1 Q1

By looking at Figure 1, we see that there are parts of the data that are very close together and some that are very far apart. From looking at this figure, it is not really obvious how many clusters there are and how the clusters are shaped, so doing a task like this by hand would be very difficult.

By observation, a possible grouping is shown in Figure 3. Comparing Figure 3 to Figure 2, we see that my group of clusters is far from the actual clustering. This is due to many reasons, mainly:

1. Overlapping clusters
2. Not knowing the number of clusters in advance

Now, looking at Figure 3, we see that the clusters are overlapping and spread out and as such, it is difficult to do an unsupervised task like clustering without knowledge in advance and it is even harder to do just by observation.

2 Q2

After running the code multiple times, the most common cluster that the model converges to is shown in Figure 4. In some cases, the model wasn't able to converge within 100 iterations. For example, in Figure 5 the model hasn't converged yet but we can see just by observation that it is likely to converge to the same clusters as in Figure 4. The model is also likely to converge to a different cluster group, for example, looking at Figure 6, we see that the cluster group is completely different from that of Figure 4. These are all perfectly sensible and the reason that we get different cluster groups is dependent on the initialization of the clusters, with different initializations resulting in different clusters that the model converges to. This is due to the fact that the model gets stuck in a local minimum in cost when doing the optimization.

Another observation was that even if the model converged to the same cluster group, it would usually be a permutation of the cluster group. For example, when the model converges to the most common cluster group, as shown in Figure 4, the individual clusters themselves are usually permutated, with the means and covariance matrices being permuted.

3 Q5

In words. To do classification using this, we select 2 different clustering models (for example, a clustering model that clusters phoneme 1 instances and a clustering model that clusters phoneme 2 instances). Then, for each of these clustering models, we take all points and work out the probability of each point being in each cluster. Then, we take the weighted sum of the probabilities and have a single probability for each model and then compare the probabilities. If the probability of a point is higher in model 1 than in model 2, then we say that

