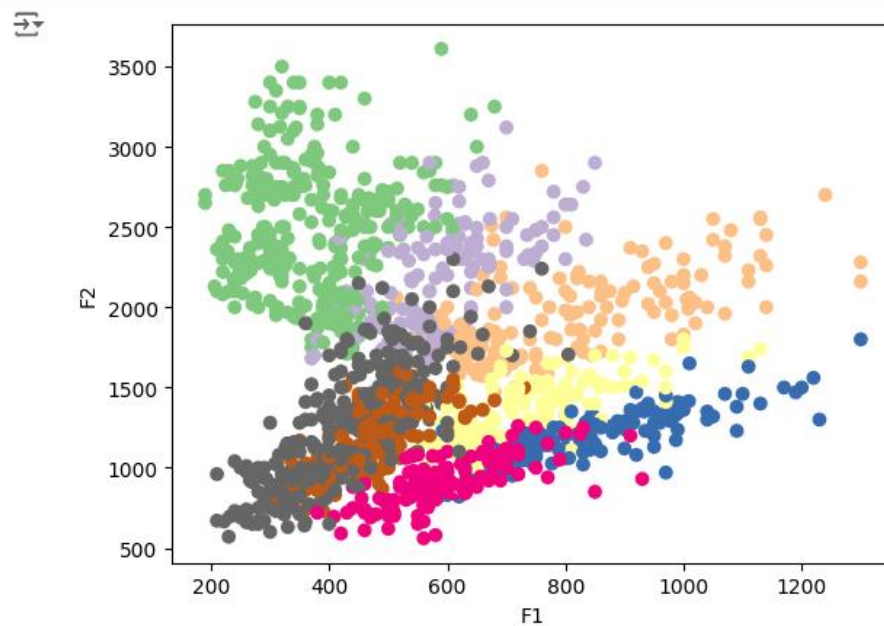


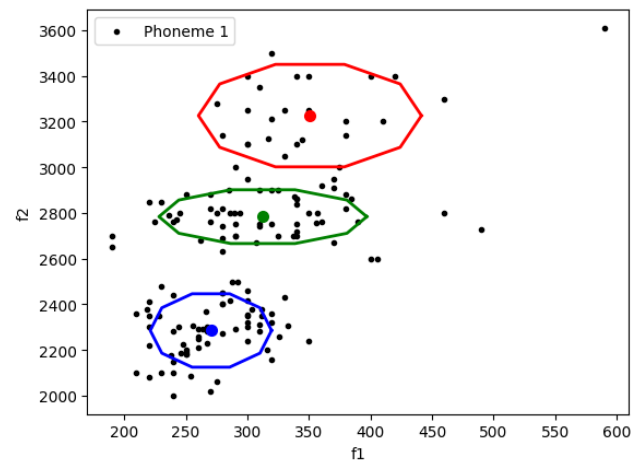
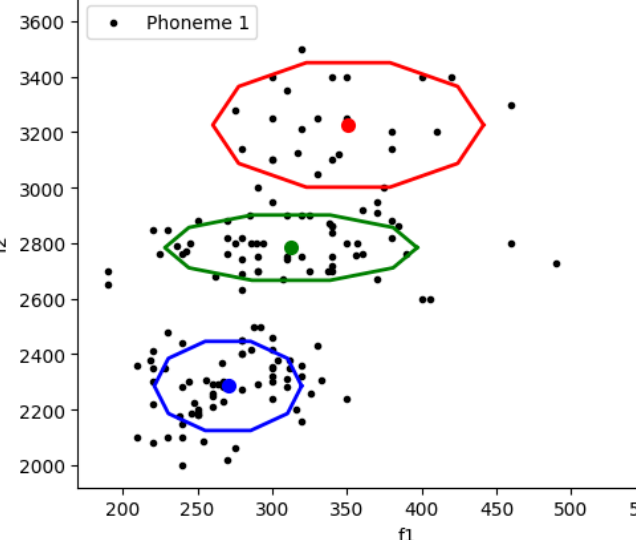
Q1)



The graph above is showing relationship between F1 and F2, where the points are represented by phonemes with the color codes matching phoneme_id (there are 10). We can see that the data forms distinguishable clusters, corresponding to the phonemes. While there is some overlap between clusters, many are visually separable and exhibit a non-linear distribution. The figures clustering suggests that F1 and F2 are key distinguishing features for phoneme classification. Although the clusters are not perfectly linearly separable, the general patterns indicate the potential for successful classification using a supervised learning model with non-linear capabilities.

