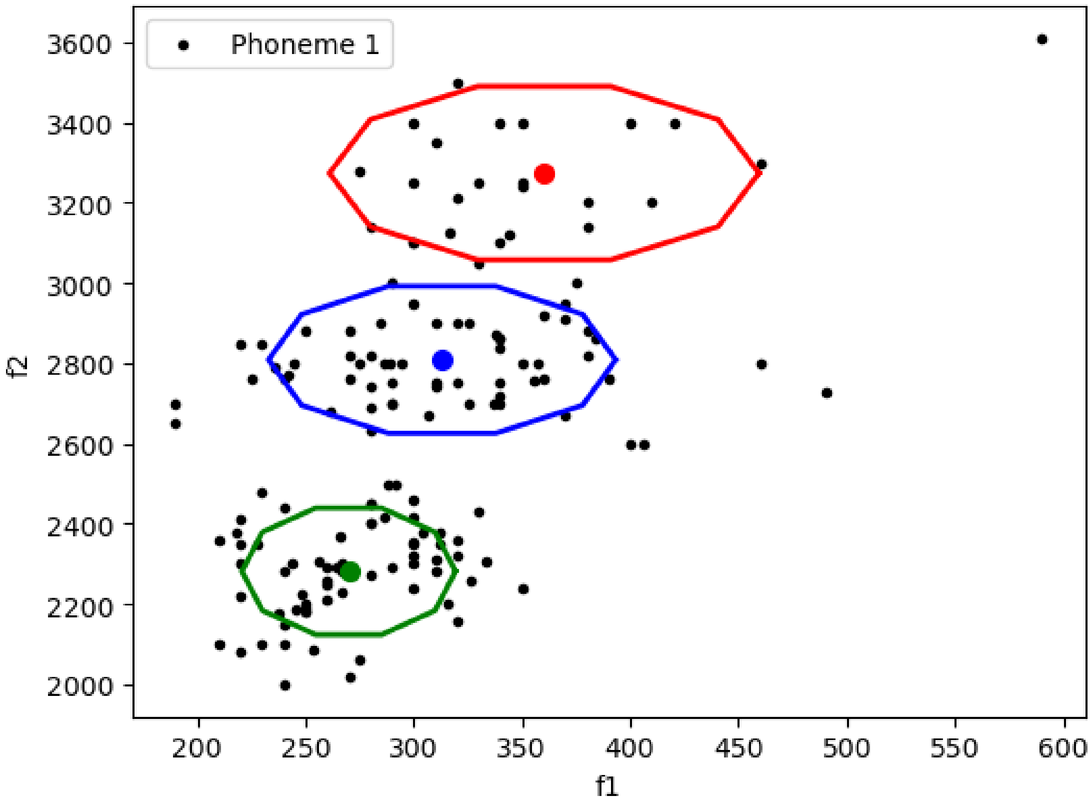
If we are plotting the graph without considering phoneme then it's difficult to determine the clusters shown in the picture above. To clearly see different clusters, we have to scatterplot with phoneme_id, which is an array containing (1-10) of each samples.

Each phonemes is distinguished by different colour in the figure below.



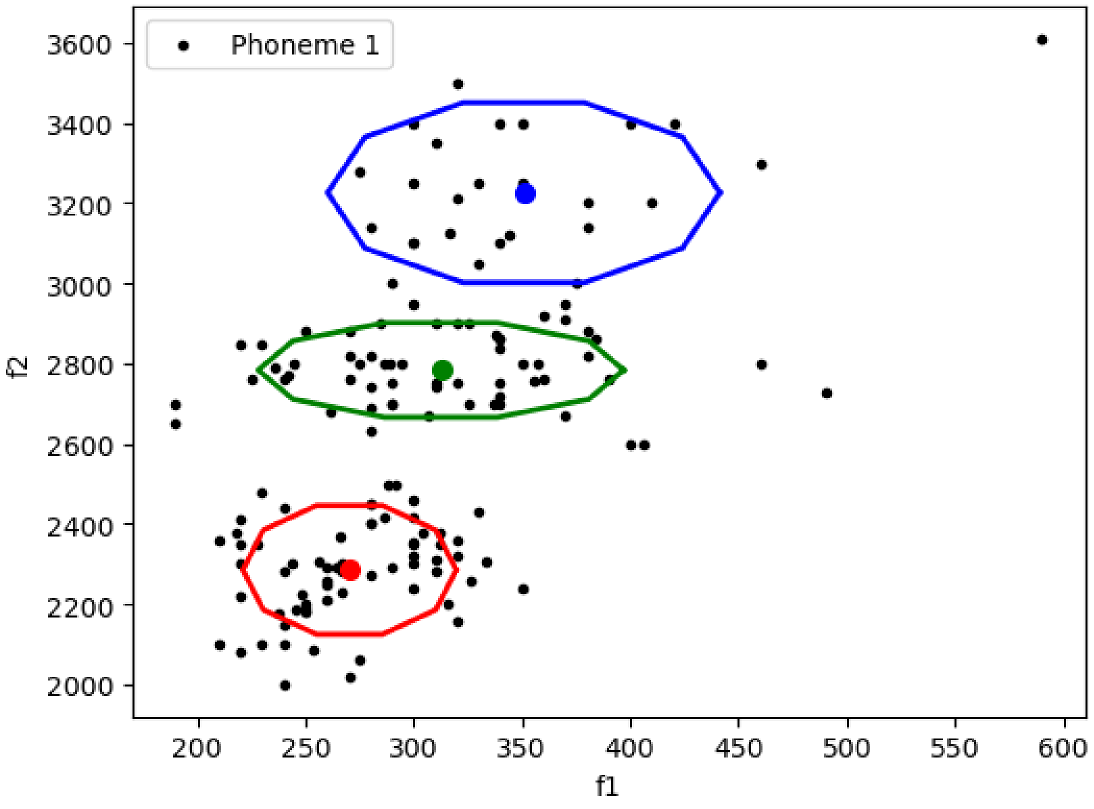
Q2. Run the code multiple times for K=3, what do you observe? Use figures and the printed MoG parameters to support your arguments.

