CS5691: Pattern Recognition and Machine Learning
Assignment #2

Topics: LDA, GMM, DBSCAN **Deadline:** 28 April 2023, 11:55 PM

Teammate 1: Aditya Viraj Rao Ponugoti **Roll number:** CS20B005
Teammate 2: Neel Prabhanjan Rachamalla **Roll number:** CS20B056

- **For any doubts regarding questions 1 and 2**, you can mail cs22s013@smail.iitm.ac.in and cs21s043@smail.iitm.ac.in
- **For any doubts regarding question 3**, you can mail cs21d015@smail.iitm.ac.in and cs22s015@smail.iitm.ac.in
- Please refer to the **Additional Resources** tab on the Course webpage for basic programming instructions.
- This assignment has to be completed in teams of 2. Collaborations outside the team are strictly prohibited.
- Any kind of plagiarism will be dealt with severely. These include copying text or code from any online sources. These will lead to disciplinary actions according to institute guidelines. Acknowledge any and every resource used.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- You should submit a zip file titled **'rollnumber1_rollnumber2.zip'** on Moodle where rollnumber1 and rollnumber2 are your institute roll numbers. Your assignment will **NOT** be graded if it does not contain all of the following:
    1. Type your solutions in the provided LaTeX template file and title this file as **'Report.pdf'**. **State your respective contributions in terms of percentage at the beginning of the report clearly.** Also, embed the result figures in your LaTeX solutions.
    2. Clearly name your source code for all the programs in **individual Google Colab files**. Please submit your code only as Google Colab file (.ipynb format). Also, embed the result figures in your Colab code files.
- We highly recommend using `Python 3.6+` and standard libraries like `NumPy, Matplotlib, Pandas, Seaborn`. Please use `Python 3.6+` as the only standard programming language to code your assignments. Please note: the TAs will only be able to assist you with doubts related to Python.
- You are expected to code all algorithms from scratch. **You cannot use standard inbuilt libraries for algorithms until and unless asked explicitly**.
- **Any graph that you plot is unacceptable for grading unless it labels the x-axis and y-axis clearly.**

- Please note that the TAs will **only** clarify doubts regarding problem statements. The TAs won't discuss any prospective solution or verify your solution or give hints.
- Please refer to the CS5691 PRML course handout for the late penalty instruction guidelines.

---

1. **[Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA)**] You will implement dimensionality reduction techniques (LDA, PCA) as part of this question for the dataset1 provided here.

   Note that you have to implement **LDA from scratch** without using any predefined libraries (i.e. sklearn, scipy) . However, you can use **predefined libraries to implement PCA.**

   (a) (2 marks) Use Linear Discriminant analysis (LDA) to convert dataset1 into the two-dimensional dataset and then visualize the obtained dataset. Also, perform an analysis on how results will change if we perform normalization (i.e., zero mean, unit variance normalization) on the initial dataset before applying LDA.

Solution:
Linear Discriminant Analaysis :-

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that focuses on maximizing class separability. It projects high-dimensional data onto a lower-dimensional space while preserving the discriminatory information. LDA achieves this by finding a set of orthogonal axes that maximize the ratio of between-class scatter to within-class scatter. It is commonly used for feature extraction and classification tasks, aiding in reducing computational complexity and enhancing classification accuracy. The calculations invovled in LDA are :-

Within Class Scatter Matrix :-

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

Between Class Scatter Matrix :-

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

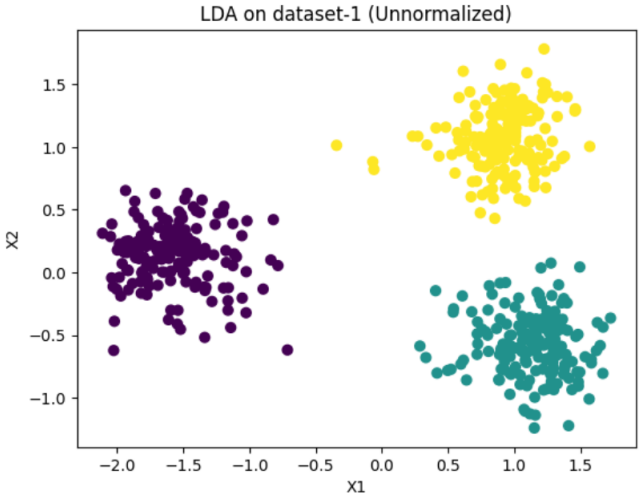
$$Y = W_{opt}^T X \quad \text{where} \quad W_{opt} = \arg\max \frac{|W^T S_B W|}{|W^T S_W W|}$$

Now, here we are given a 64-D data in dataset-1. We are to reduce this data into 2-D data using Linear Discriminant Analysis.

Please find the link to the google colab file here.

Results :-

Note that firstly we perform LDA on unnormalized data.

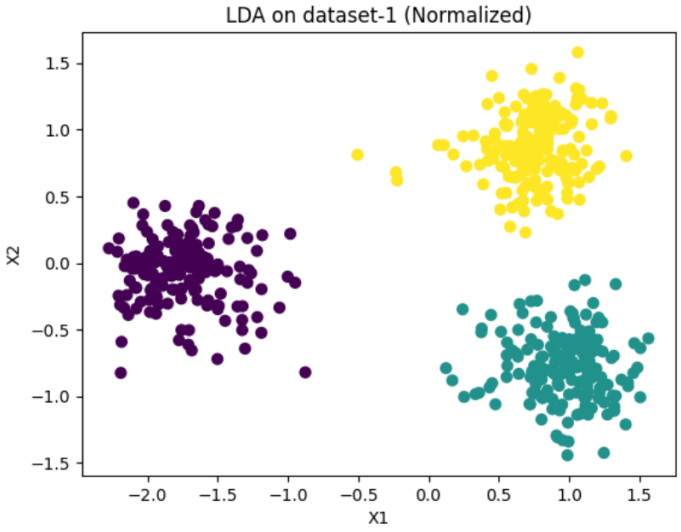


Now, we perform LDA on normalized data.