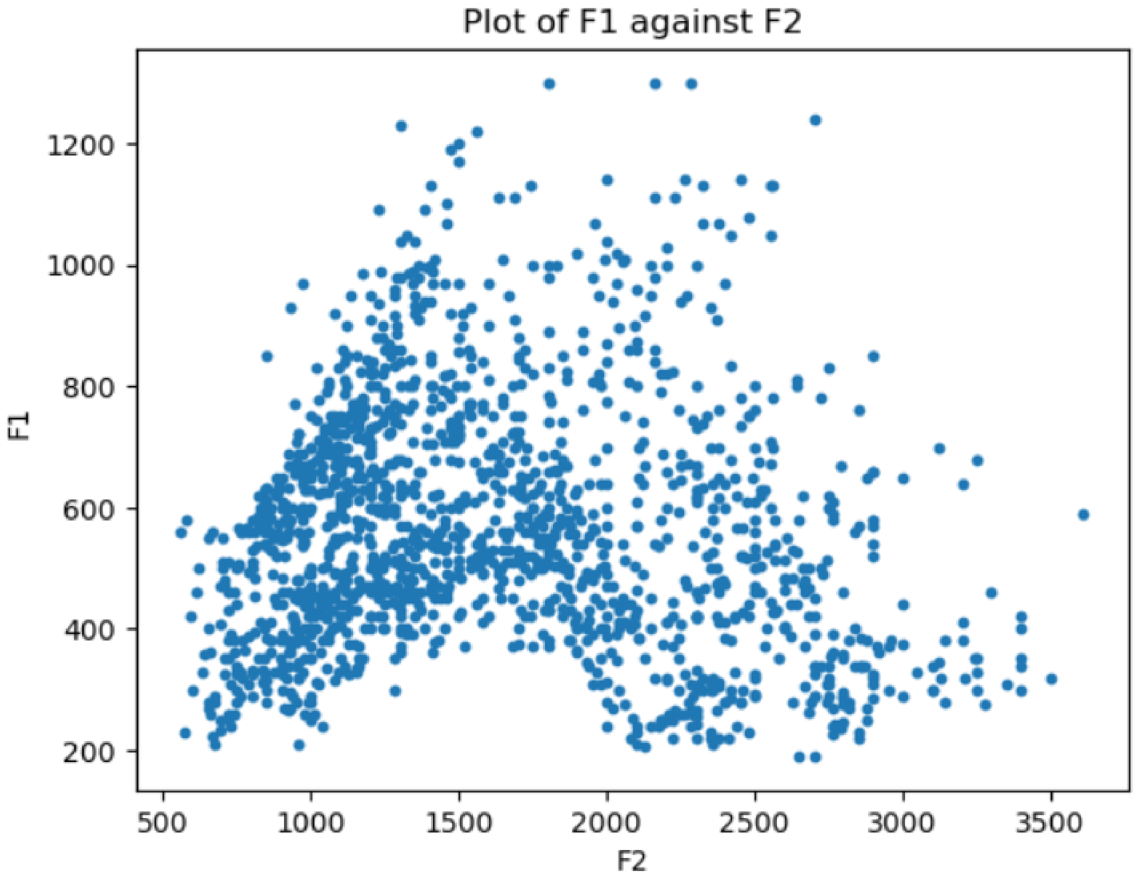


Figure 1: Scatter plot of F1 against F2

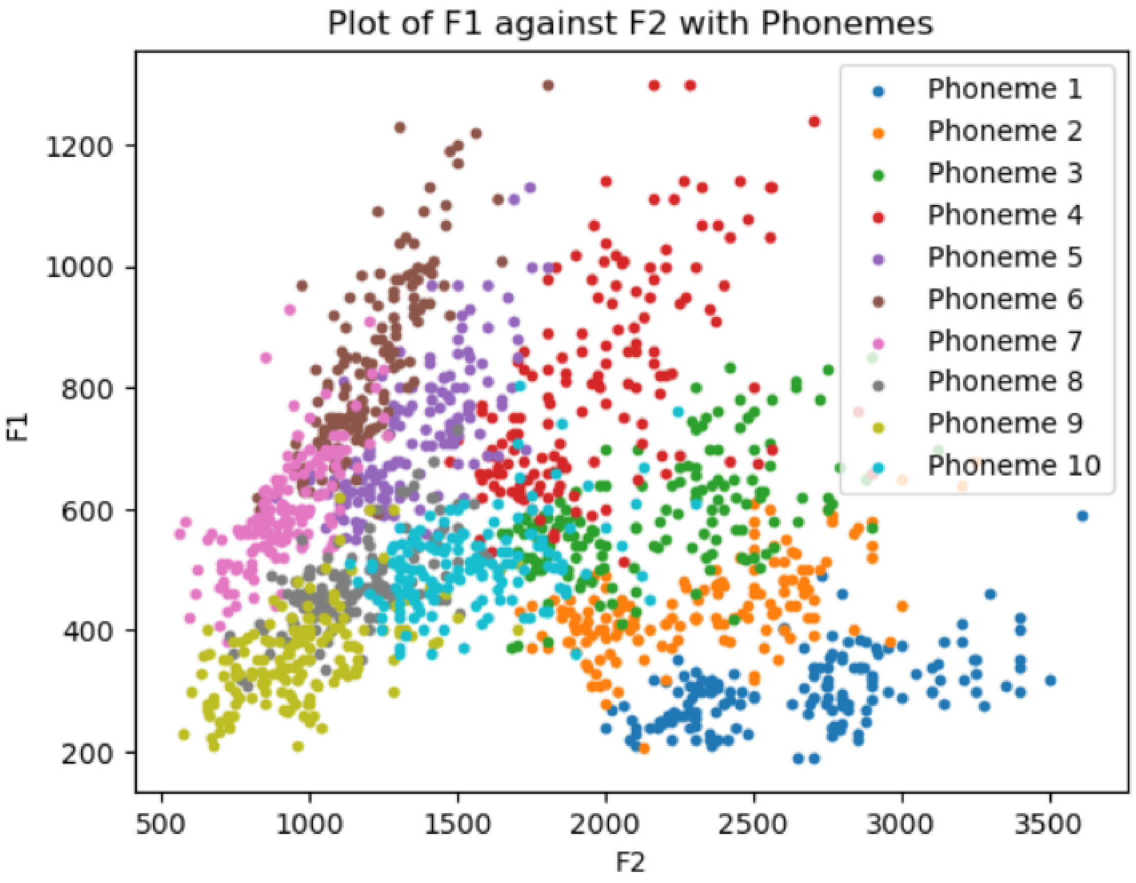