First run:



```
Finished.

[[ 360.27072 3274.0378 ]
 [ 270.00995 2281.5938 ]
 [ 313.1415  2808.8208 ]] [[[ 4920.8985909       0.        ]
  [    0.         25843.08911323]]

 [[ 1216.20078556     0.        ]
  [    0.         13789.97252527]]

 [[ 3214.14069117     0.        ]
  [    0.         18581.28577416]]]
```

Second run:

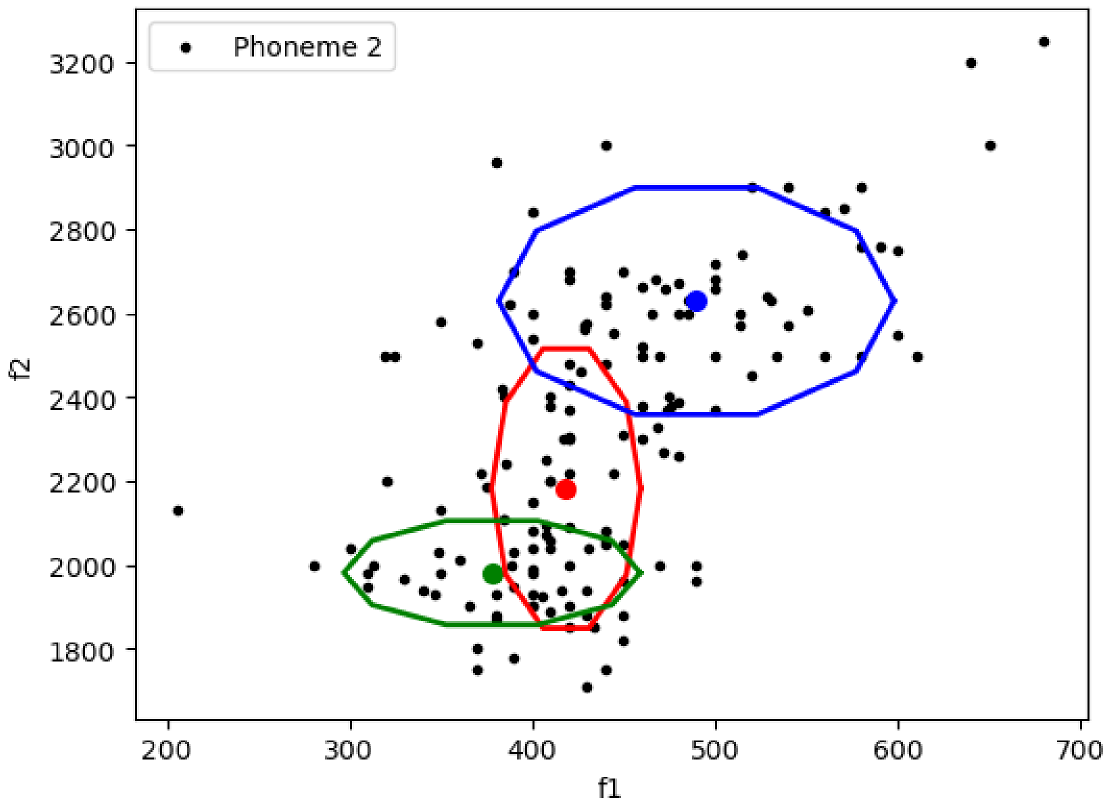


```
Finished.

[[ 270.3952   2285.4653 ]
 [ 312.59125 2783.898   ]
 [ 350.8446   3226.3394 ]] [[[ 1213.73843494      0.          ]
  [    0.          14278.42029989]]

 [[ 3562.59743769      0.          ]
  [    0.           7657.84896997]]

 [[ 4102.87537456      0.          ]
  [    0.          27829.54223973]]]
```

When running the code multiple times, the clusters are changing due to random initialization of the cluster parameters like mean, covariance and weights. The random initialization can affect the cluster parameters because the data points are not well separated. Between first and second runs we observe that there is a difference in their means. In the first run we get mean of 360.27 and in the second run we get 270.39 for cluster 1(difference can be visually observed). This indicates that there is a change in the centres for each run. For the second cluster there is a change in covariance which can be depicted by the size of the cluster. As in the fist run the cluster 2 takes covers more space and hence the chance of predicting the correct observation is very less. Cluster 3 can be predicted easy in almost of the runs.

