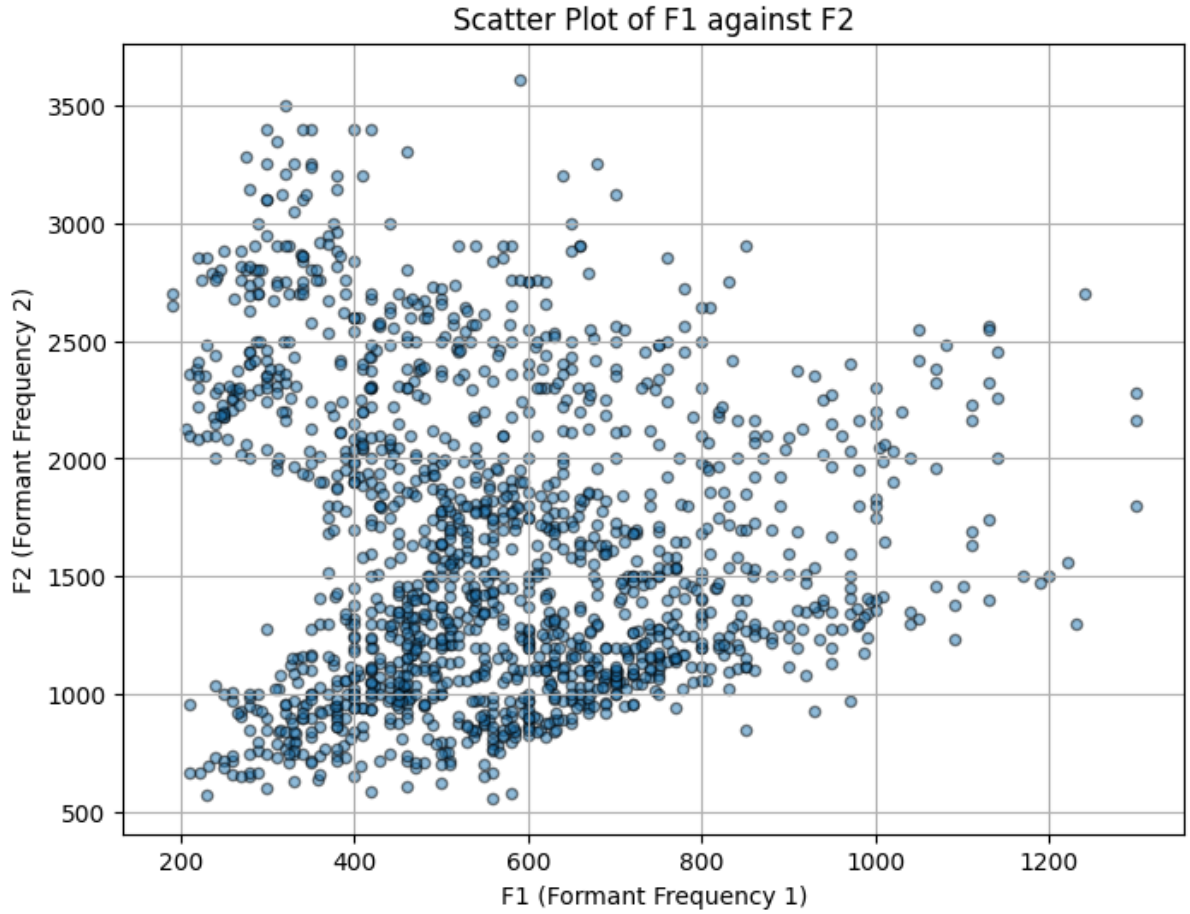
REPORT

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]

Ans:

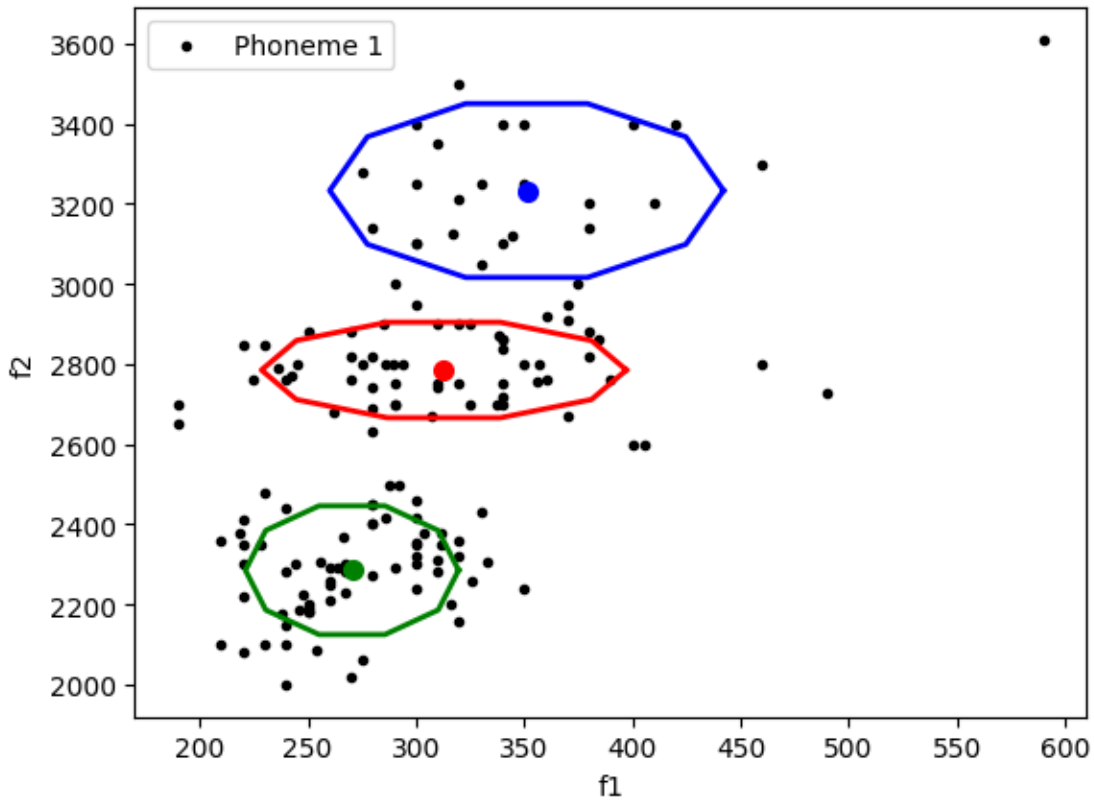


The plot of F1 against F2 shows that there is a non-uniform distribution with the points in the scatter plot which suggests the presence of several clusters. There are denser regions where the points are closely packed and these regions suggest that the data might contain distinct groups of phonemes that show similar formant characteristics. In certain clusters, as F1 increases F2 decreases this shows that in some clusters there is an inverse relationship between F1 and F2. The points are more densely clustered around lower values of F1(between 300 and 600) and F2(between 800 and 2000). We can see clusters forming at low F1 and medium F2 as well as medium F1 and F2 potentially representing distinct types of phonemes. The plot suggests the presence of natural groupings of phonemes, which could be further analysed with clustering algorithms such as k-means.

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]

Ans: Each run of the code produces slightly different Gaussian components (means, covariances, and weights) due to the random initialization of the means. This highlights the sensitivity of the EM

algorithm to initial conditions. Since the EM algorithm uses a greedy approach to maximize the likelihood, it converges to a local optimum. Different initializations may lead to different solutions, i.e., the placement and size of Gaussian clusters can vary between runs. While the exact clusters differ, the algorithm typically identifies similar general groupings in the data.



[[312.87854 2785.311]

[270.39297 2285.4417]

[351.10776 3233.4675]] [[[3555.66316423 0.]