Here, the data of dataset-1 has variance zero for some features so while normalizing the data, we are normalizing the data in two versions:-

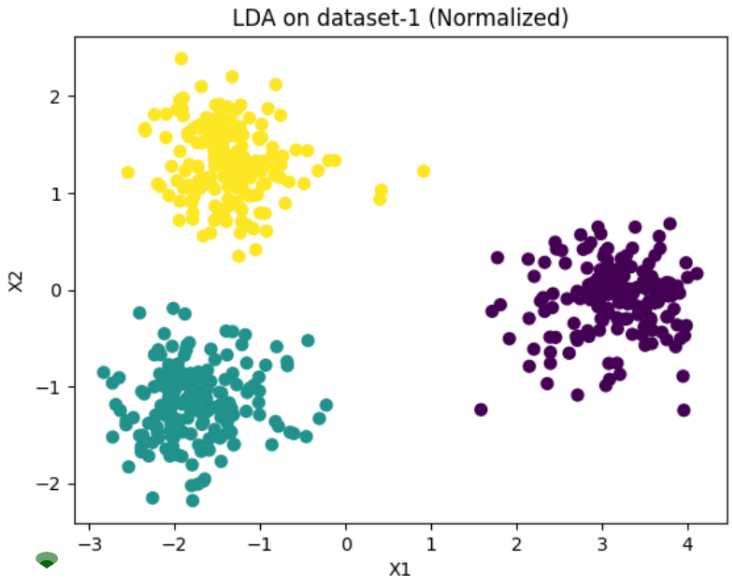
Version-1 :-

Here, we are just calculating the mean-subtracted data and not ensuring variance one, since it is zero for some features and may give a 'Division by zero' error.



Version-2 :-

Here, we are calculating the mean-subtracted data and ensuring variance one for non-zero variance features and setting the other zero variance features to zero.



Note :- Here, the scatter plot of version-2 might appear as the flipped version of version-1, this is due to choosing of eigen vectors which can be negative.So version-1 and version-2 are the same.

Obervations :-

LDA on both unnormalized and normalized data comes out to be the same. This is because the normalization process does not address the underlying differences in mean and variance between the classes.

(b) (1.5 marks) Use PCA to convert dataset1 into two-dimensional data and then visualize the obtained dataset. Now, compare and contrast the visualizations of the final datasets obtained using LDA and PCA.

Solution:
Principal Component Analaysis :-

Principal Component Analysis (PCA) is a dimensionality reduction technique. It identifies the principal components, which are orthogonal axes that capture the most significant variations in the data. By ranking and selecting the top components, PCA allows for dimensionality reduction while preserving the essential structure of the data. The calculations invovled in PCA are :-

Total Scatter Matrix :-

$$S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T$$

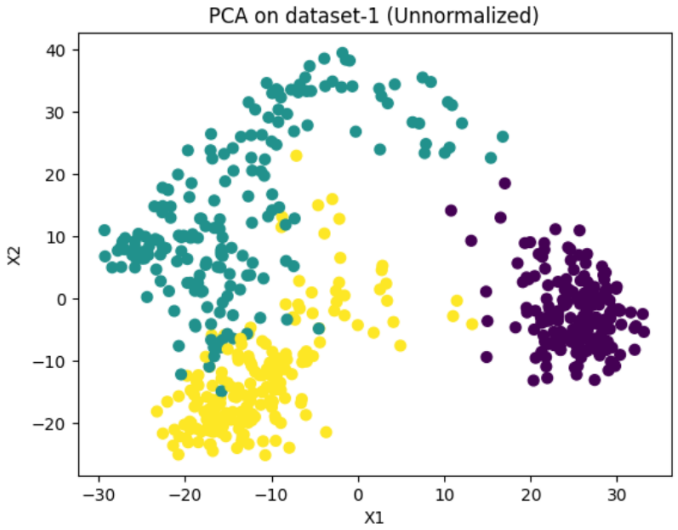
$$Y = W_{opt}^T X \quad \text{where} \quad W_{opt} = \arg\max |W^T S_T W|$$

Now, here we are given a 64-D data in dataset-1. We are to reduce this data into 2-D data using Principal Component Analysis.

Please find the link to the google colab file here.

Results :-

Now, we perform PCA on unnormalized dataset-1.



Obervations :-

We observe that the scatter plots obtained from LDA and PCA are quite different from each other.This is because:-

LDA maximizes the class separability by finding axes that maximize the ratio of between-class scatter to within-class scatter whereas PCA focuses on finding the maximum variance in the data by identifying orthogonal axes that explain the most significant variations.

LDA emphasizes class separation, resulting in a projection that maximizes the separation between different classes.In contrast, PCA focuses on overall data variance, and its projection may not prioritize class separability.

(c) (1.5 marks) Randomly shuffle and split the obtained dataset from part (a) into a training set (80%) and testing set (20%). Now build the Bayes classifier using the training set and report the following:

- Accuracy on both train and test data.
- Plot of the test data along with your classification boundary.
- confusion matrices on both train and test data.

Solution:

Naive Bayes Classifier Using Multivariate Gaussian Distribution :-

The D-dimensional Gaussian distribution is parameterised by a mean vector, $\mu = (\mu 1, ..., \mu D)^T$ and a DXD covariance matrix, $\Sigma = \sigma(ij)$. The probability density function is given by:

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

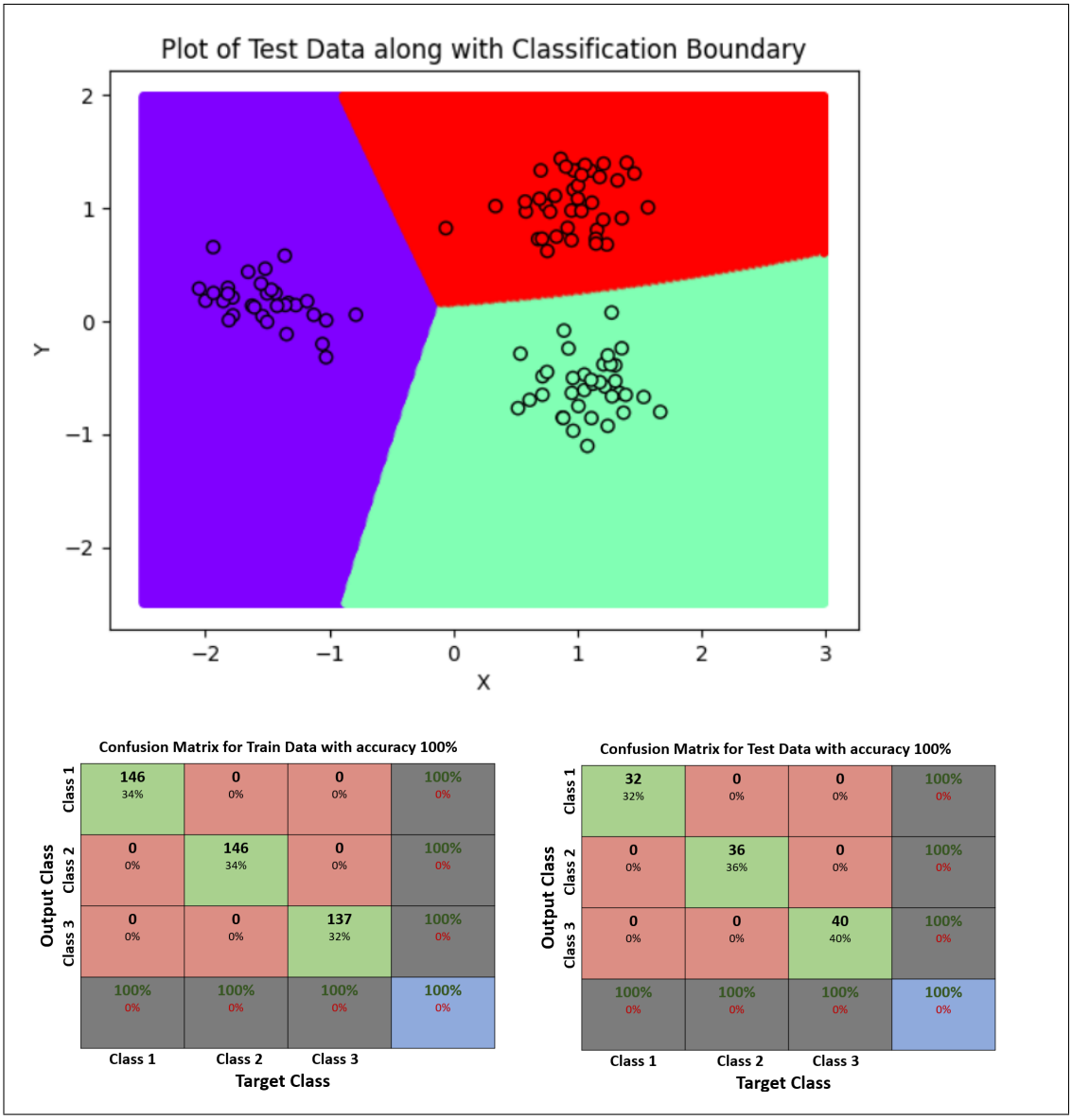
where $|\Sigma|$ is the determinant of the covariance matrix Σ.

Naive Bayes Classifier with Covariance different for different classes on dataset-1

Please find the link to the google colab file here.

Results :-

Accuracy on train data is 100%.
Accuracy on test data is 100%.

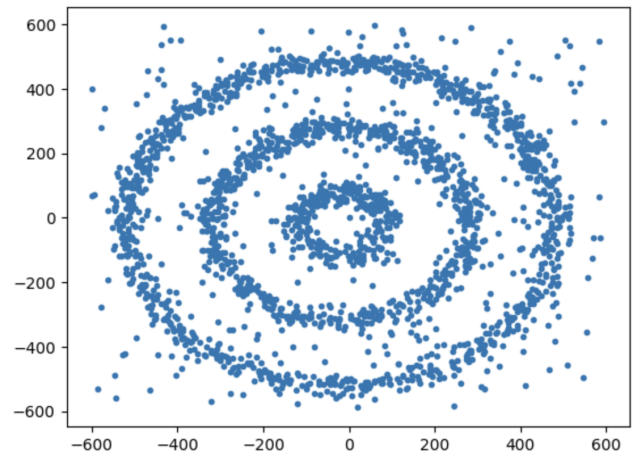


2. [**DBSCAN**] In this Question, you are supposed to implement **DBSCAN algorithm from scratch** on dataset2 provided here and dataset3 provided here. You also need to compare and contrast your observations from above with K-Means applied on both datasets. **However, you can use predefined libraries to implement K-means.**

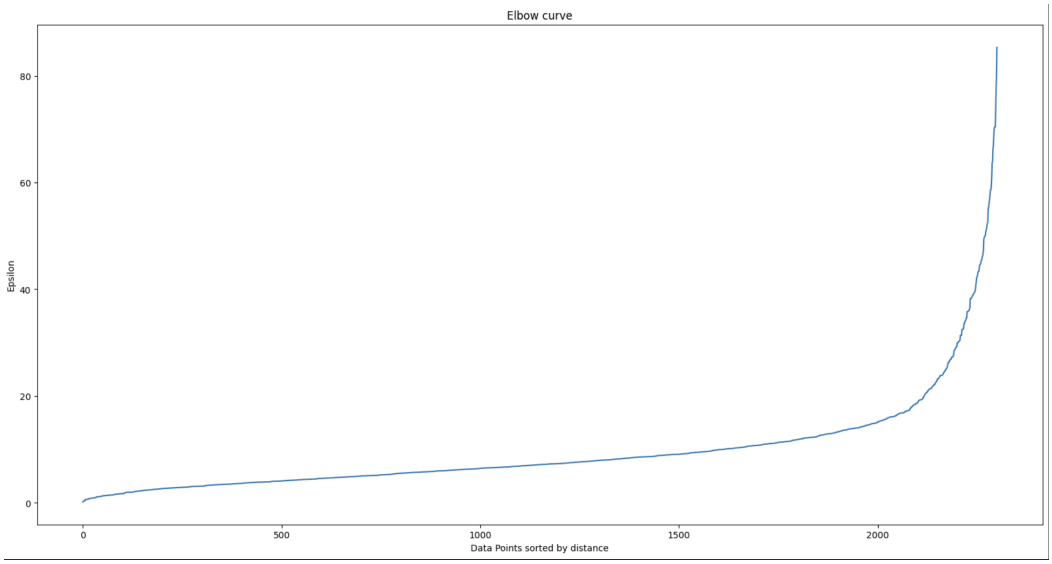
   (a) (1 mark) Visualize the data in dataset2. Then, find a suitable **range of values for epsilon** (a hyperparameter in DBSCAN algorithm) by using the 'Elbow Curve' of Datapoints plotted between K-Distance vs Epsilon. For simplicity, take only integer values for epsilon. **You can use predefined libraries to implement K-distance.**

Solution:
Please find the link to the google colab file here.