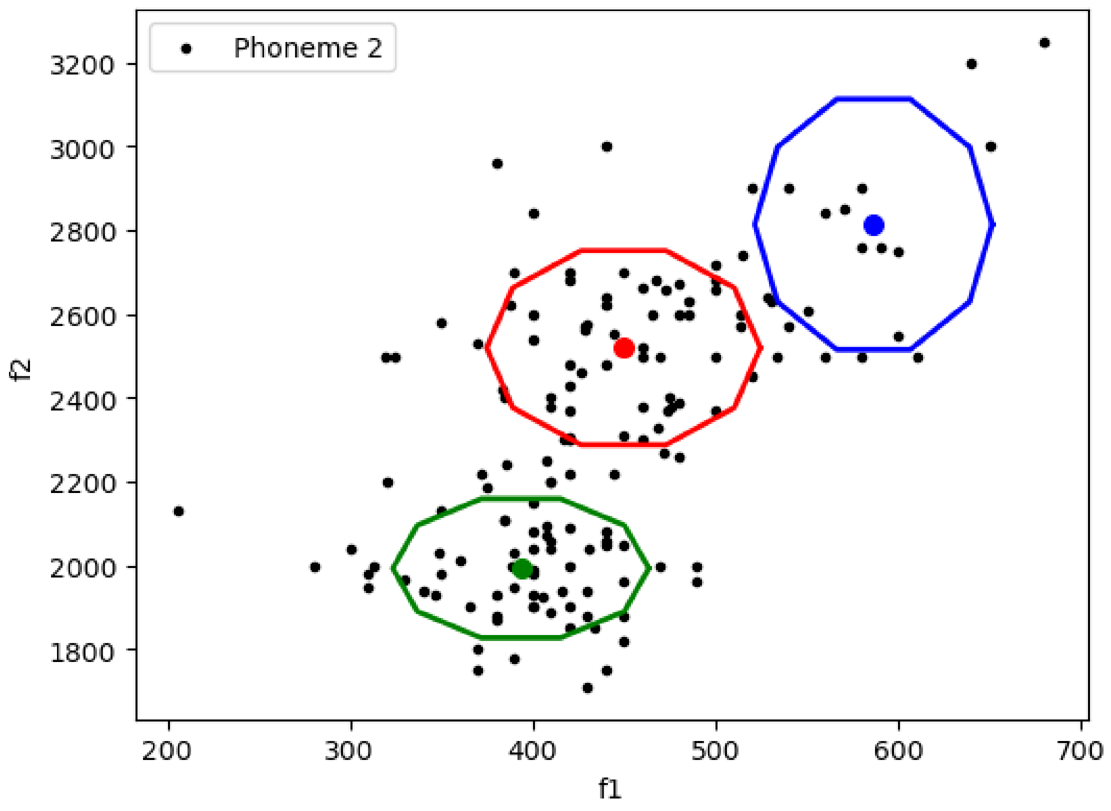
Q3. Repeat the training for phoneme_id = 2 and save the parameters.

First run:



```
Finished.

[[ 418.51523 2182.2402 ]
 [ 377.9256  1981.0283 ]
 [ 489.79764 2629.1648 ]] [[[  832.65501022      0.          ]
  [    0.          61348.96052183]]

 [[ 3292.50402994      0.          ]
  [    0.           8497.7253794 ]]

 [[ 5859.84604811      0.          ]
  [    0.          40463.66026988]]]
```

Second run:

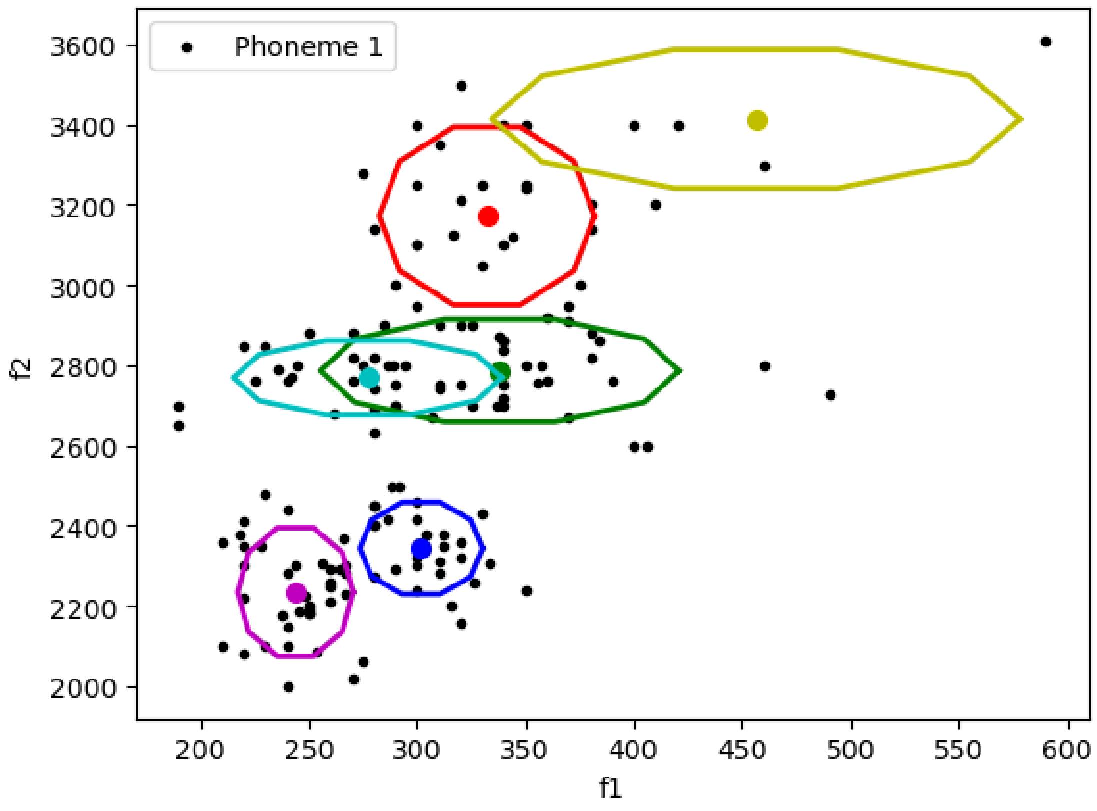


```
Finished.

[[ 449.6616   2519.6519 ]
 [ 393.45148 1993.0685 ]
 [ 586.5339   2814.0676 ]] [[[ 2808.31194549      0.          ]
  [    0.          29723.58265146]]

 [[ 2454.98266237      0.          ]
  [    0.          15260.99006444]]

 [[ 2112.3847094       0.          ]
  [    0.          49424.48559975]]]
```