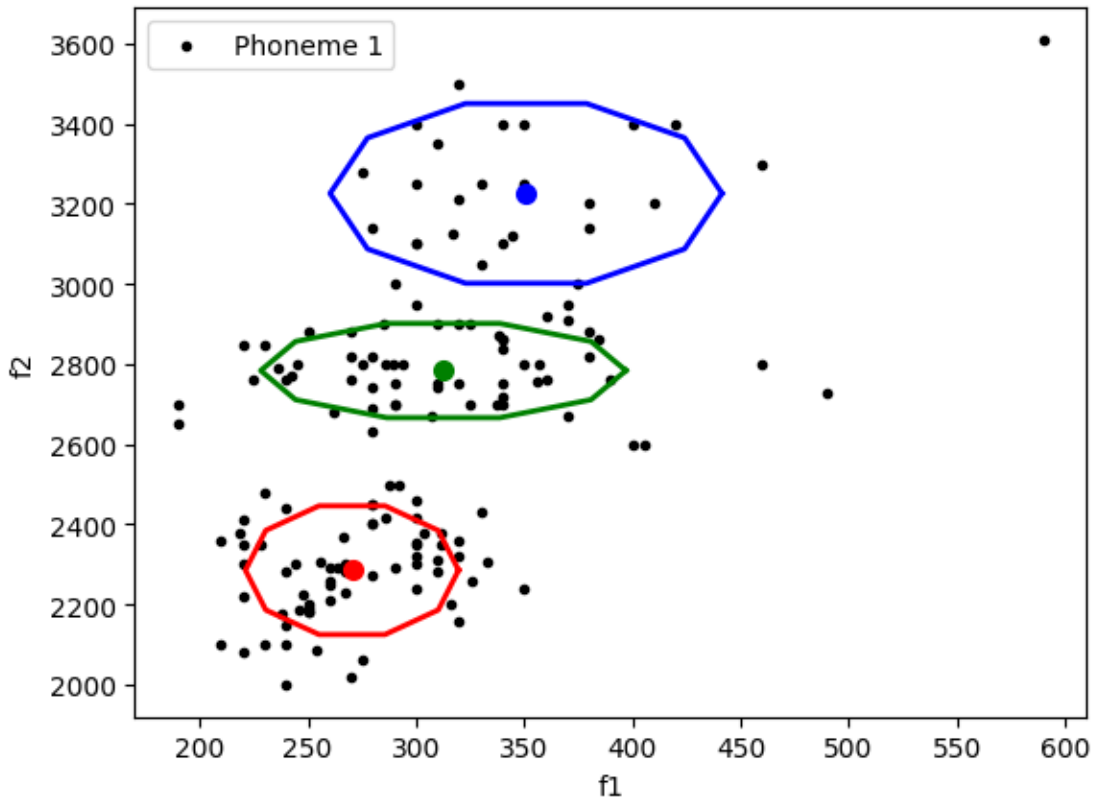
[0. 7863.05638527]]

[[1213.81208685 0.]

[0. 14274.92946472]]

[[4142.92466338 0.]

[0. 26043.27122315]]]



[[

270.3952 2285.4653]

[312.59125 2783.898]

[350.8446 3226.3394]] [[[1213.73843494   0.   ]

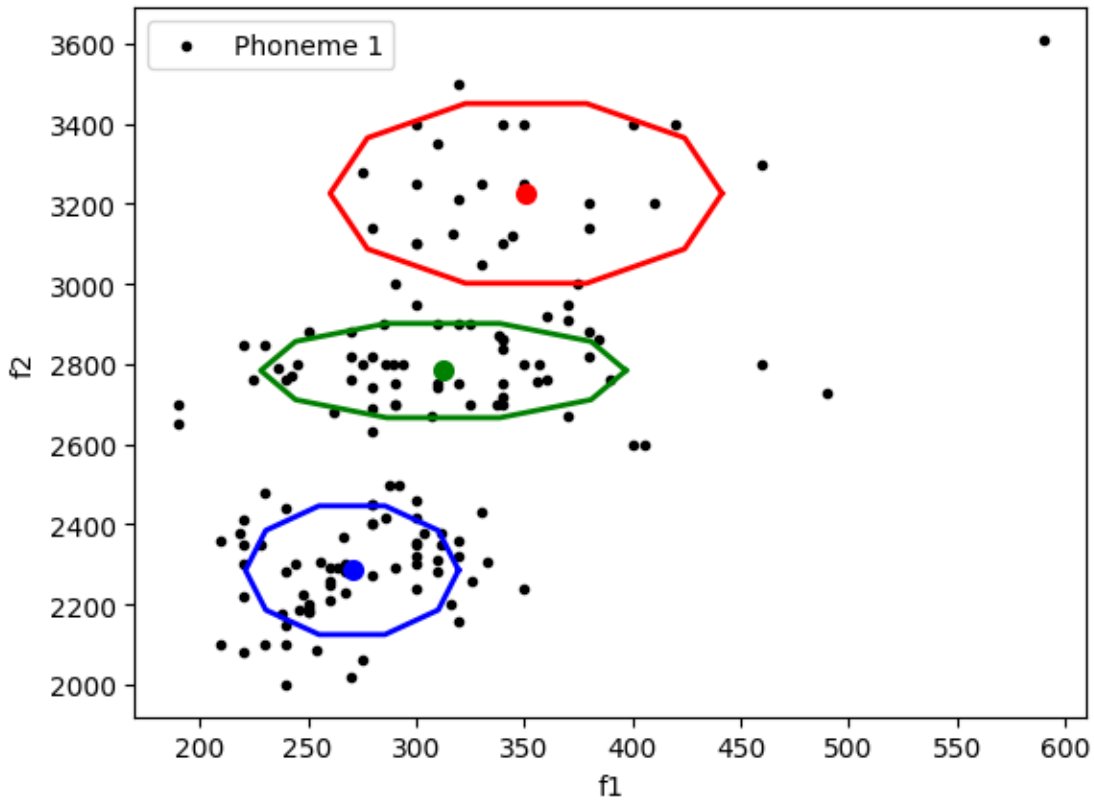
[0.   14278.4202995]]