Visualising the data



We use the nearest neighbours API feature from the sklearn library to find the list's distances and their corresponding distances.
We take out the point itself in each list and plot the distance data.

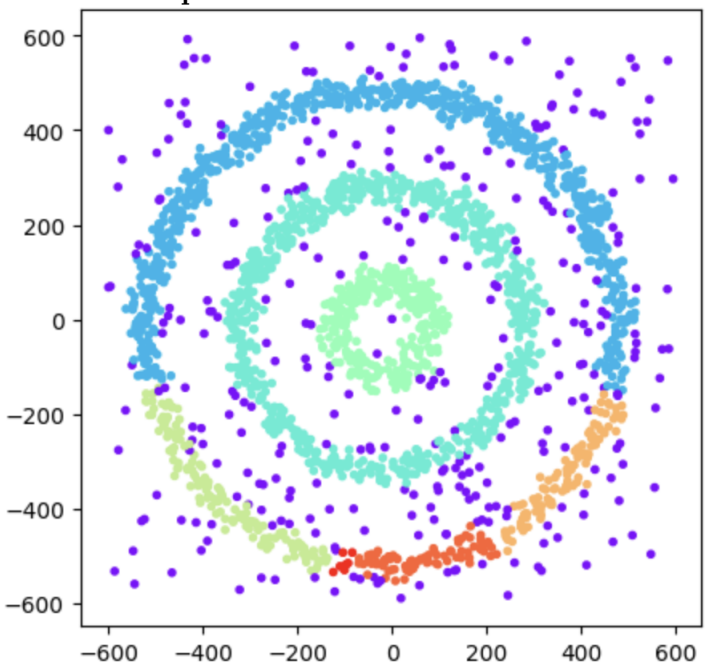


From the given elbow graph we can observe the sharp turn is around the range 25-35.

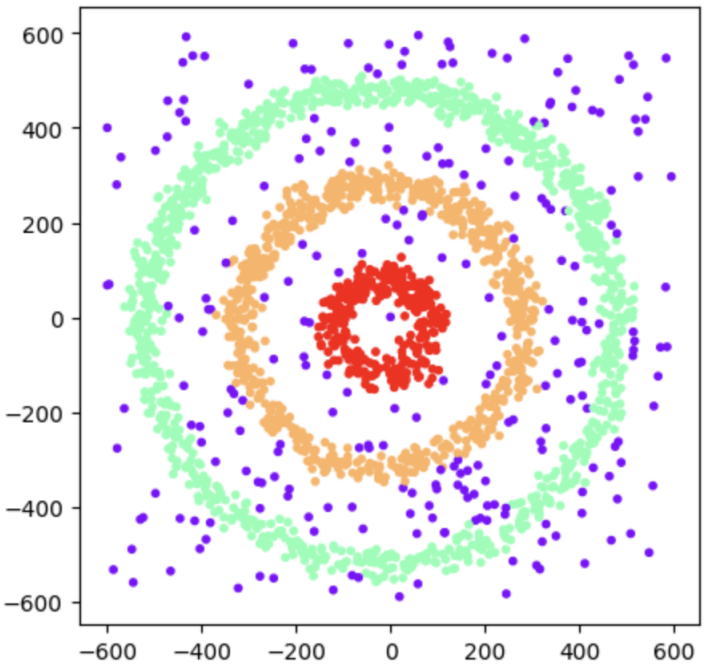
(b) (2 marks) Implement DBSCAN with the above suitable range of values of epsilon and detect the optimal value of epsilon, which gives the best clustering visually on the dataset. Show a visualization of the clusters formed for the best value of epsilon.

Solution:
Please find the link to the google colab file here.

We have optimised the DBSCAN algorithm by using the concept of DFS instead of the naive algorithm to cluster the points.
clusters for epsilon = 25