Figure 2: Scatter plot of F1 against F2 with the Phoneme labels

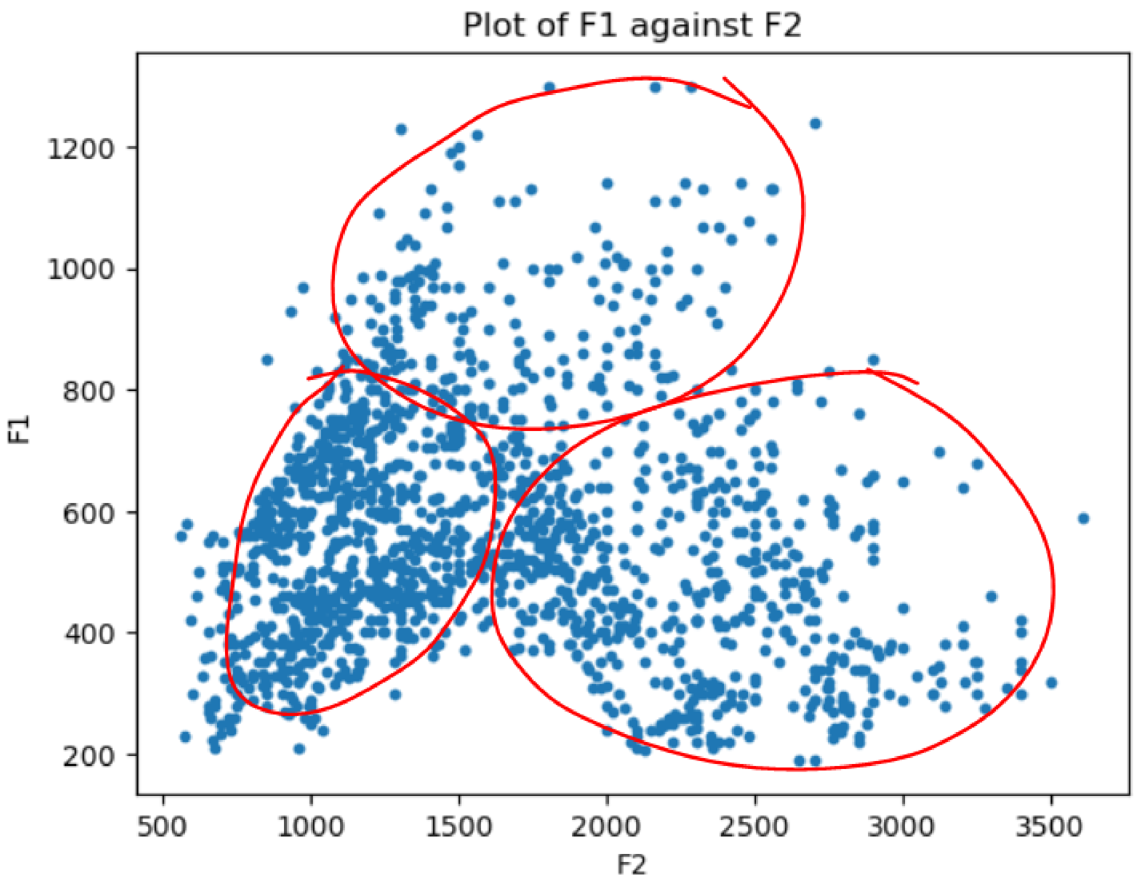
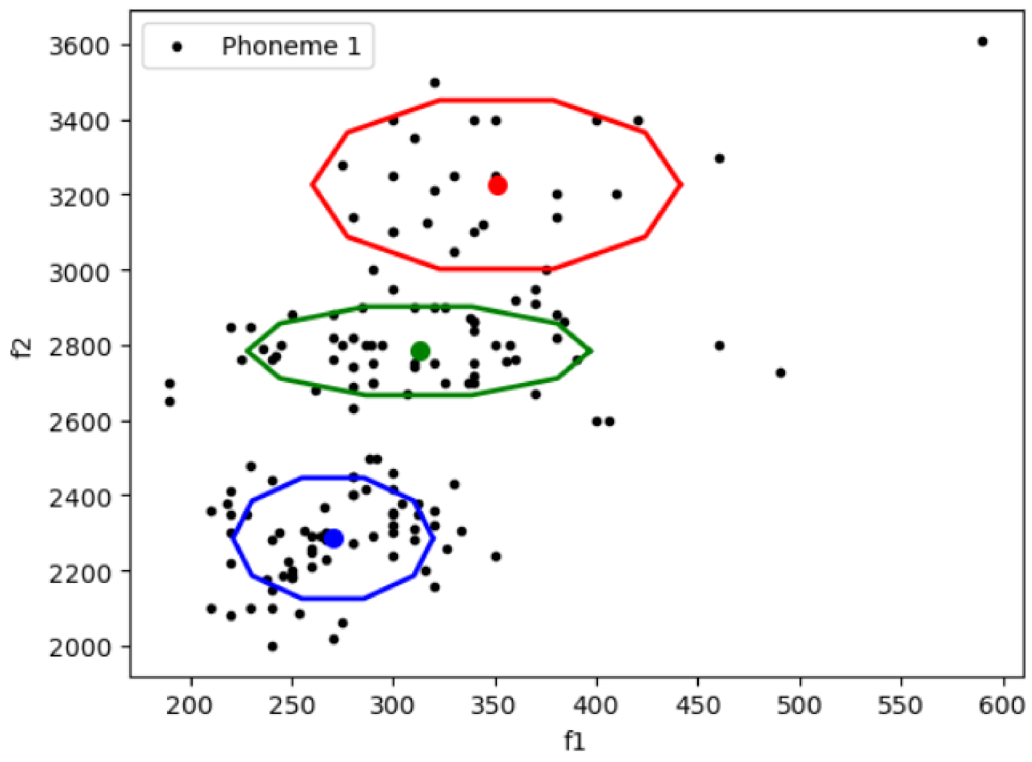