[[3562.59743765   0.   ]

[0.   7657.84897245]]

[[4102.875375   0.   ]

[0.   27829.54221422]]]



[[

350.8446 3226.3394]

[312.59125 2783.898]

[270.3952 2285.4653]] [[[4102.8753751   0.   ]

[0.   27829.5422082]]

[[3562.59743764   0.   ]

[0.   7657.84897304]]

[[1213.73843494   0.   ]

[0.   14278.42029941]]]

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]

Ans: For each phoneme (1 and 2), a Gaussian Mixture Model (GMM) with K=3 components was trained separately using the Expectation-Maximization (EM) algorithm.

To classify each data point, the Maximum Likelihood criterion was used. The likelihood of each data point being generated by either phoneme 1 or phoneme 2 was computed. If the likelihood for phoneme 1 was greater than that for phoneme 2 the point was classified as phoneme 1. Otherwise, it was classified as phoneme 2.

The log-likelihood for each data point was calculated by summing the probabilities across the Gaussian components and then taking the logarithm. Using the log-sum of likelihoods helps avoid numerical issues that can arise when dealing with very small probability values.

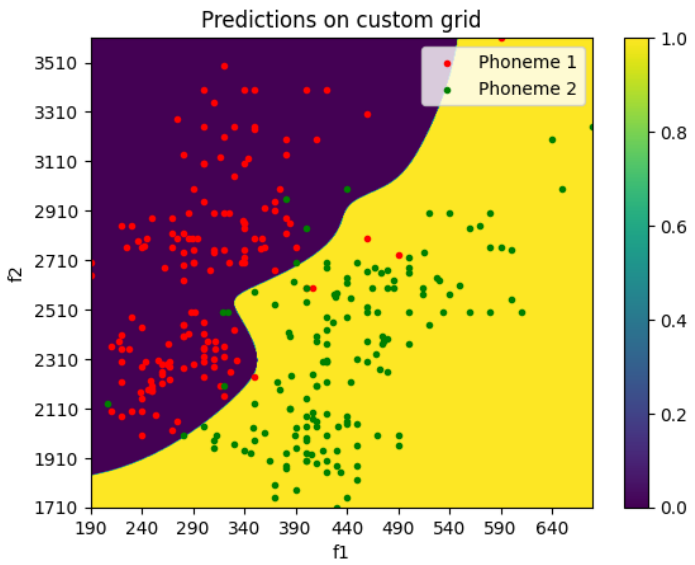
Finally, the accuracy and misclassification error was calculated by comparing the predicted labels with the ground truth labels. A high accuracy indicates that the two phonemes are well separated in the feature space defined by F1 and F2.

