CS5691: Pattern Recognition and Machine Learning
Assignment #2

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

Teammate 1: (M.KARTHIK) (% 75) **Roll number:** CS20B048
Teammate 2: (PAVAN NARASIMHA GOUD) (% 25) **Roll number:** CS20B038

- **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.

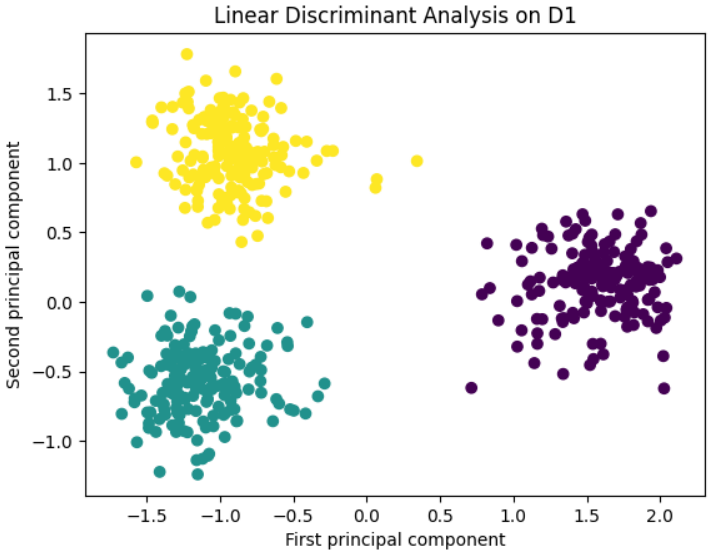
   Note that you have to implement **LDA from scratch** without using any predefined libraries (i.e. sklearn, scipy) . However, you can use **predefined libraries to implement PCA.**

   (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.

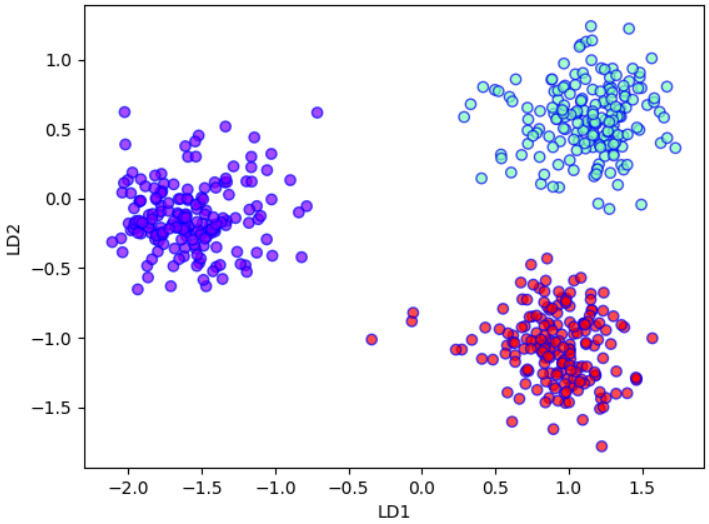
   Solution:

   We use LDA to convert high dimensional data into lower dimensional
   Here given some n dimensional data and we shoud convert that into 2 dimenonal using LDA

   Let's see the visualization of dataset2 after perfoming LDA



   Let's see how the results will change **if we perform any normalization on the data** (i.e with means as zero and variance as 1)

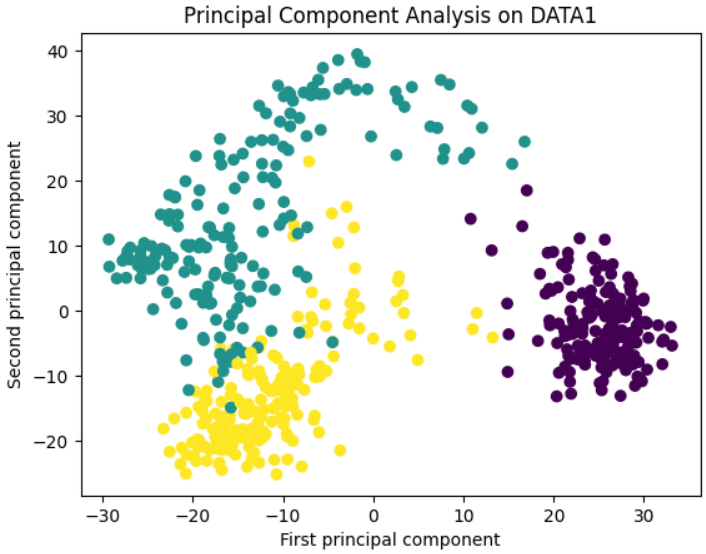


We can observe that the data has been changed after **normalization**

(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:

Visualization of the obtained data



Comparison between LDA and PCA:
1) We can clearly see that LDA seperated the data set very clearly
2) But PCA didn't seperated Well, that the data is also getting overlap
3) We already know that the LDA will also take care of the seperation between clusters which we can observe here

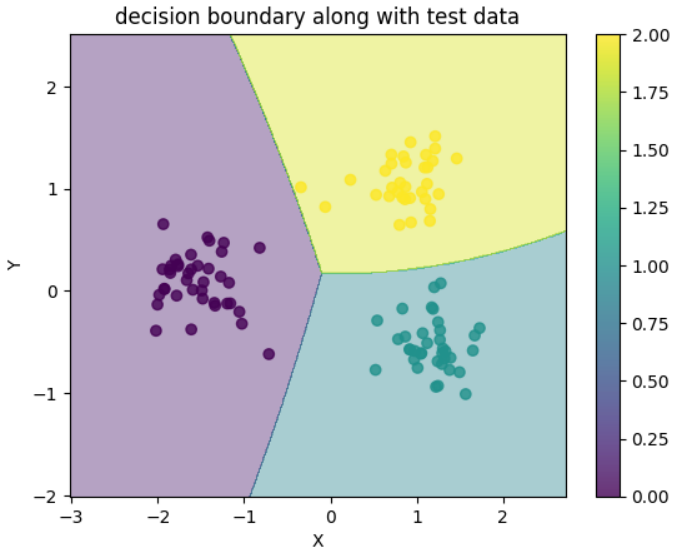
(c) (1.5 marks) Randomly shuffle and split the obtained dataset from part (a) into a training set (80%) and testing set (20%). Now build the Bayes classifier using the training set and report the following:

- 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.

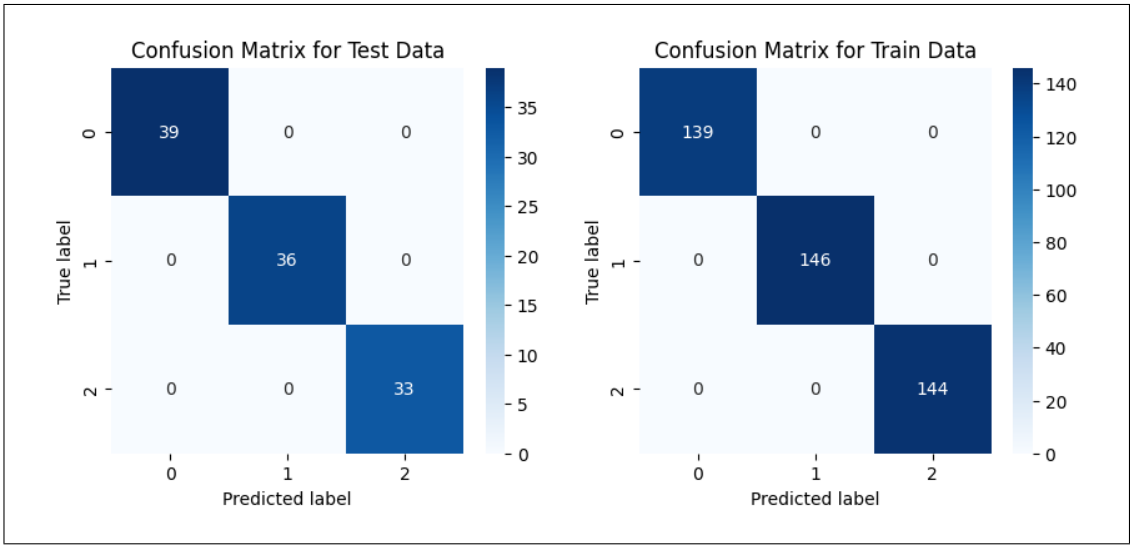
Solution:
After shuffling the data and consedering the training data 80 percent and testing data 20 percent
By implementing the Bias Classifier the ouput's are as follows

