Name: Tianxiu Ma

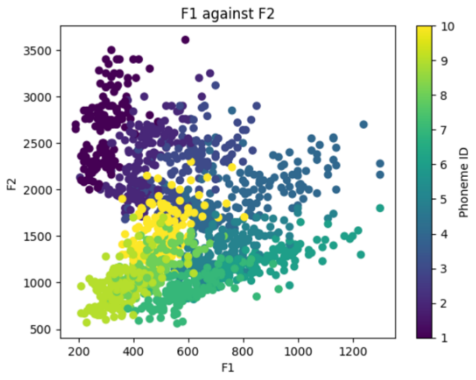
Student number: 230277302

Assignment number: Assignment 2

module code: ECS708U/ECS708P

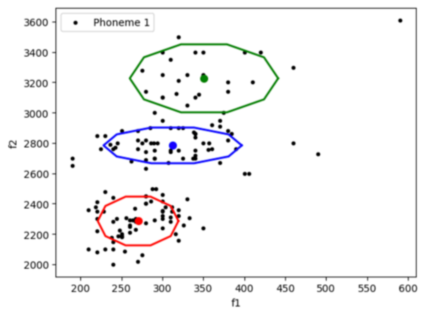
Assignment 2: Unsupervised Learning

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]



Points with the same color (phoneme) tend to concentrate in certain areas, suggesting possible clusters or separations in the feature space. It indicates that the features F1 and F2 offer helpful information for differentiating between phonemes if there are discernible patterns.

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]



MoG parameters | Mean values:

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

MoG parameters | Covariances:

```
[[[ 1213.73843438         0.            ]
  [       0.            14278.4204861 ]]
 [[ 4102.87511804         0.            ]
  [       0.            27829.55808178]]
 [[ 3562.59745624         0.            ]
  [       0.             7657.84757031]]]
```

MoG parameters | Weights:
[0.43514449 0.18387822 0.38097729]

The ellipses represent the clusters based on the mean values and covariance matrices and the scatter points are sampled from the Gaussian distributions defined by the MoG parameters.

Observations:
(1) **Varying Cluster Shapes:** Each cluster's shape is shown by the covariance matrices in the MoG parameters. The algorithm may converge to solutions with different cluster shapes in different runs, producing ellipsoidal clusters with various sizes and orientations.
(2) **Varying Cluster Positions:** The mean values represent the centers of the clusters. Different runs may lead to convergence with clusters located at different positions in the feature space.
(3) **Varying Cluster Weights:** The weights are not uniform, indicating that some clusters are more prevalent than others. The cluster with the highest weight (0.435) is expected to have a more significant impact on the overall distribution.

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]

The code process:
(1) Fill X with the samples of X_full that belong to the chosen phonemes.
(2) Load the pretrained GMM model for k = 3.
(3) Make a copy of the X to get the samples for p_id.
(4) Get the predictions (likelihood) for both phonemes. Sum up the predictions and save in pred_i, i = 1|2.
(5) Repeat steps 2-4 for the second phoneme.
(6) Initialise the predictions array that holds the cumulative likelihood from the individually summed-up predictions from step 4. Use the information to calculate the accuracy %.

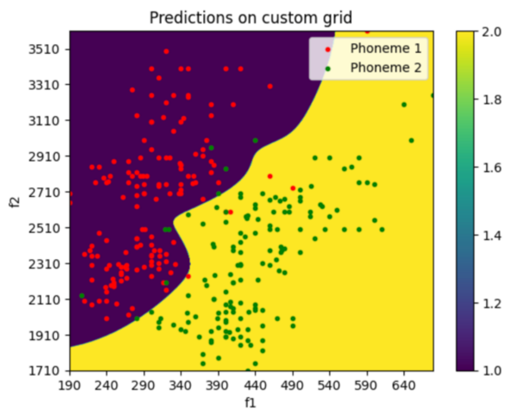
Q6. Repeat for K=6 and compare the results in terms of accuracy. [2 mark]

K=3: `Accuracy: 96.38157894736842`
K=6: `Accuracy: 96.05263157894737`
Discussion:
(1) The decision on the number of components (K) is crucial in GMM clustering.
(2) A higher K provides a more detailed clustering but may lead to overfitting and increased computational complexity.
(3) The results indicate that, in this particular dataset, both K=3 and K=6 configurations yielded high accuracies, suggesting that the underlying data structure can be well-captured by multiple clusters.