Figure 3: Scatter plot of F1 against F2 with a possible clustering



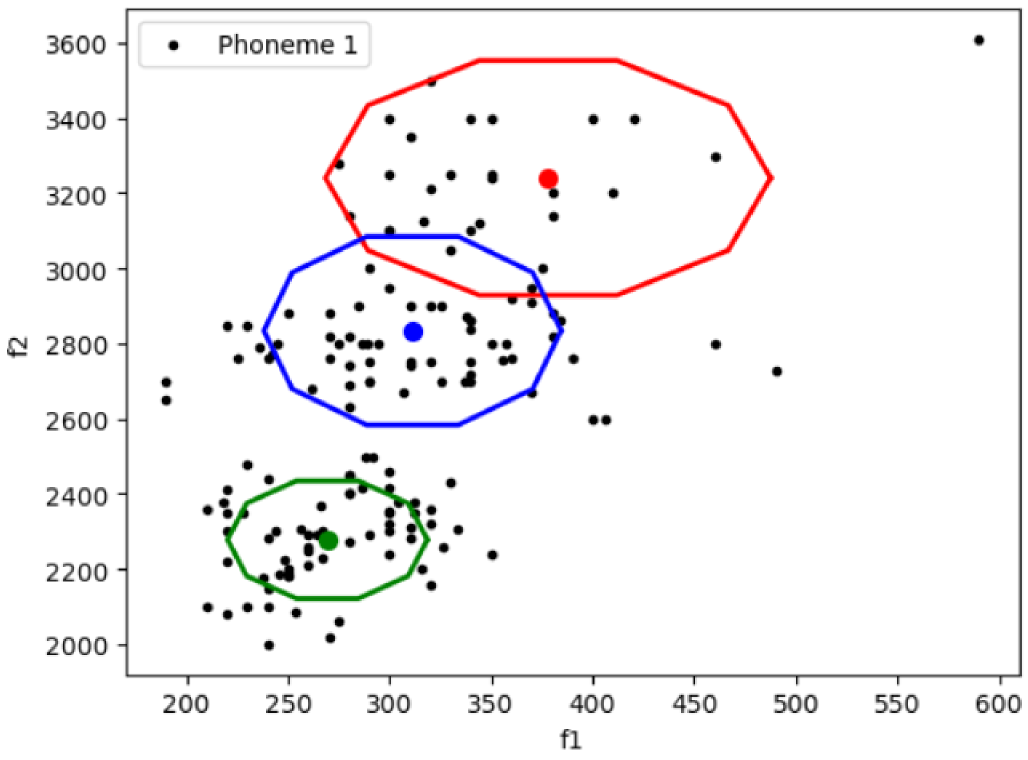
```
Finished.

[[ 350.8445  3226.3367 ]
 [ 312.59113 2783.8975 ]
 [ 270.3952  2285.4653 ]] [[[ 4102.86113629     0.        ]
  [    0.          27830.26007539]]

 [[ 3562.59929748     0.        ]
  [    0.           7657.77917016]]

 [[ 1213.73841297     0.        ]
  [    0.          14278.42036871]]]
```