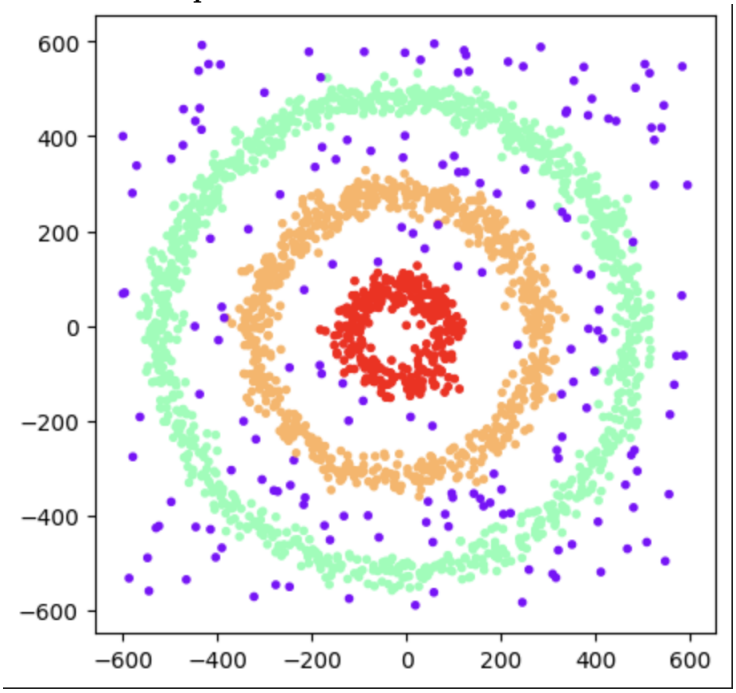


too many clusters
clusters for epsilon = 30



good clustering

clusters for epsilon = 35



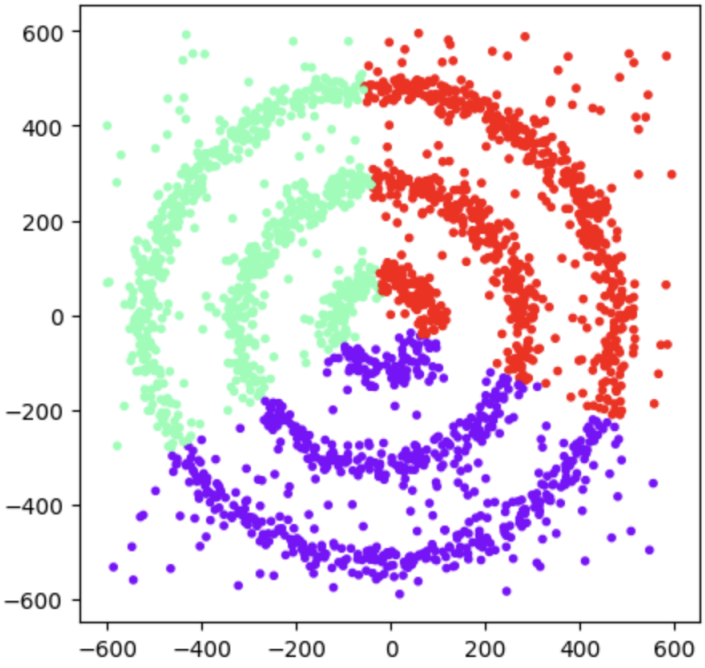
lot of outliers included

By looking at the visualised clustered data for various data, we feel epsilon =30 looks the best.

(c) (1.5 marks) Implement K-Means and use it on dataset2 with value of K (number of clusters) set to the optimum number of clusters that you get from (b) above. Suggest various techniques to improve the clustering by KMeans in this case.

Solution:
Please find the link to the google colab file here.

Clusters after using the K-Means algorithm.



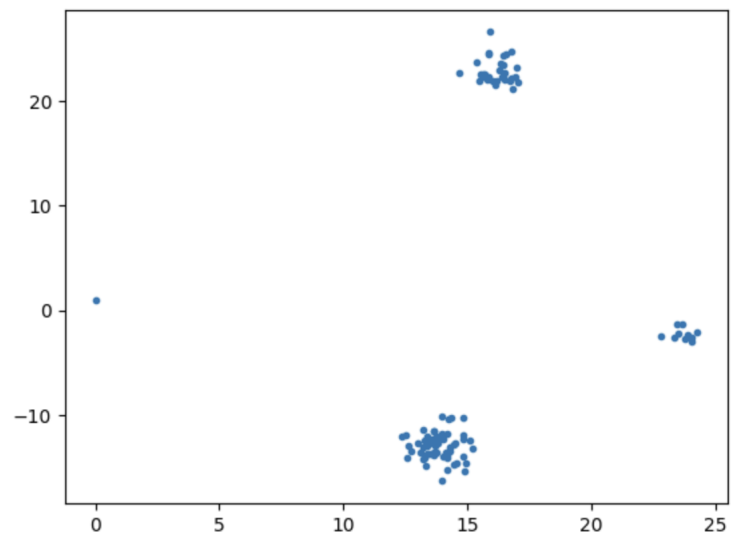
suggestions to improve K-means

- using various measures to measure the distance between neighbours.
- initialising means as apart as possible might not work in this case. Random initialisation is probably better.

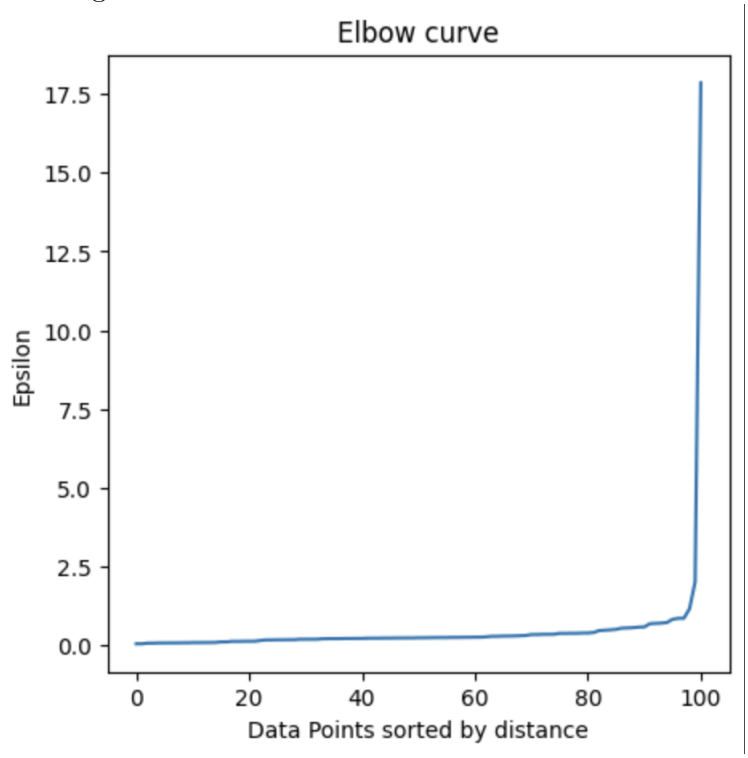
(d) (1.5 marks) Show a visualization of the data in dataset3. Use your implementation of DBSCAN with `minPts=15` on dataset3. Plot 'Elbow curve' to get an optimal range of values for `eps`. Detect the optimal value of epsilon which gives the best clustering visually on the dataset. Show a visualization of the clusters formed for the best value of epsilon.

Solution:
Please find the link to the google colab file here.