For phoneme = 2 we can clearly see that there are changes in the cluster for each run, which is due to picking random mean form the data samples.

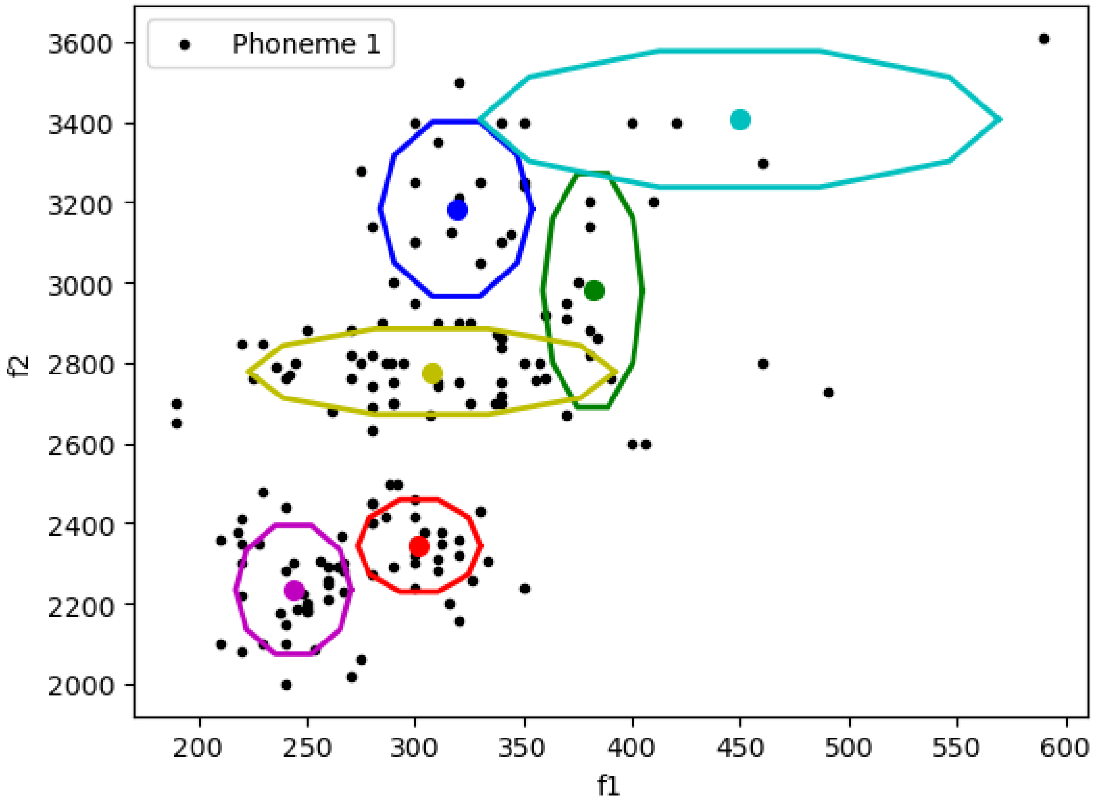
Q4. Repeat training for K=6 for phoneme_id =1 and phoneme_id=2 and save the model parameters [1 mark]