Figure 4: Most common cluster that the model converges to.



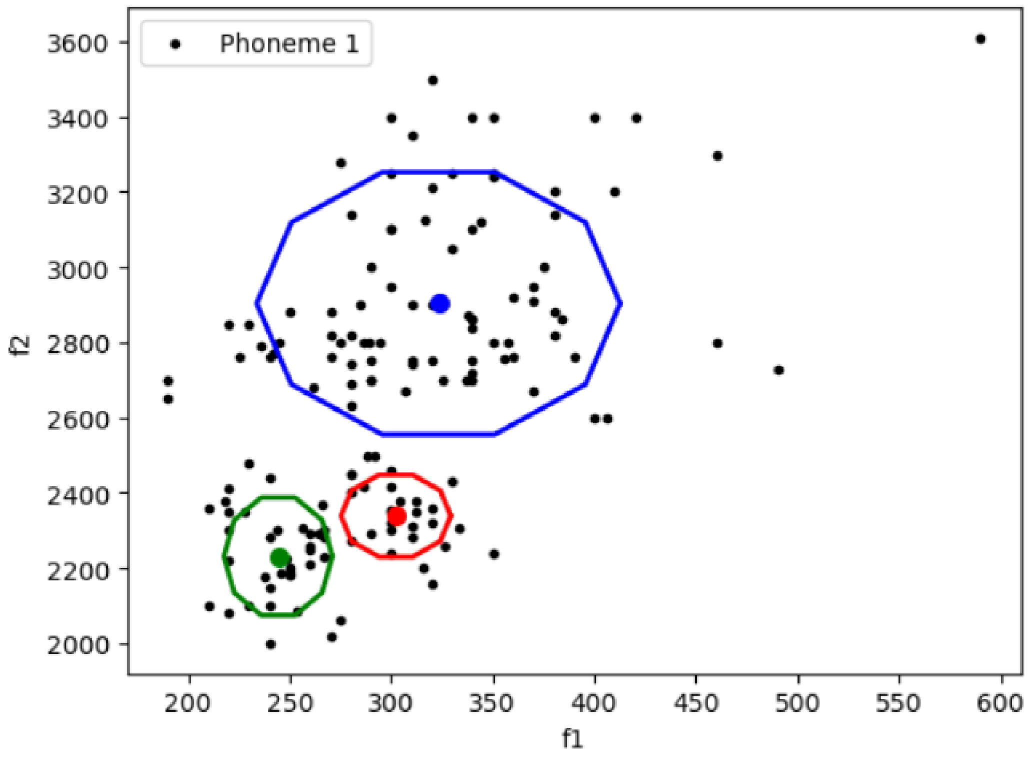
```
Finished.

[[ 378.0335   3240.903   ]
 [ 269.39517 2278.238   ]
 [ 311.28342 2834.159   ]] [[[ 6010.83631691     0.        ]
  [    0.         53868.64365804]]

 [[ 1210.04804822     0.        ]
  [    0.         13557.89923684]]

 [[ 2678.45390906     0.        ]
  [    0.         34692.03956196]]]
```

Figure 5: Model that hasn't converged in 100 iterations.



```
Finished.

[[ 302.26315 2338.855  ]
 [ 244.30171 2231.018  ]
 [ 323.24017 2903.5637 ]] [[[  365.77191084     0.        ]
  [    0.          6568.98351255]]

 [[  356.12422517     0.        ]
  [    0.         13577.92425959]]

 [[ 4002.53696926     0.        ]
  [    0.         67340.65015358]]]
```

Figure 6: A non-frequent cluster that the model converges to.

the point is classified as the points that model 1 is trying to cluster. Otherwise, the point is classified as the points that model 2 is trying to cluster.