Q6. Repeat for K=6 and compare the results in terms of accuracy. [2 mark]

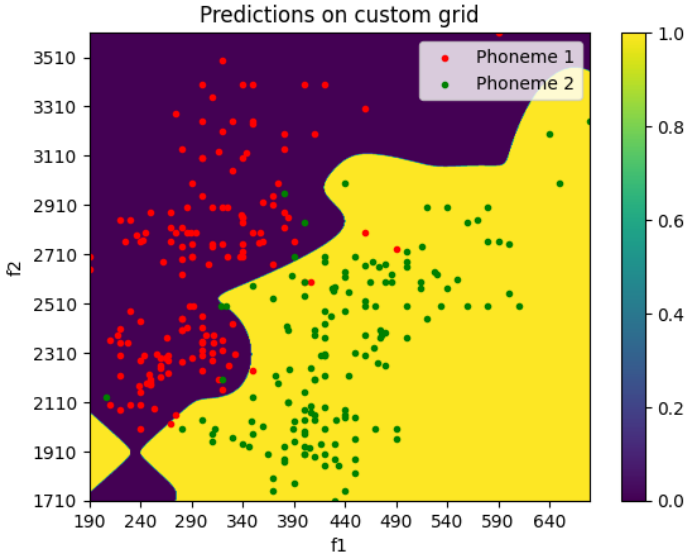
Ans: The classification accuracy increases noticeably from 95.07% to 96.05% when the number of Gaussian components is increased from K=3 to K=6. This rise suggests that the more intricate model can more accurately classify the two phonemes by better representing their underlying distributions. K=6 enables the model to capture the finer structure in the data by representing each phoneme with six Gaussian components. Because the model is better able to represent sub-clusters within each phoneme, the increase in accuracy indicates that the additional components offer better discrimination between phonemes 1 and 2. In conclusion, adding more components makes the classification model more reliable and accurate and better reflects the data's underlying structure.

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]

Ans:



For K=3



For K=6

The classification matrices for K=3 and K=6 GMM classifiers illustrate how the number of Gaussian components affects decision boundaries for phoneme classification.

With K=3, the decision boundary is smooth and generalized, reflecting a simpler model that broadly separates the two phonemes. The purple and yellow regions are well-defined but lack complexity, which may struggle to capture overlapping areas of the data effectively.

In contrast, K=6 produces a more intricate boundary, adapting better to the data's finer structure. This increased complexity allows the model to identify sub-clusters, providing a more precise separation between phonemes. However, the added flexibility introduces a risk of overfitting.

Overall, K=3 offers a more generalized approach, while K=6 provides better discrimination at the cost of increased model complexity, highlighting the trade-off between accuracy and overfitting.

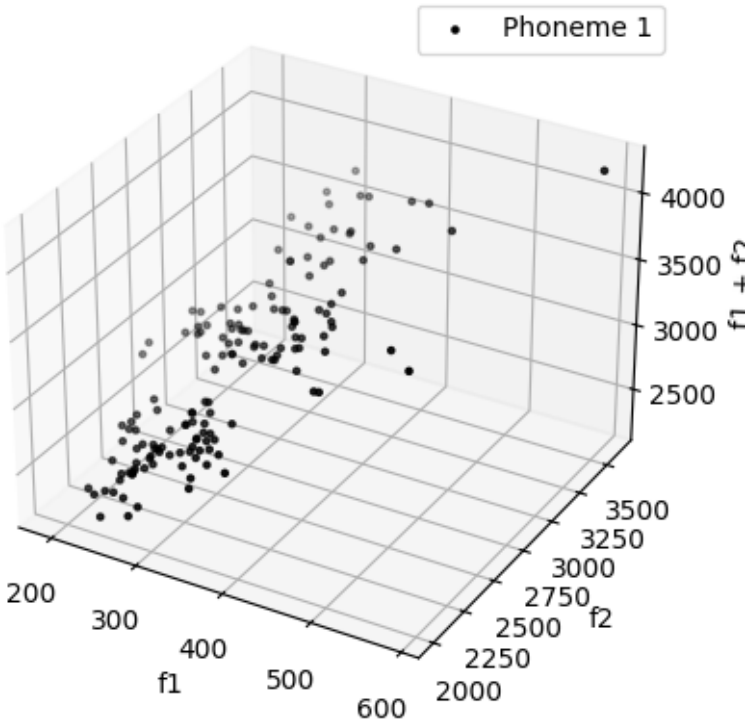
Q8. Try to fit a MoG model to the new data. What is the problem that you observe? Explain why it occurs [2 marks]

Ans: The problem with fitting the Mixture of Gaussians (MoG) model arises due to singular covariance matrices, which cause numerical instability during the E-step of the Expectation-Maximization algorithm. This issue is indicated by the ValueError ("inputs contain infinity or a value too large for the dtype") that occurs during the normalization step within the EM algorithm.