First run: p_id= 1



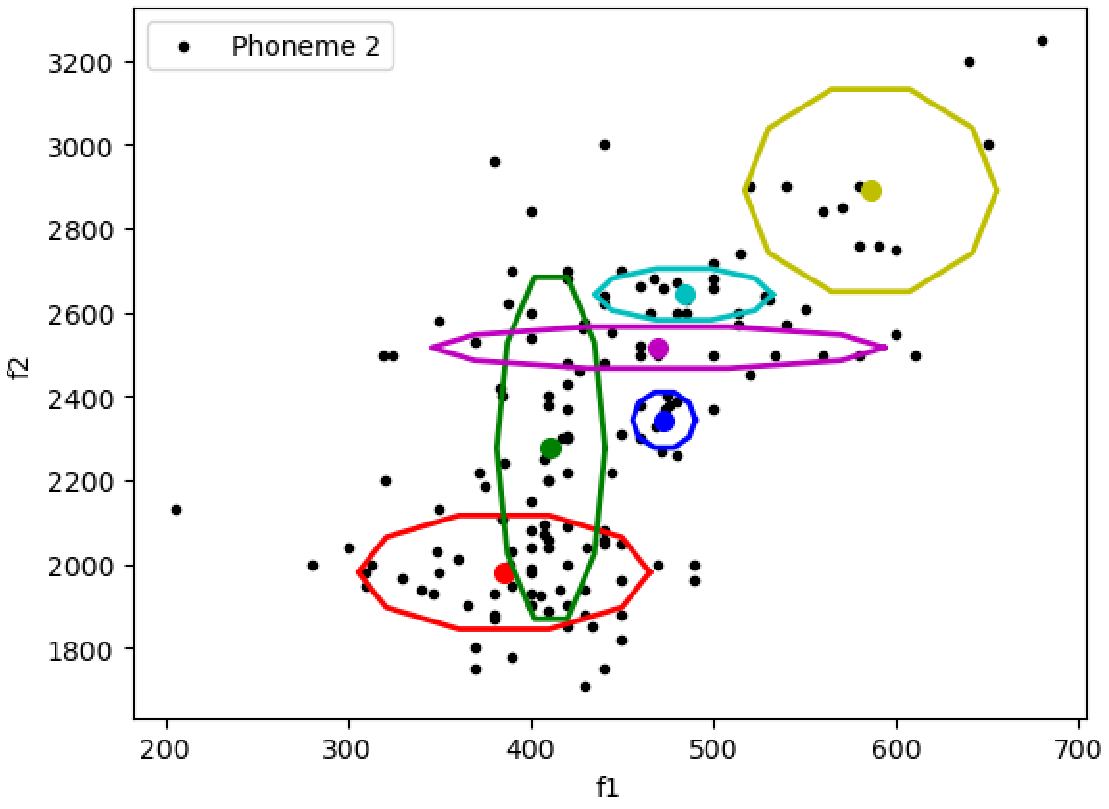
```
Finished.

[[ 332.19617 3172.5486 ]
 [ 338.09848 2786.807  ]
 [ 301.88916 2343.8782 ]
 [ 277.17642 2769.303  ]
 [ 243.85179 2234.581  ]
 [ 456.36472 3414.644  ]] [[[ 1232.6982703      0.         ]
  [    0.         27171.90924259]]

 [[ 3414.832072      0.         ]
  [    0.          8997.69961066]]

 [[  401.40626557     0.         ]
  [    0.          7241.39390788]]

 [[ 1919.03068593     0.         ]
  [    0.          4752.18047656]]

 [[  356.25724014     0.         ]
  [    0.         14192.93246887]]

 [[ 7446.24038945     0.         ]
  [    0.         16547.09349079]]]
```

Second run:



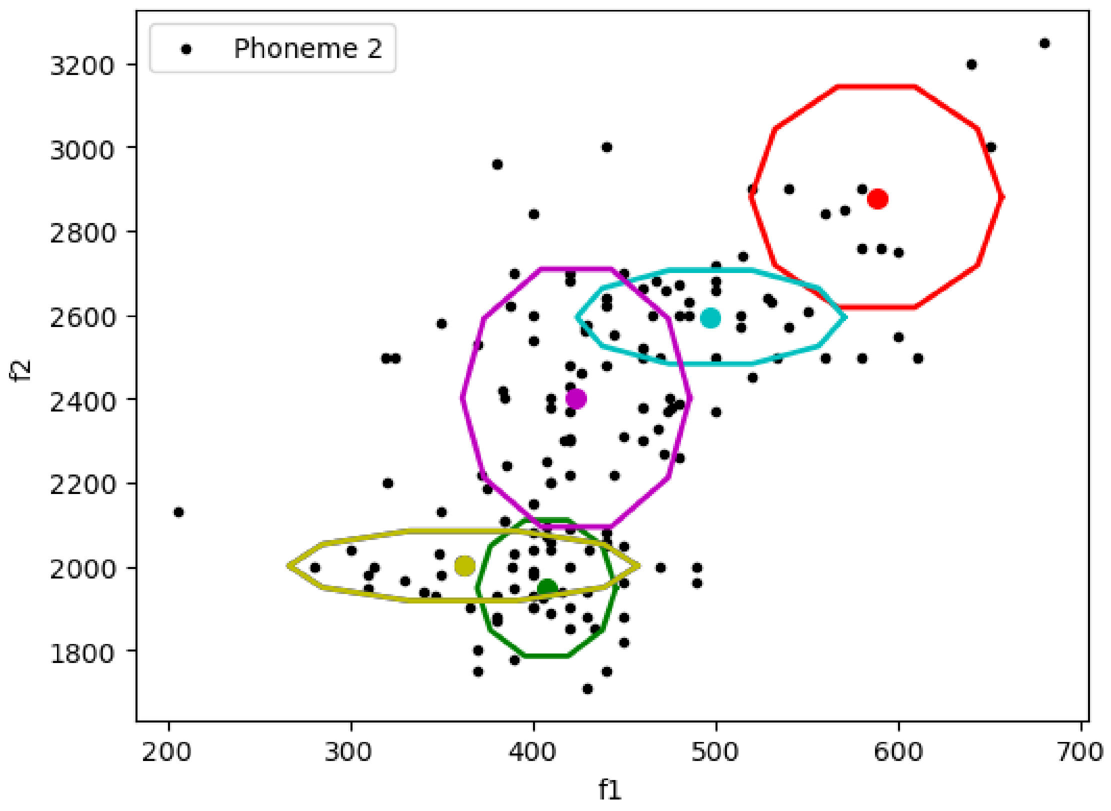
```
Finished.

[[ 301.87732 2344.017  ]
 [ 381.99838 2980.461  ]
 [ 318.88983 3183.2075 ]
 [ 449.5497  3407.1775 ]
 [ 243.8524  2234.6006 ]
 [ 307.70386 2777.8835 ]] [[[  401.09141017     0.        ]
  [    0.          7258.77168191]]

 [[  263.49752194     0.        ]
  [    0.         46869.03224024]]

 [[  613.73766044     0.        ]
  [    0.         26207.65186903]]

 [[ 7167.72718156     0.        ]
  [    0.         15966.31457687]]

 [[  356.21052282     0.        ]
  [    0.         14197.35323097]]

 [[ 3581.45746994     0.        ]
  [    0.          6251.48137889]]]
```