In Practice. For each Gaussian cluster model (I.e., clustering phoneme 1 instances in one model and clustering phoneme 2 instances in the other model), we want to work out the probability of each point existing in that clustering model (I.e., the probability of it being in that class given that we assume it is a part of that class). This is given by:

$$p(\underline{x}) = \sum_{i=1}^{k} p(\underline{x}, c_i) = \sum_{i=1}^{k} p(c_i) p(\underline{x}|c_i)$$

Where the second equality is from the partition theorem. We also have:

1. $p(c_i)$ is the mixture coefficients,
2. $p(\underline{x}|c_i)$ is the multivariate Gaussian and it is the probability of a point given that it is in a particular Gaussian cluster c_i.

This is partially implemented by the helper function, get_predictions, which returns an array where for each input point $\underline{x}$ each element of the array is the probability of the point given that it is in a cluster (I.e., $p(\underline{x}|c_i)$). So, to get the probability of that point in the model, we have to sum over all of these to get $p(\underline{x})$.

So, to do a classification between phonemes 1 and 2, we have to take the clustering models for phonemes 1 and 2, and compute the probability of each point being in each clustering model by the equation given above. For example, for phoneme 1, we compute $p(\underline{x}|c_i)$ for each cluster and then sum them by the weights $p(c_i)$. We do the same for phoneme 2. Suppose we get:

1. $p_1(\underline{x})$
2. $p_2(\underline{x})$

For phonemes 1 and 2 respectively. Then, we compare the probabilities. The classifier assigns 1 to phoneme 1 and 0 to phoneme 2 and we get:

$$\hat{y} = \begin{cases} 1, & p_1(\underline{x}) > p_2(\underline{x}) \\ 0, & p_2(\underline{x}) > p_1(\underline{x}) \end{cases}$$

4 Q6

K	Accuracy
3	95.06578947368422
6	95.72368421052632

From the table above, we see that the K=6 classifier model has higher accuracy and as such, it is a better classifier.