ACCURACY:
For train data i got 100 percent
For test data i got 100 percent

CLASSIFICATION BOUNDRY for both train and test data is as follows



CONFUSION MATRICES for both train and test data is as follows

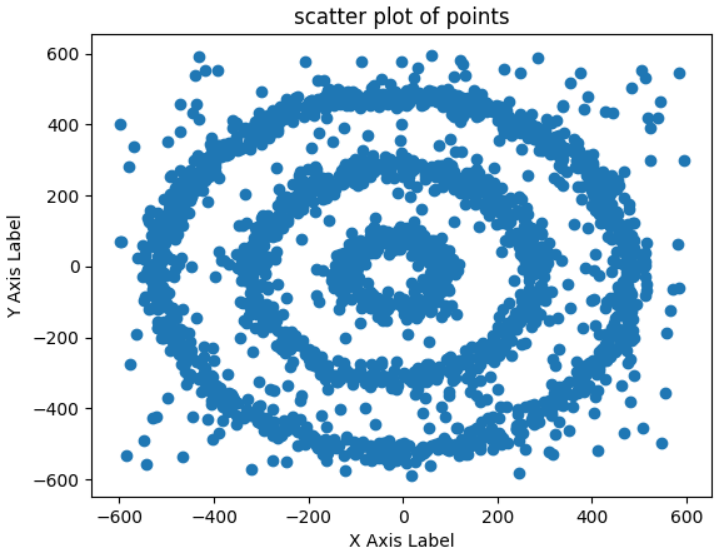


2. [**DBSCAN**] In this Question, you are supposed to implement **DBSCAN algorithm from scratch** on dataset2 provided here and dataset3 provided here. You also need to compare and contrast your observations from above with K-Means applied on both datasets. **However, you can use predefined libraries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

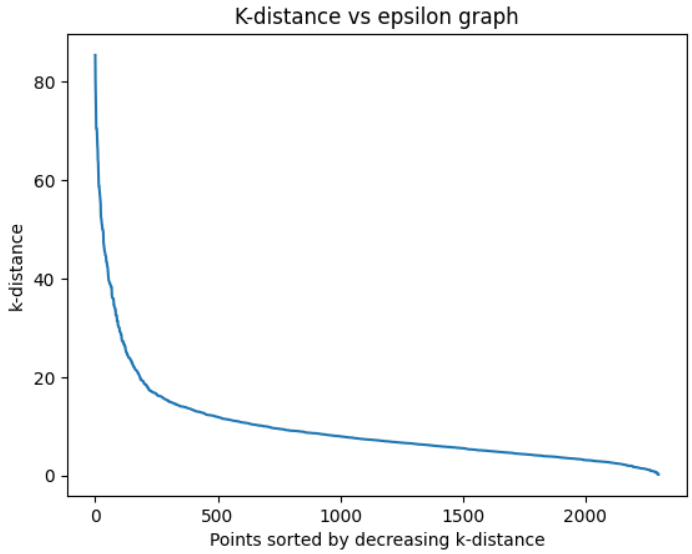
   (a) (1 mark) Visualize the data in dataset2. Then, find a suitable **range of values for epsilon** (a hyperparameter in DBSCAN algorithm) by using the 'Elbow Curve' of Datapoints plotted between K-Distance vs Epsilon. For simplicity, take only integer values for epsilon. **You can use predefined libraries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

   Solution:

   Now let's apply **PCA** on the dataset1 and convert them into 2 dimensional data
   Data visualization for the dataset2 is as below



K-distance vs epsilon for the givnen data is as follows

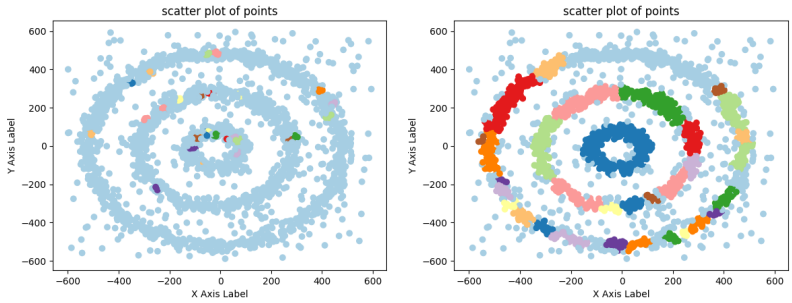


(b) (2 marks) Implement DBSCAN with the above suitable range of values of epsilon and detect the optimal value of epsilon, which gives the best clustering visually on the dataset. Show a visualization of the clusters formed for the best value of epsilon.