P_id = 2
First run:



```
Finished.

[[ 385.47995 1980.3473 ]
 [ 411.13806 2276.6367 ]
 [ 473.10397 2343.6028 ]
 [ 483.82513 2643.2913 ]
 [ 469.74683 2516.383  ]
 [ 586.1576  2891.1787 ]] [[[ 3182.36151835      0.        ]
  [    0.          10153.42301383]]

 [[  437.18133373      0.        ]
  [    0.          91838.94671834]]

 [[  146.70902766      0.        ]
  [    0.           2410.89089225]]

 [[ 1184.32152715      0.        ]
  [    0.           2070.99887576]]

 [[ 7678.98732038      0.        ]
  [    0.           1354.28774952]]

 [[ 2383.25961988      0.        ]
  [    0.          32121.81558442]]]
```

Second run:



```
Finished.

[[ 588.04987 2880.6833 ]
 [ 407.47556 1948.0458 ]
 [ 361.9156  2001.0953 ]
 [ 497.33145 2594.4016 ]
 [ 423.69812 2401.4875 ]
 [ 361.9156  2001.0953 ]] [[[ 2350.7395569      0.        ]
  [    0.          38315.22783661]]

 [[  720.59577665     0.        ]
  [    0.          14550.96951502]]

 [[ 4547.63268841     0.        ]
  [    0.           3823.6827656 ]]

 [[ 2682.82799302     0.        ]
  [    0.           6884.37427068]]

 [[ 1940.67074081     0.        ]
  [    0.          52200.15908677]]

 [[ 4547.63268841     0.        ]
  [    0.           3823.6827656 ]]]
```

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.

In the first step we have inds_1 and inds_2 that contain indices of phoneme 1 and 2 respectively. Then we concatenate both to create inds_1_and_2. We are using N and D to store sample size and dimensions of the dataset created.

Next, we are loading the values from trained data model that we saved in Q3 and getting the values of mu(mean), s(covariance) and p(weights) of the components.

Then we are calculating the predictions for each data point in the dataset for phoneme_1 and similarly for phoneme_2.

Next, we are calculating the maximum likelihood by summing the predictions of phoneme_1 and phoneme_2.

To calculate accuracy, we are dividing the dataset in half. First half contains phoneme_1 and second contains phoneme_2. By comparing the maximum likelihood of each half to the expected phoneme, the number of correct classifications is determined. At the end we are calculating the accuracy by adding the classified samples and dividing with total number of samples, resulting in an accuracy of 95.06% for K=3 between 1 and 2.

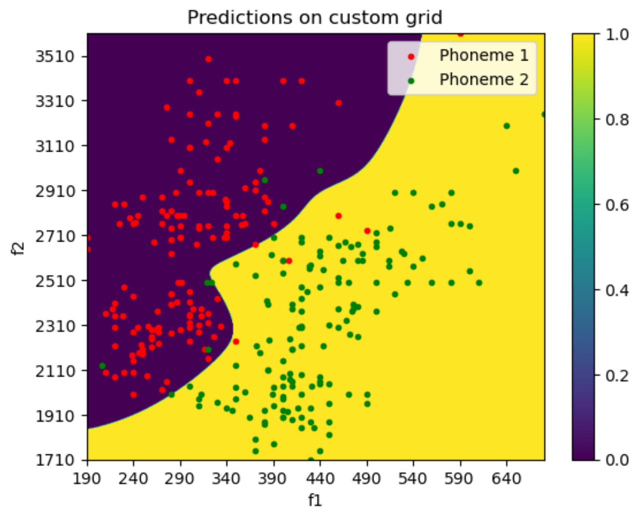
Result:
```
Accuracy for K=3 95.06578947368422
```

Q6. Repeat for K=6 and compare the results in terms of accuracy.

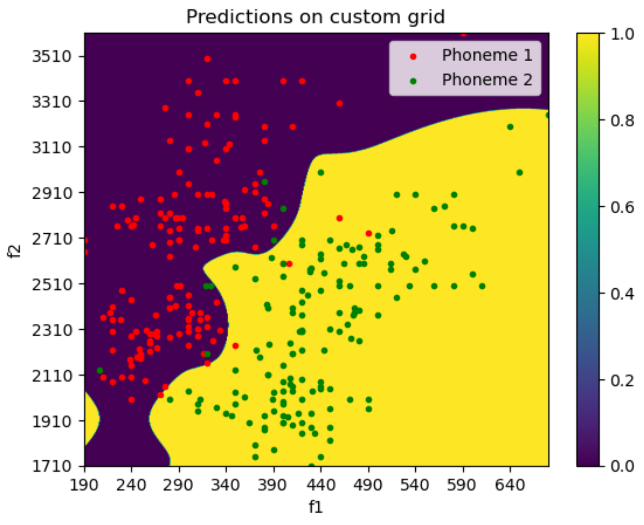
The GMM's trained on the K=3 has an accuracy of ~95.06% and for K=6 it has accuracy of ~95.72. Both models have approximately the same accuracy for. But the model where K=3 was found to train faster and getting approximately the same accuracy.