The covariance matrix becomes singular because the dataset includes a linearly dependent feature (F1+F2), resulting in zero or very small determinants that cannot be properly inverted. This causes instability when calculating probabilities.

To resolve this, we should consider adding regularization to the covariance matrices, removing redundant features, or increasing data variability to prevent singularity and stabilize the model fitting process.

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

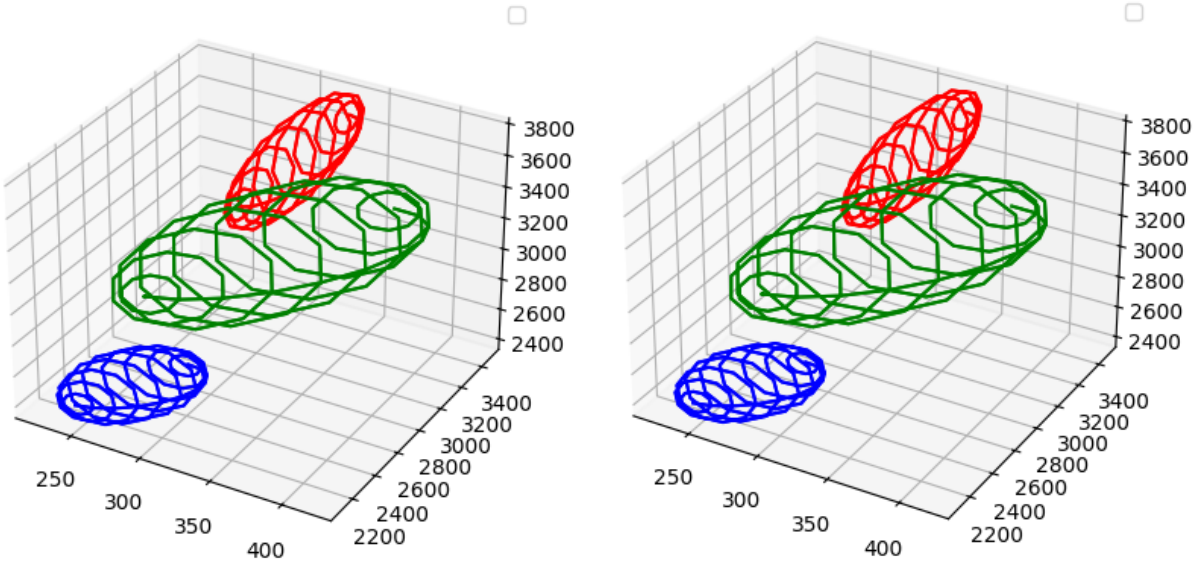


Q9. Suggest ways of overcoming the singularity problem and implement one of them. Show any training outputs in the report and discuss. [3 marks]

Ans: Ways to Overcome the Singularity Problem

- Regularization: Add a small value to the diagonal elements of each covariance matrix. This ensures that the covariance matrix remains positive and definite to avoid numerical issues during inversion.
- Use a Diagonal Covariance Matrix: Instead of a full covariance matrix, use only the diagonal elements. This reduces the complexity and prevents singularity, especially when features are linearly dependent.
- Data Preprocessing: Normalize or standardize the data to ensure that all features have similar scales. This helps to avoid large discrepancies in the range of feature values, which can lead to singular covariance matrices.
- Remove Collinear Features: If features are collinear, it may result in singularity. Removing highly correlated features or using dimensionality reduction techniques (e.g., PCA) can prevent singular matrices.

To overcome the singularity problem, a regularization term was added to the diagonal of each covariance matrix during both initialization and updates in the M-step. This ensures that the covariance matrices are always invertible, thus mitigating the singularity problem.



The figures show the Gaussian components after running the Expectation-Maximization (EM) algorithm code multiple times for K=3 components. The ellipses representing the components are better defined compared to previous attempts without regularization, indicating improved numerical stability and reliable clustering.

By adding a small positive value (1e−4) to the diagonal, the model maintained invertibility, which allowed for better estimation of parameters during the E-step and M-step. The resulting model, shown in the plots, successfully fits Gaussian components to the data, capturing different clusters more accurately compared to the initial implementation without regularization where the code caused an error.