Q2. Below we run the code multiple times for $k=3$

Run	Output	Graph
1	<pre>[[323.24017 2903.5635] [244.30173 2231.018] [302.26318 2338.8552]] [[[4002.5369457 0.] [0. 67340.67891838]] [[356.12435852 0.] [0. 13577.91929416]] [[365.77104812 0.] [0. 6568.98447513]]]</pre>	
2	<pre>[[312.59125 2783.898] [270.3952 2285.4653] [350.8446 3226.3394]] [[[3562.59744053 0.] [0. 7657.84879584]] [[1213.73843487 0.] [0. 14278.420327]]</pre> <pre>[[4102.87534392 0.] [0. 27829.54405226]]]</pre>	

3	<pre>[[350.8446 3226.3394] [312.59125 2783.898] [270.3952 2285.4653]][[4102.875375 0.] [0. 27829.5422143]] [[3562.59743765 0.] [0. 7657.84897245]] [[1213.73843494 0.] [0. 14278.4202995]]]</pre>	
4	<pre>[[350.84412 3226.3245] [312.59064 2783.8953] [270.3952 2285.4653]][[4102.80063305 0.] [0. 27833.36844834]] [[3562.60704942 0.] [0. 7657.47995897]] [[1213.73831533 0.] [0. 14278.42392768]]]</pre>	

Question 2 continued in the next page

After running the training code for many iterations for value of $K=3$, the model consistently identified 3 distinct clusters in the data. However, the specific placement of the Gaussian components and their bounding areas varied between runs. The reason it is not same in each of the runs is due to randomness in the initialization of the Gaussian parameters.

Over the course of each run the clusters they sometimes “swapped their positions”, for example in run 1 the “red” larger cluster is the top one but in run 2 it is in the middle. This is because of the random initialization we mentioned before but the solutions gave the same overall structure observed in the graphs. This is because the em algorithm converged to reasonable solutions and shows how robust the model is despite random initialization.

The means μ differed across runs due to the random starting points. For example:

- In Run 1, the mean of the red Gaussian was $[323.24, 2903.56]$, while in Run 3, it shifted to $[350.84, 3226.34]$.
- These values increased in run 3 and run 4 our outputs $[[350.8446 \ 3226.3394] [312.59125 \ 2783.898] [270.3952 \ 2285.4653]]$ and $[[350.84412 \ 3226.3245] [312.59064 \ 2783.8953] [270.3952 \ 2285.4653]]$ respectively, these values are very similar and the model seems to be converging at these values.

Q5)

In order to implement the Maximum Likelihood (ML) classifier, we first loaded our pre-trained Gaussian Mixture Models parameters for $K=3$ clusters. These parameters were stored in two files: one for `phoneme_id=1` and another for `phoneme_id=2`. Each file contained the covariance, means and weights for our GMMs.

Next, we used the `get_predictions` function to calculate the likelihood of each data point under the two GMMs. This function computes the Gaussian probability density values for all data points in the dataset, which allowed us to estimate the likelihood that a given point belongs to each phoneme.

For each data point, we summated the likelihood values across the $K=3$ components for both models. This gave us two overall likelihood values: $p(x; \Theta_1)$ for `phoneme_id=1` and $p(x; \Theta_2)$ for `phoneme_id=2`.

To classify each point, we compared the two likelihoods and assigned the data point to the phoneme with the higher likelihood.

Finally, we compared the predicted classifications with the actual labels to calculate the classification accuracy. Using the pre-trained models for $K=3$, we achieved an accuracy of approximately 95.07%.

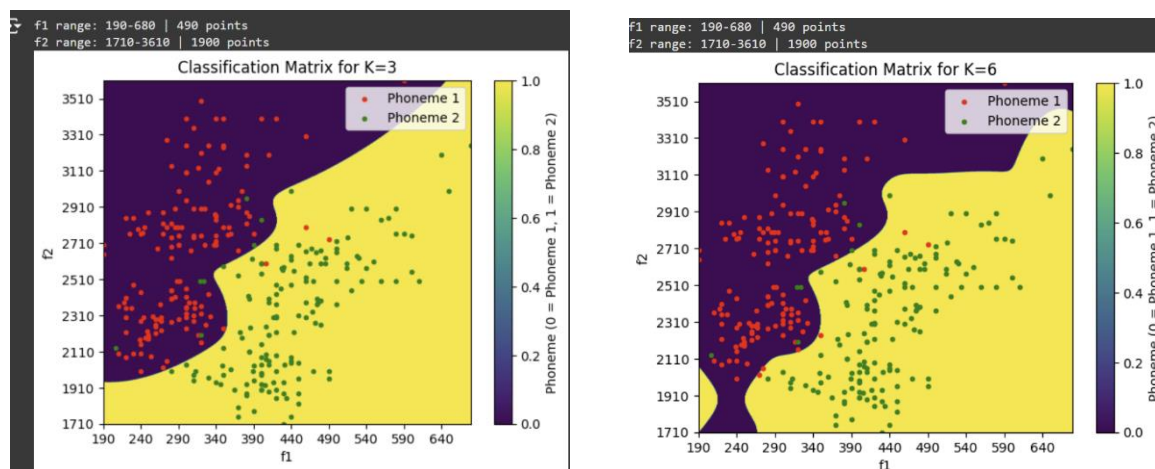
To conclude, the 95.07 % value indicates that the GMMs are effective in distinguishing between the two phonemes based on their formant frequencies.