Result:
```
Accuracy for K=6 95.72368421052632
```

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.



K = 3



K = 6

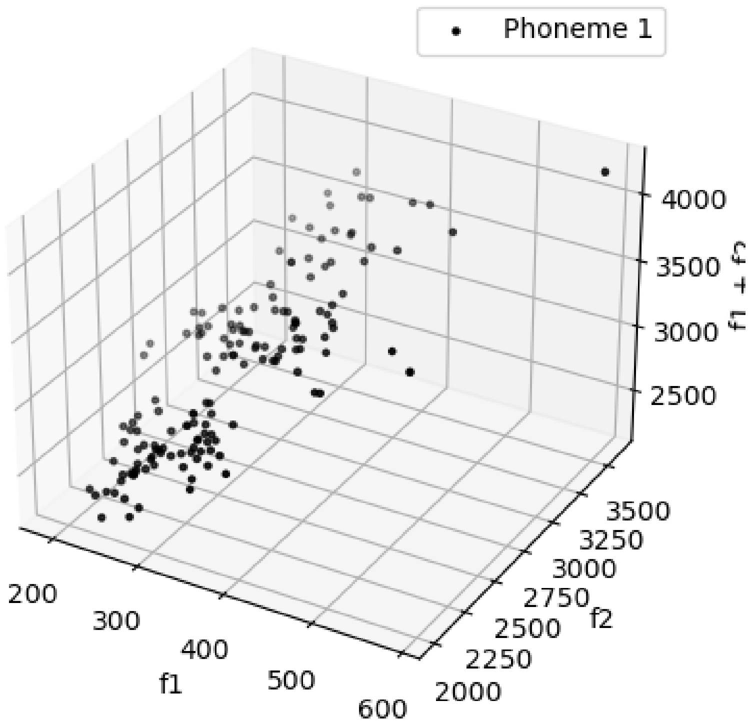
From the above figures we can see that for classification k=3, it can draw the boundary between two phonemes is much clear compared to K=6. For K=6 the boundary between two phoneme is more complex. There are more regions where classification seems purple.

From this we can conclude that with K=3 the model is simpler and has better view of the data points, which suggests that the model is a perfect fit for the data set. In the case of K=6 the model is fitting the data more closely indicating that the model is overfitting the dataset.

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

```
Iteration 001/150
Iteration 002/150
Iteration 003/150
Iteration 004/150
Iteration 005/150
Iteration 006/150
Iteration 007/150
Iteration 008/150
```

```
C:\Users\vivek\AppData\Local\Temp\ipykernel_8060\3568656960.py:53: RuntimeWarning: divide by zero encountered in double_scalars
  Z[:, i] = p[i]*(1/np.power(((2*np.pi)**D) * np.abs(s_i_det), 0.5)) * np.exp(-0.5*np.sum(x_s_x, axis=1))
```



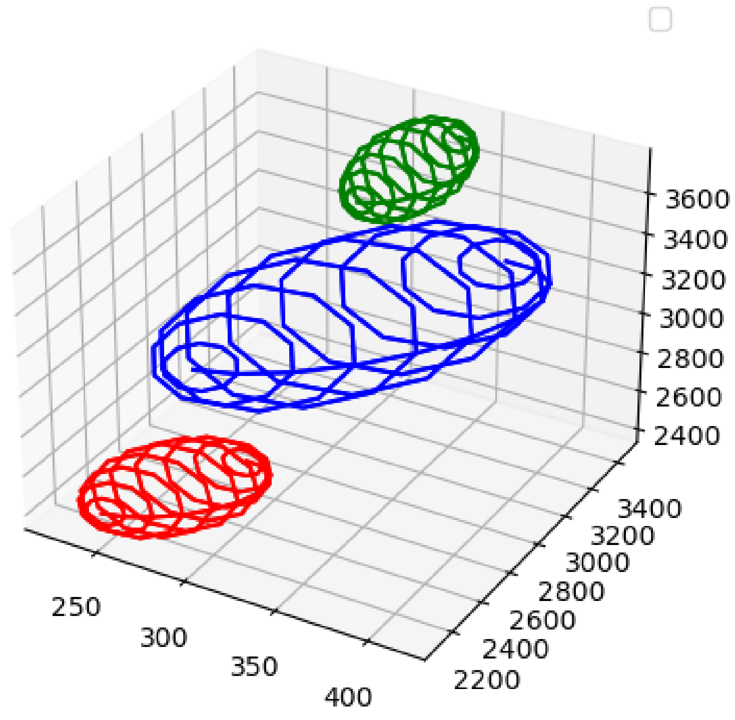
The errors shown in the figure, is due to singularity problem. In this the addition of the third dimension which is dependent on the F1 and F2 resulting to a singularity covariance matrix that has zero as non-diagonal entries.