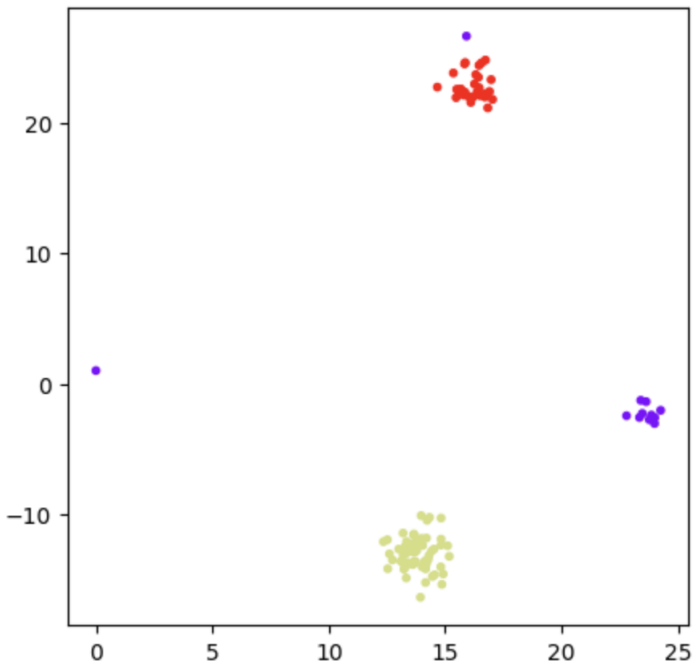
Visualising the data



Drawing the elbow curve.



The best value of epsilon value = 2.

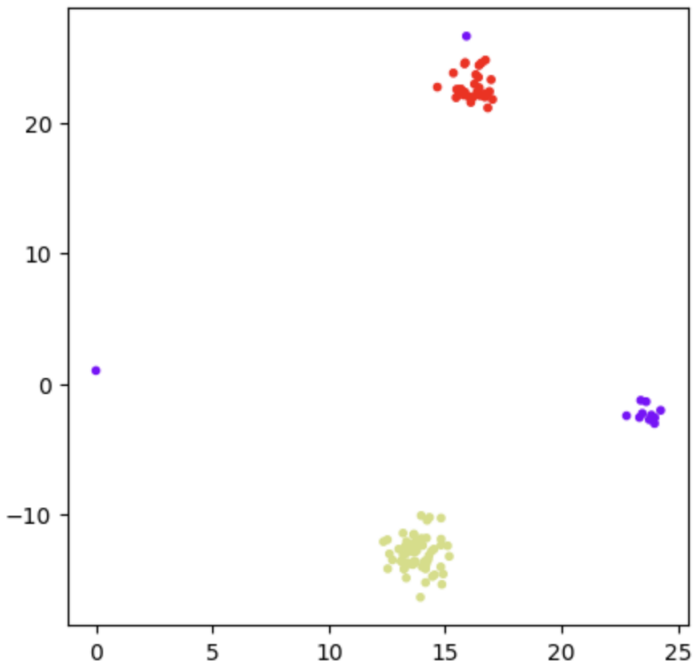


But the third cluster here is classified as outliers.

(e) (1 mark) Now perform KMeans with K=3. Write your observations for obtained results in (d) and (e). Did we give you bad initialization values?

Solution:
Please find the link to the google colab file here.

clustering after doing k-means.



Initialisation values for a minimum number of neighbours were bad. The third cluster is so small that it doesn't even have 15 points. This causes the DBSCAN algo to classify all these points as outliers.
Doing dimensional reduction using PCA or reducing dimensions, especially onto the y-axis here, will give more efficient results. Also, the curse of dimensionality will not be prevalent.

(f) (1 mark) Based on all your learnings from this question, state the relative pros and cons of KMeans vs DBSCAN.

Solution:
Please find the link to the google colab file here.

Please find the link to the comparison table here.

3. **[GMM]** In this question, you are supposed to implement the Expectation-Maximization algorithm for Gaussian mixture models on the given dataset4. The data can be found here.

(a) (3 marks) Implement EM for GMM and plot the log-likelihood as a function of iterations.

Solution:
Expectation Maximization for Gaussian Mixture Model :-

The Expectation-Maximization (EM) algorithm in the context of Gaussian Mixture

Models (GMM) is to find the optimal parameters that best fit the given data. It aims to estimate the means, covariances, and mixing coefficients of the GMM by iteratively maximizing the likelihood function.The EM algorithm achieves this by alternately updating the responsibilities (E-step) and adjusting the parameters (M-step) until convergence resulting in a reliable probabilistic model GMM for clustering and density estimation tasks.The calculations invovled in EM are :-

E-step :-

$$\gamma_k(x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

M-step :-

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)}$$

$$\Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)}$$

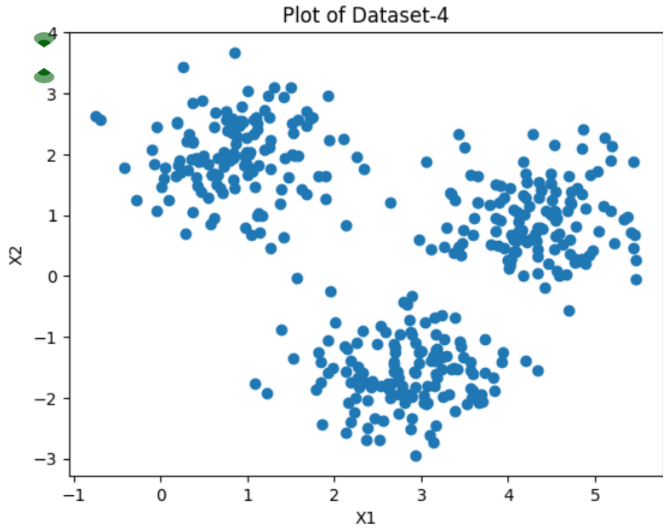
$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)$$

Log-Likelihood function :-

$$\ln p(\mathbf{X} \mid \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathbf{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k) \right\}$$

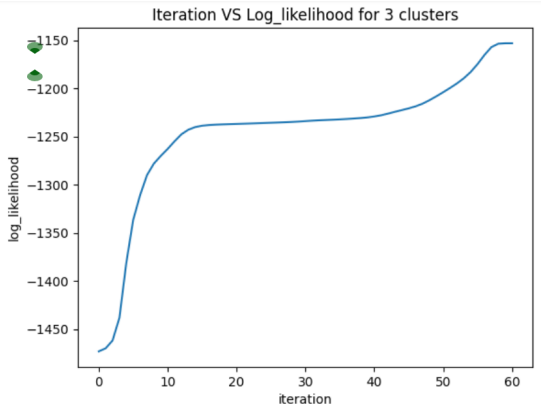
Please find the link to the google colab file here.