Q6)

When running the model with $K=6$, our accuracy increases slightly to approximately 95.72%, compared to the 95.07% accuracy achieved with $K=3$. While the 0.65% improvement is not substantial, it shows that increasing the number of components allows the model to better capture the underlying structure of the data, leading to more accurate classification.

In conclusion, the improvement demonstrates that a larger number of components allows for slightly more detail in modeling the data distributions, which can result in reduced overlap between the clusters for different phonemes. However, because the increase in accuracy was only 0.65% this suggests that the model with $K=3$ was already sufficient to capture most of the variability in the data.

Q7)



From the classification matrices for $K=3$ and $K=6$, we observe that the decision boundary becomes more refined and complex as K increases. For $K=3$, the boundary is relatively smoother, but a few points are misclassified, especially near overlapping regions. In contrast, for $K=6$, the increased complexity of the Gaussian Mixtures allows for a more intricate decision boundary, capturing finer details of the data distribution. However, this increased complexity may lead to overfitting in some cases, as the model tries to adapt tightly to the training data.

Overall, the classification accuracy improves slightly with $K=6$.

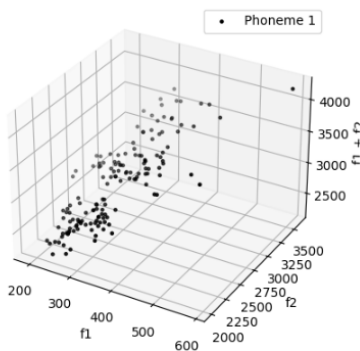
Q8

```

Iteration 001/150
Iteration 002/150
Iteration 003/150
<ipython-input-9-5c11a34e070c>:26: RuntimeWarning: divide by zero encountered in scalar divide
  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))
-----
ValueError                                Traceback (most recent call last)
<ipython-input-23-16aeb341fc2f> in <cell line: 47>()
     51 # Do the E-step
     52 Z = get_predictions(mu, s, p, X)
--> 53 Z = normalize(Z, axis=1, norm='l1')
     54 # Do the M-step:
     55 for i in range(k):

4 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite_element_wise(X, xp, allow_nan, msg_dtype, estimator_name, input_name)
    170     "#estimators-that-handle-nan-values"
    171 )
--> 172     raise ValueError(msg_err)
    173
    174
ValueError: Input contains infinity or a value too large for dtype('float64').

```



Output when running the code for Question 8

Trying to fit the MoG model to the data arranged in the format $X=[f1, f2, f1+f2]$ causes a singularity issue because the third feature ($f1+f2$) is a linear combination of the first two features ($f1, f2$). This introduces perfect collinearity in the dataset, making the covariance matrix singular or nearly singular. During the E-step of the Expectation-Maximization algorithm, this results in a zero determinant for the covariance matrix, causing division by zero and numerical instability.

As a result, our model fails to compute probabilities correctly, leading to incorrect classifications and preventing the model from functioning as intended. This issue highlights the importance of ensuring feature independence when applying Gaussian Mixture Models.

Q9:

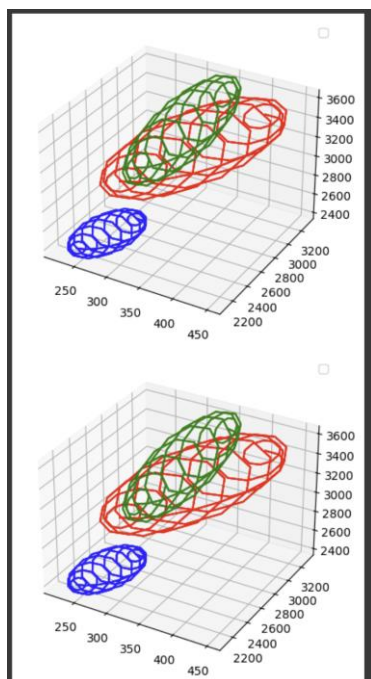
To overcome the singularity problem from the previous question we can Regularize the covariance matrix by adding a small value to its diagonal to prevent singularity. This regularization constraint will help prevent the error in our code from the previous question.

The formula used:

$$\sigma_j^2 = \frac{1}{N} \sum_n (x_n(j) - \hat{\mu}(j))^2$$

Source: https://qmulplus.qmul.ac.uk/pluginfile.php/2344378/mod_resource/content/3/08-DensityEstimation-2020.pdf

The output to the code for Question 9:



The training outputs generated Gaussian ellipsoids for the 3 clusters (red, green, and blue) and they were appropriately scaled and oriented. No unexpected artifacts, such as infinity values or degenerate shapes, appeared. Therefore, the regularization successfully prevented singularity, allowing the EM algorithm to converge without numerical instabilities.