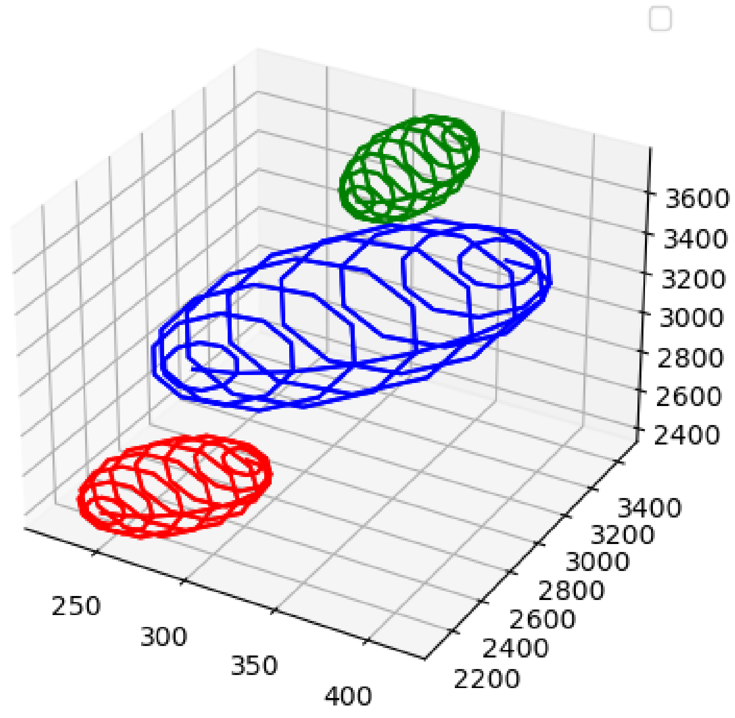
We can fix this problem by implementing regularization to the covariance matrix by adding the a diagonal matrix which contains the covariance so its always above some minimum value.

Q9. Suggest ways of overcoming the singularity problem and implement one of them. Show any training outputs in the report and discuss.

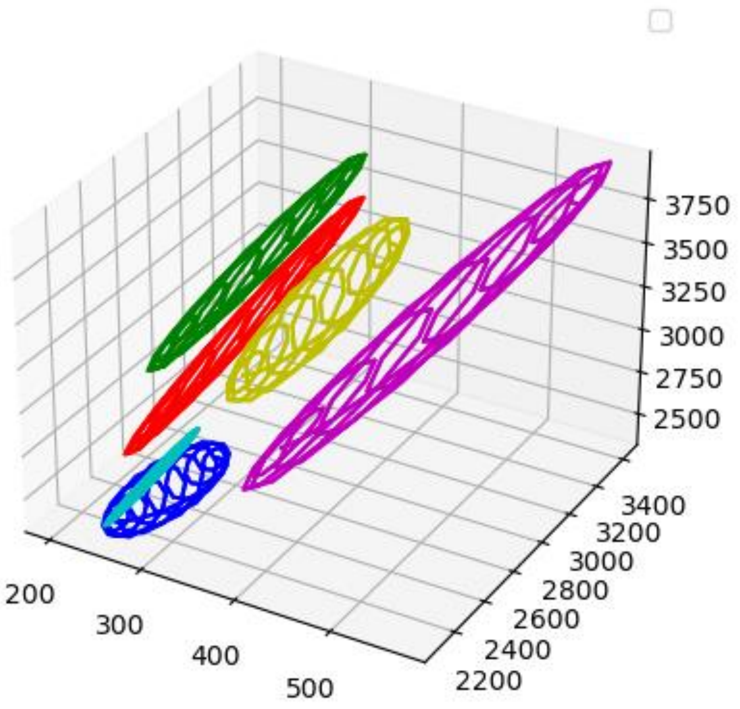
We implemented regularisation to the covariance matrix. First, we compute the covariance matrix of the dataset X and its transpose that is calculating the covariance between the features. Then the matrix is divided and add a small value of 0.001 to the diagonal so it never reaches 0.

k=3

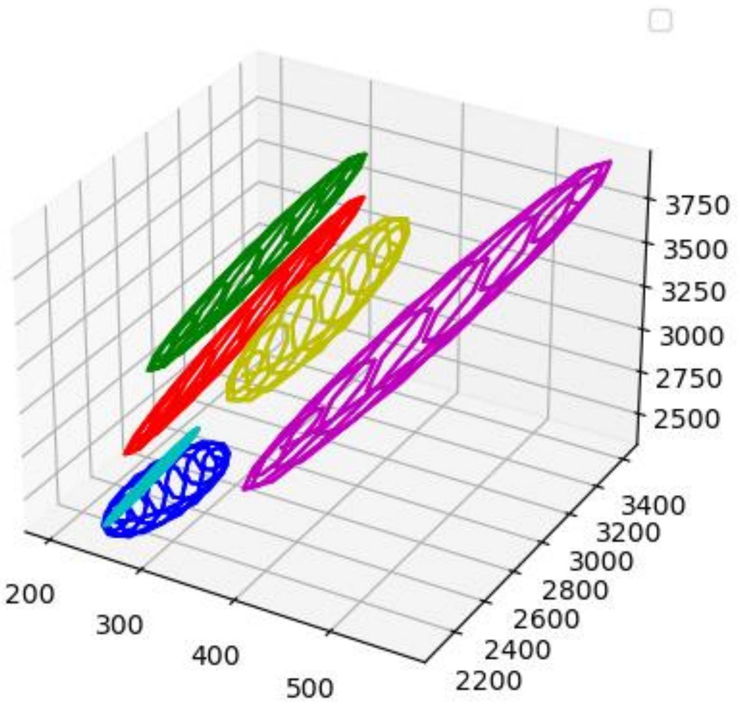




K=6:



3750
3500
3250
3000
2750
2500
3400
3200
3000
2800
2600
2400
2200
200
300
400
500



3750
3500
3250
3000
2750
2500
3400
3200
3000
2800
2600
2400
2200
200
300
400
500