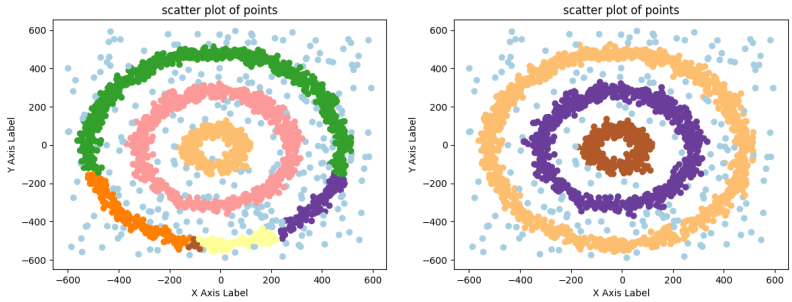
Solution:

from the above graph i have taken the epsilon values from 10 - 40 and i have implemented the DBSCAN

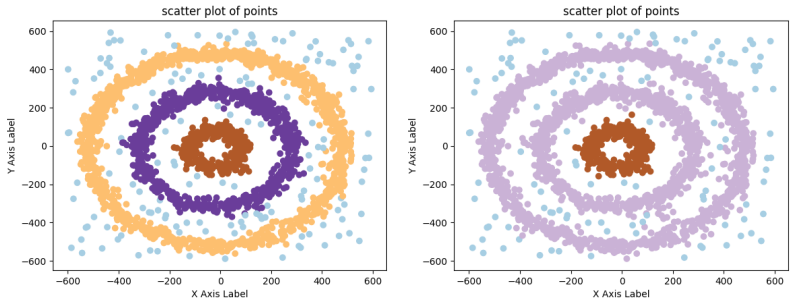
Cluster Visualization for epsilon = **10 and 20** is as below



Cluster Visualization for epsilon = **25 and 30** is as below



Cluster Visualization for epsilon = **35 and 40** is as below



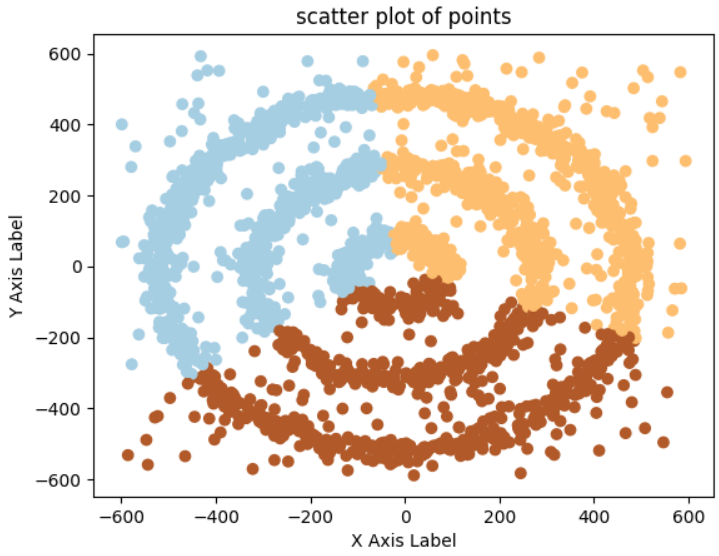
From the above we are getting the best visualization for epsion = **30**

(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:

from the above i got no of clusters = 3

If i use K-means the clusttering is as below



K-means algorithm works well when the clusters are well-separated and have a spherical shape, but it can struggle when the clusters are more complex, such as concentric circles.