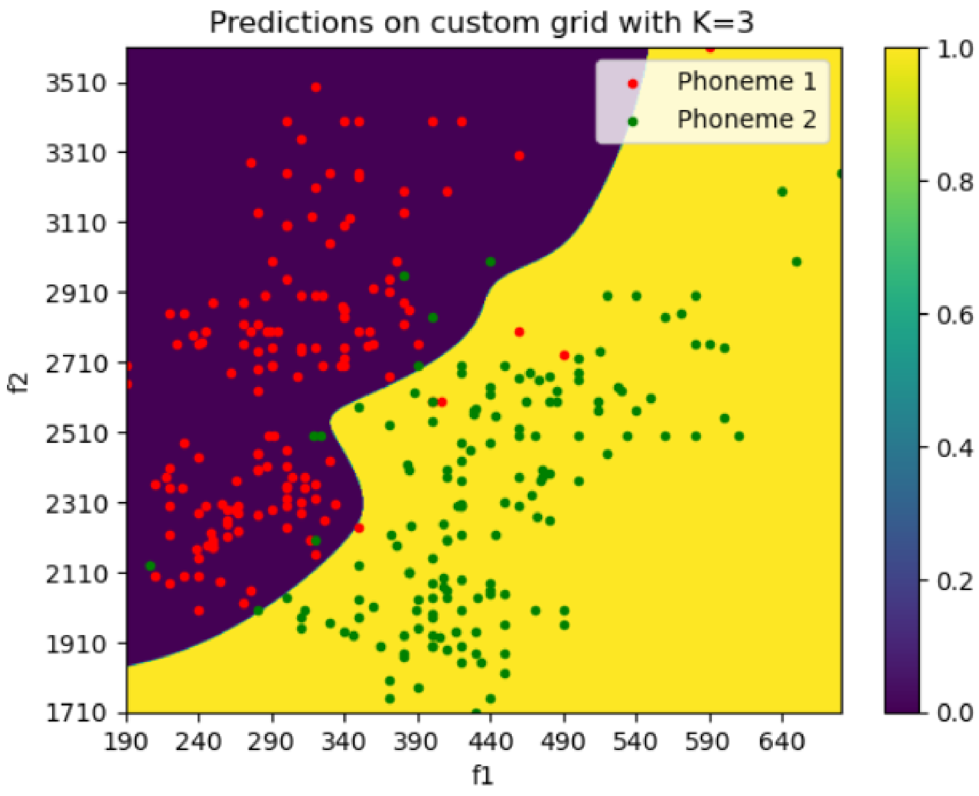


Figure 7: Prediction grid with K=3.

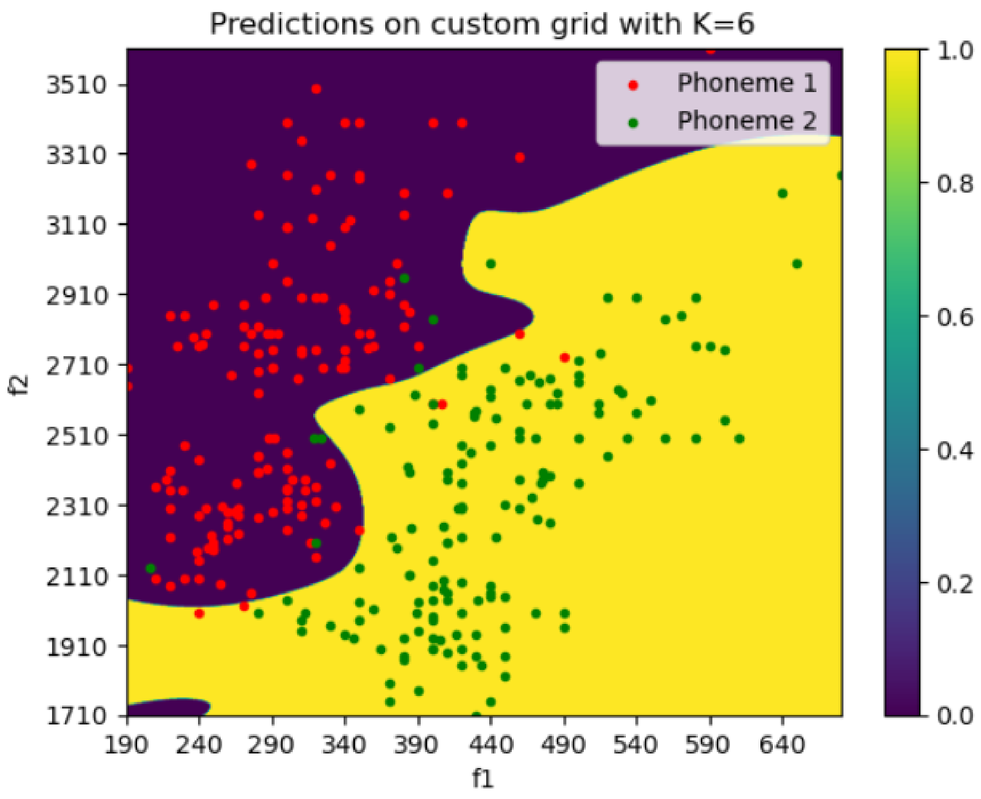


Figure 8: Prediction grid with K=6.

5 Q7

Comparing Figure 7 and Figure 8, we see that in Figure 7, the boundary is a lot smoother and perturbs less. The perturbation of the K=6 is due to the model being too strong and it trying to correctly identify points on the wrong side of the decision boundary, in other words, the model is too strong when k=6, and as such it may not generalize as well to unseen data. For example, in the bottom left of the figure for K=6, the model classifier the region around f1:[190,245], f2:[1710,1730] as phoneme 1 but identifies everything above it (same f1 but f2:[1730,2100]) as phoneme 2. This doesn't look contiguous and probably won't generalize well to unseen data around that region

6 Q8

Trying to first an MoG model on this data generally results in an error with sometimes the code running to completion. This error is due to the Gaussian model having an infinitesimally small width in some directions as the model over-fits the data. This is due to the fact that the covariance matrix has a determinant of 0 which makes it non-invertible and as such we get an due to a division by 0.