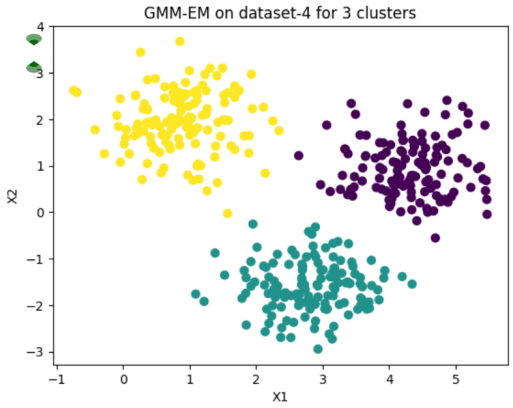
The initial scatter plot of dataset-4 :-



Results :-

Now,applying Expectation Maximization algorithm for GMM with K=3 clusters:-





(b) (2 marks) Run EM for different numbers of Gaussians (k)(Try 2,3,4,5,6). Plot figures that can help in visualization and also log likelihood as a function of iteration for different values of k. Report the observations.

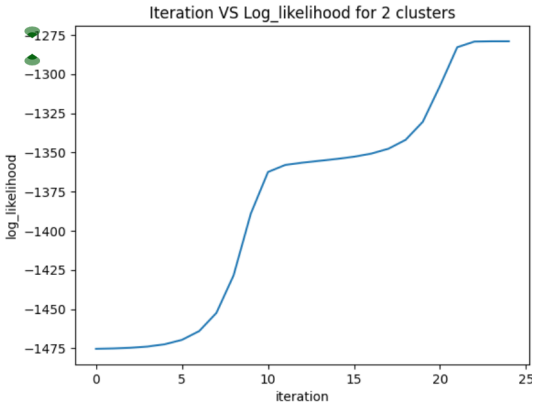
Solution:
Please find the link to the google colab file here.