suggestions: to improve the clustering by K-Means in this case

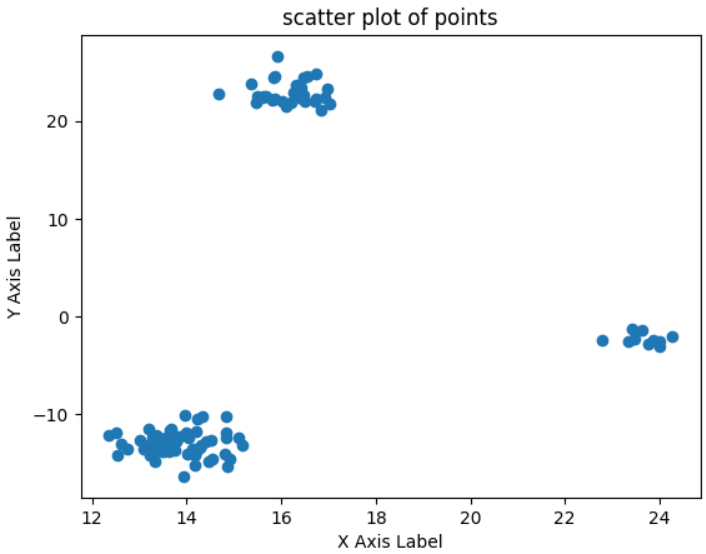
1 : **Transform the data**: One approach is to transform the data to a new feature space where the clusters are more separable. For example, in the case of concentric circles, the data can be transformed into polar coordinates, where the distance from the center and the angle can be used as features.
2 : **use different distance matrices**, i.e K-means uses the ecledian distance , instead of that use a different distance metric such as the Mahalanobis distance can be used
3: Use hierarchical clustering
4: Use density-based clustering
5: Use ensemble methods

(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.

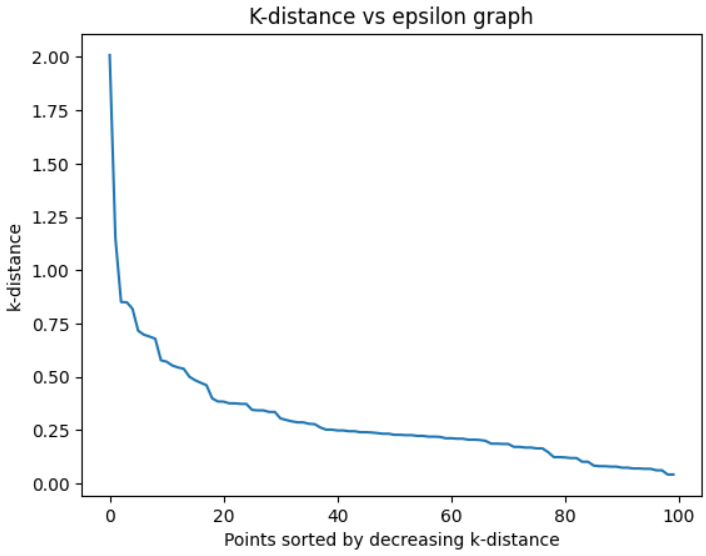
Solution:

Visualization of dataset3 :



By implementing the DBSCAN with minpoints as 15

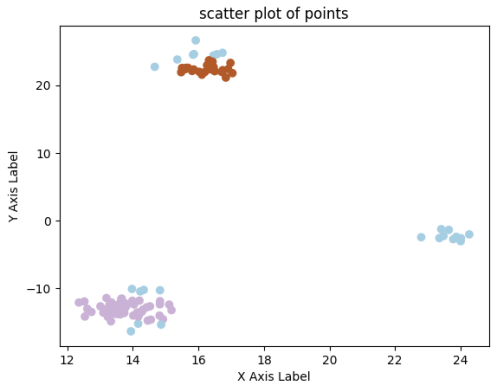
The elbow curve is as follows

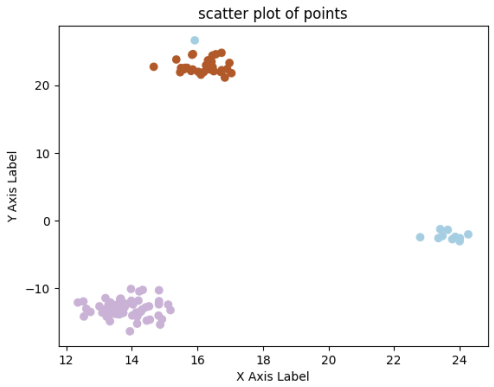


Since we are thaking the integer values of epsilon from the above graph i am trying