Density Estimation:

$$p(\underline{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp\{\frac{-1}{2}(\underline{x} - \mu)^T \Sigma^{-1}(\underline{x} - \mu)\}$$

where Σ is the covariance matrix. So, if the determinant of Σ is 0 (also meaning it is singular) then we have a division by 0 (due to $|\Sigma|$ on the denominator).

7 Q9

The way of overcoming the singularity problem is by putting a constraint on the covariance matrix to ensure that it is invertible.

We have learned 2 ways of doing this:

1. $\Sigma_0 = \sigma I$, where $\sigma^2 = \frac{1}{N}\sum_n (x_n - \hat{\mu})^T (x_n - \hat{\mu})$
2. $\Sigma_0 = [\sigma_1 ... \sigma_d]I$, where $\sigma_j^2 = \frac{1}{N}\sum_n (x_n(j) - \hat{\mu}(j))^2$

So to avoid over-fitting, we regularise the covariance matrix by adding one of the constraints above to it, which gives a new covariance matrix of: $\hat{\Sigma} = \Sigma + \Sigma_0$

By doing this, we simply add a diagonal element to our covariance matrix to ensure that it remains invertible.

Figure 9 shows a possible clustering where I have used the former method stated above to put a constraint on the covariance matrix. This results in the Models being more concentric. Whereas if I chose the latter method, I would've resulted in models being more elliptical w.r.t. the axes.

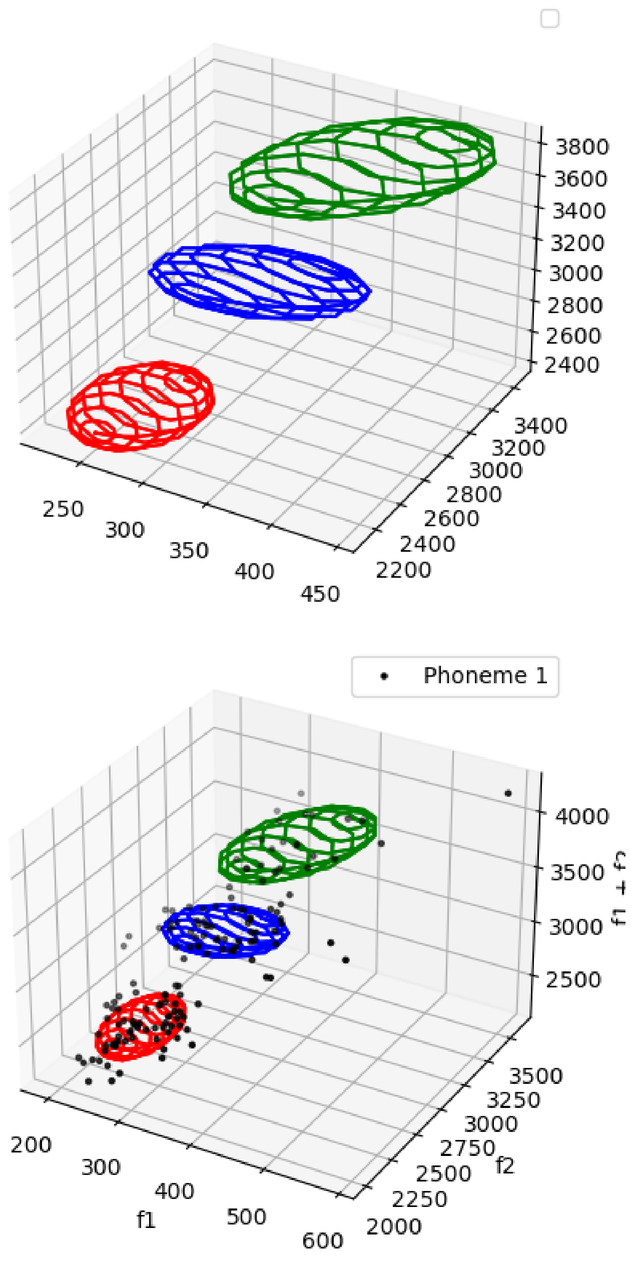


Figure 9: A possible cluster that the model converged to with a regularization.