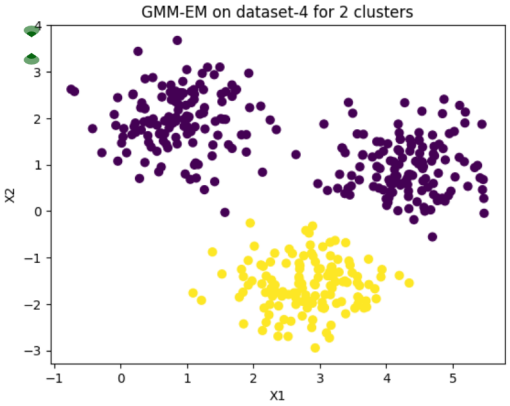
Results :-

Now, applying Expectation Maximization algorithm for GMM with K=2,3,4,5,6,7

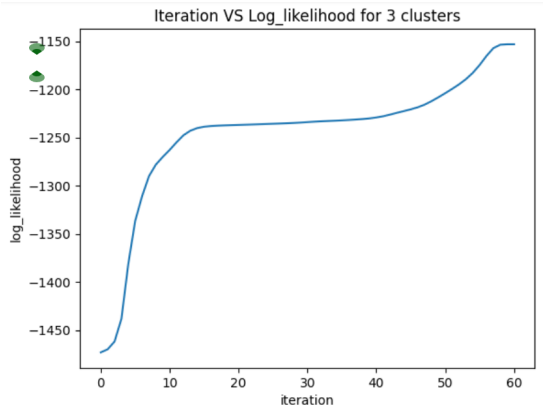
clusters:-

Clusters - 2 :-

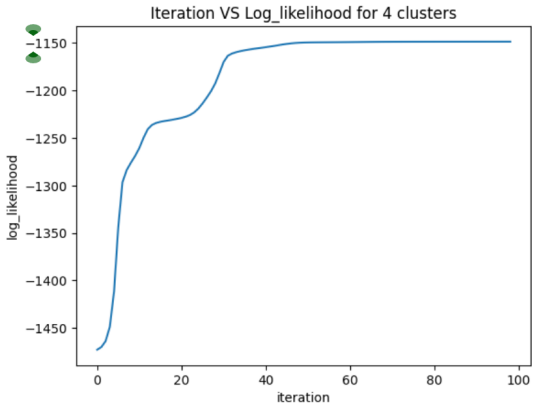


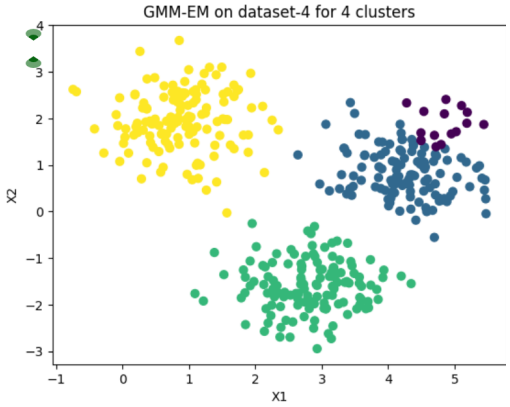


Clusters - 3 :-

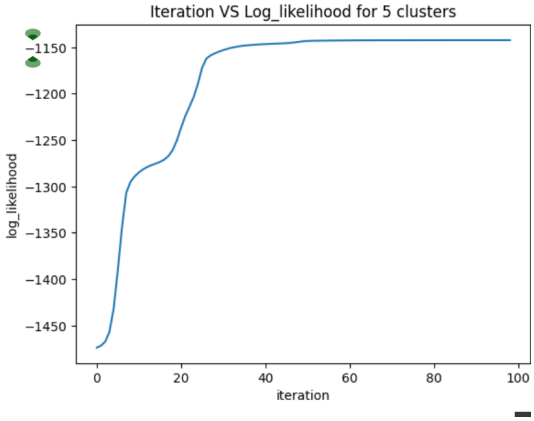


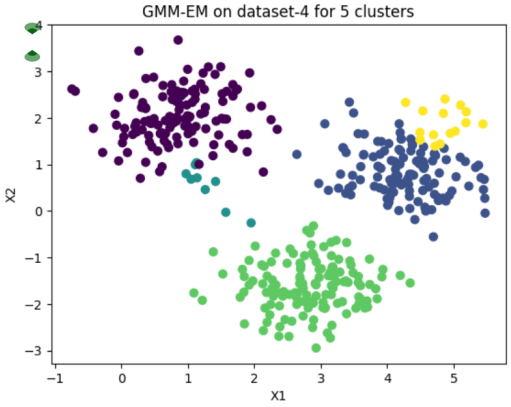
Clusters - 4 :-



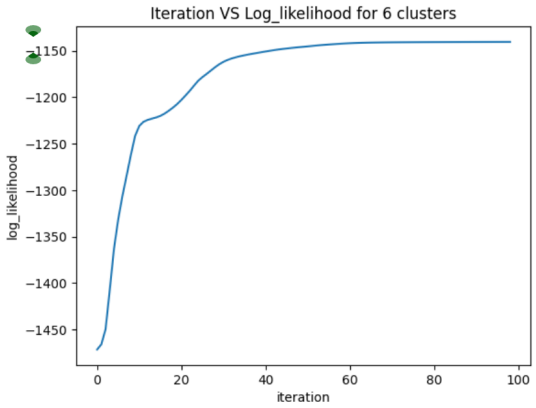


Clusters - 5 :-

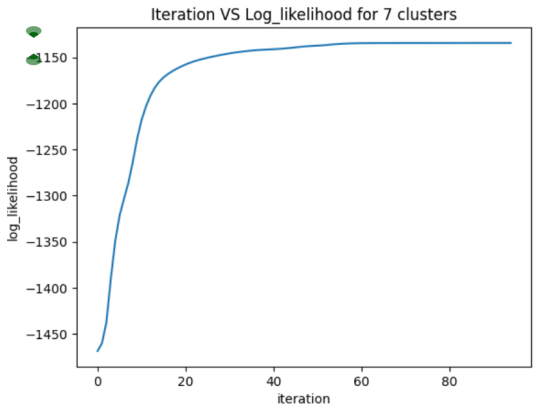


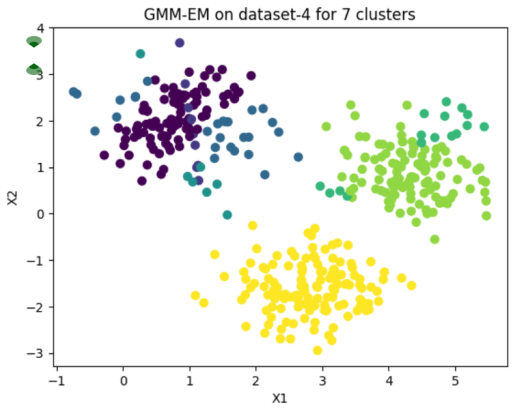


Clusters - 6 :-



Clusters - 7 :-





Obervations :-

For K=2, the Log-Likelihood is small, and then the value of the Log-Likelihood increases at K=3 and remains almost constant for K=4,5,6,7.

It is also clearly evident that the K=3 clustering seems to be most uniform among all the other clusterings.

For $K > 3$, majority of the data points are classified into 3 classes and a very few data points (some with zero data points also) have been classified to other clusters, which indicates that 3-clusters can span the whole dataset majorly which matches our visual observation.

(c) (2 marks) Find the optimal k. There are several metrics like Silhouette score, Distance between GMMs, and Bayesian information criterion (BIC), or even you can use log-likelihood from the last question to infer. Give a clear explanation for your decision.
Note: **You can use third-party libraries - sklearn or any other only in this subsection.**

Solution:
The mertric used to find the optimal K is :-

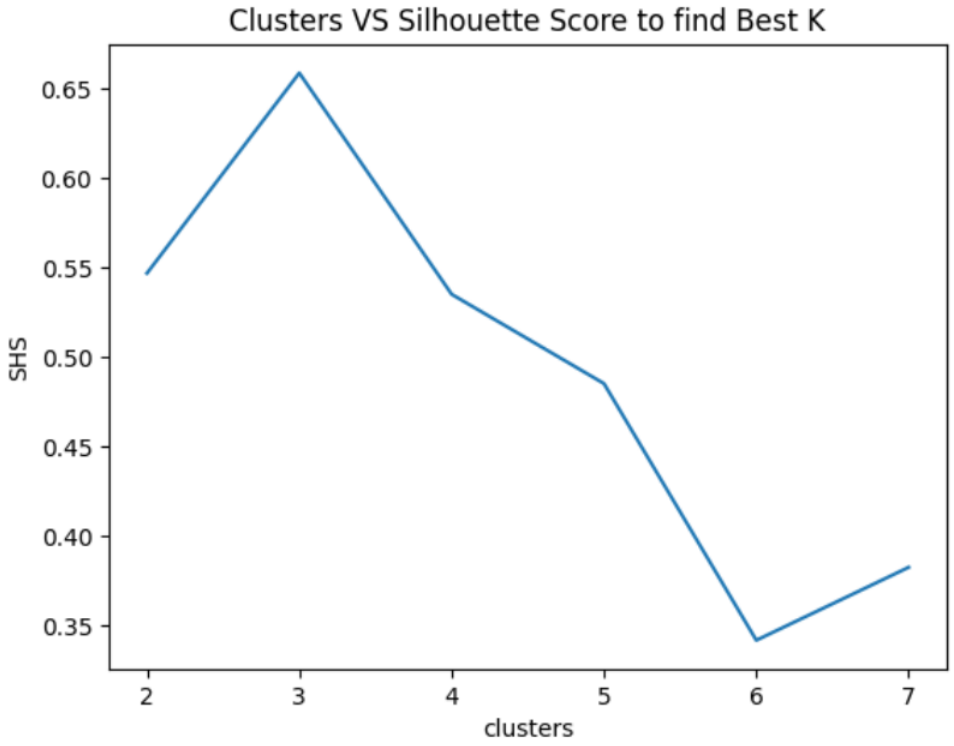
Silhouette Score :- A measure of how well-defined and separated clusters are in a clustering analysis. It is calculated based on the distances between data points within a cluster and the distances between data points in different clusters. The calculations involved are :-

$$s = \frac{b - a}{\max(a, b)}$$

where **a** - mean intra-cluster distance and **b** - mean next nearest cluster distance.

The **larger** the Silhouette Score, the **better** is the model.

Silhouette Scores for the K=2,3,4,5,6,7 GMM models :-



Here, the Silhouette Score is maximum for K=3.

Optimal Clusters K = 3