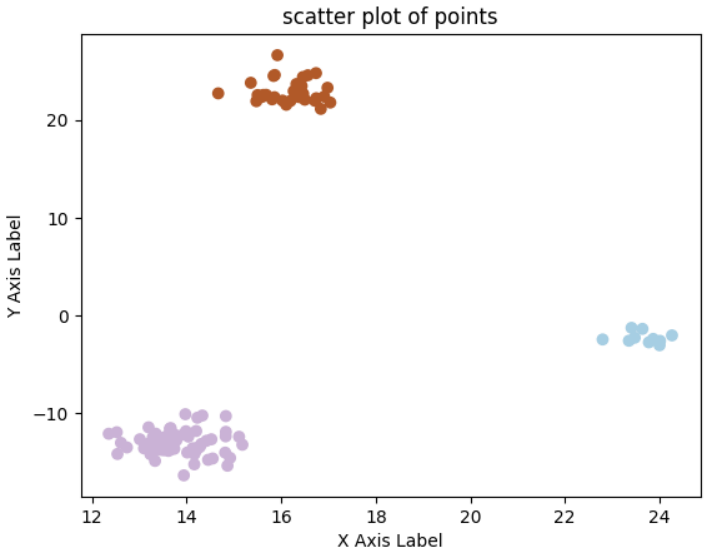
for the epsion values for 1, 2, 3 with minpoints as 15

Lets draw the clustering visualization for epsion value 1 and 2 with minpoints = 15 is as follows





The clustering visualization for epsilon value 3 with minpoints = 15

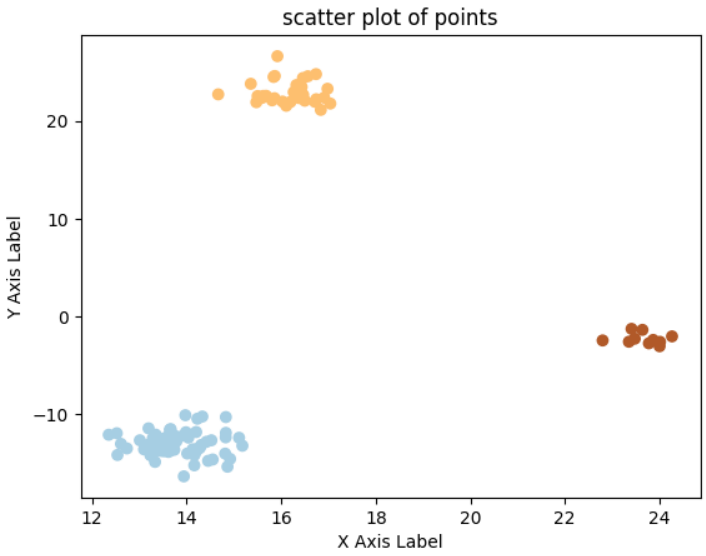


From the above observation the best clusterring is forming at epsilon = 3 for minpoints = 15
Here two clusters are formed **(not three the blue colour once are noice points)**

(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:
Now lets perform the K Means on the above data and observe (WE are doing for K = 3)

————————————K-means clustering for dataset1 ————————



OBSERVATION:
1) In k-MEANS 3 clusters are formed
2) In DBSCAN 2 clusters are formed
From the above K-means graph we can clearly see that the clustering is very good as compared to DBSCAN
3) We had some **BAD INITIALIZATION** in the case on DBSCAN (i.e we have initailized the **min no point = 15** which is very large
4) If we decrease the min no points then the DBSCAN will also work well

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

Solution:

I have learned both K-means and DBSCAN both are useful but the usage will depend on the situtaion , i.e What the type of data we are using , based on the type of data we should use K-means and DBSCAN

ABOUT K-MEANS:
PROS:
1) K-means is computationally efficient and can handle large datasets

2) K-means is effective at identifying clusters of data points that have similar characteristics.
3) K-means can be used for a variety of applications, such as image segmentation, text classification, and customer segmentation.
4) K-means can be customized by choosing the number of clusters (K) to suit the needs of the user.