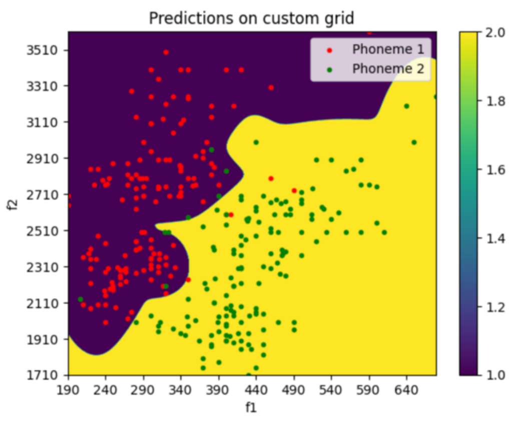
In summary, the Gaussian Mixture Model (GMM) trained with three parameters demonstrated accelerated training and improved accuracy. Conversely, the GMM with six parameters exhibited decreased accuracy attributed to overfitting and required a longer training time.

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]

K=3:



K=6:



The implement process is structured as:

(1) Fill X with the samples of X_full that belong to the chosen phonemes.

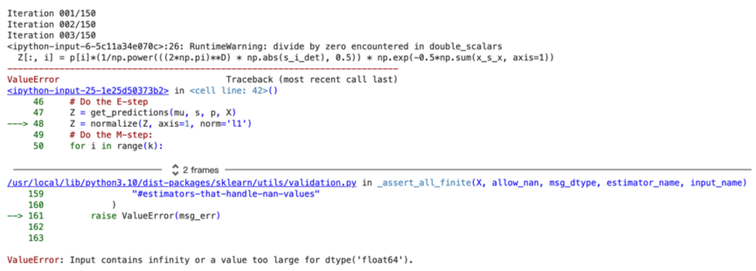
(2) Create the grid using linearly spaced vectors.

(3) Same as the process explained in Question 5.

(4) Create M array containing 0 in the points that belong to phoneme 1, and 1 in the points that belong to phoneme 2. M forms the background for the confusion matrices in the pictures above.

(5) Plot the confusion matrix after sorting the values for the color assignment.

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



```
Iteration 001/150
Iteration 002/150
Iteration 003/150
<ipython-input-6-5c11a34e070c>:26: 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))
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-1e25d50373b2> in <cell line: 42>()
     46     # Do the E-step
     47     Z = get_predictions(mu, s, p, X)
---> 48     Z = normalize(Z, axis=1, norm='l1')
     49     # Do the M-step:
     50     for i in range(k):

                          2 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name, input_name)
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

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

The error arises from the normalization process encountering infinity or an excessively large

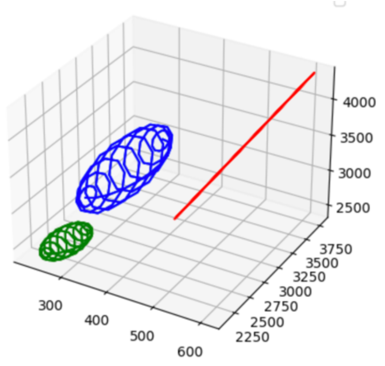
value. It specifically occurred during an attempt to use L1 normalization to normalize the array Z along axis 1 in the normalize function.
A possible division by zero problem is also indicated by the warning (RuntimeWarning: divide by zero encountered in double_scalars) during the Z calculation. This is probably the main reason for the ensuing error.
The issue arises from the fact that exponentiation and a division operation are used in the Expectation-Maximization (EM) algorithm's E-step computation. The error seen can arise from division by zero or extremely large values if any of the denominator's elements get too small (near zero).

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

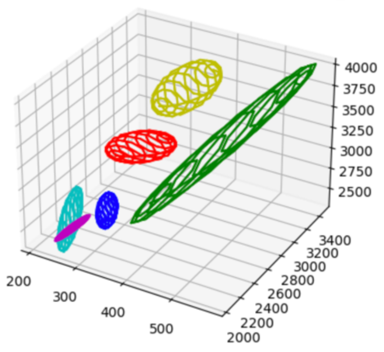
Ways of overcoming the singularity problem:

(1) Regularization: Add a small regularization term to the denominator to prevent it from becoming zero. For example, we can add a small constant to the denominator.
(2) Check for Zero Denominator: Before normalization, check for elements in the denominator that are close to zero and handle them separately to avoid division by zero.

The problem was fixed by implementing regularisation to the covariance matrices for this code. A diagonal matrix with equal dimensions is represented as an identity matrix and multiplied with a value of 0.001 to force the diagonal entries to never reach 0.
K=3:



K=6:



The interpretation of clusters heavily depends on the context and the characteristics of the data.