CONS:
1)K-means is sensitive to the initial placement of the centroids, which can lead to different results each time the algorithm is run.
2)K-means assumes that the clusters are spherical and of equal size, which may not always be the case in real-world data.
3)K-means may not work well with datasets that have a high degree of overlap between clusters.

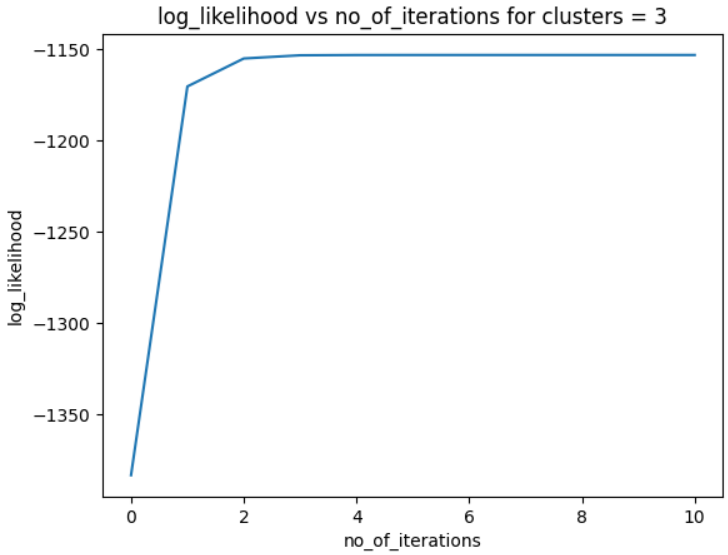
ABOUT DBSCAN:
PROS:
1)DBSCAN can automatically identify the number of clusters and their shapes, without requiring the user to specify them in advance.
2)DBSCAN is effective at identifying clusters of arbitrary shapes and sizes.
3)DBSCAN can handle noisy and outlier data points by labeling them as noise or outliers.

CONS:
1)DBSCAN can be sensitive to the choice of the distance metric used to calculate the distance between data points.
2)DBSCAN may not work well with datasets that have varying densities within clusters.
3)DBSCAN may not work well with datasets that have a large number of clusters with similar densities.

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.

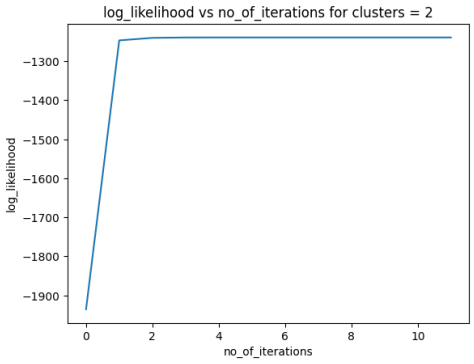
Solution:



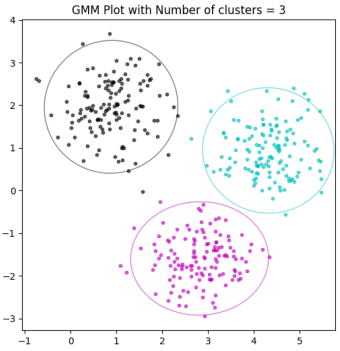
Here i have implemented the EM for GMM and the log-likelihood graph is as below and the log-likelihood value is am getting is nearly equal *-1150*

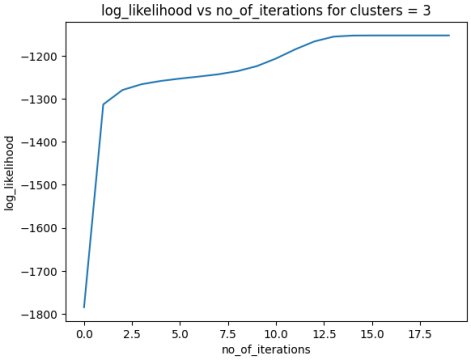
(b) (2 marks) Run EM for different numbers of Gaussians (k)(Try 2,3,4,5,6). Plot figures that can help in visualization and also log likelihood as a function of iteration for different values of k. Report the observations.

Solution: **GMM** for no of Gaussians($\mathbf{K} = 2$) is as follows
And also the log-likelihood vs no of iterations for no of clusters = 2 is as follows

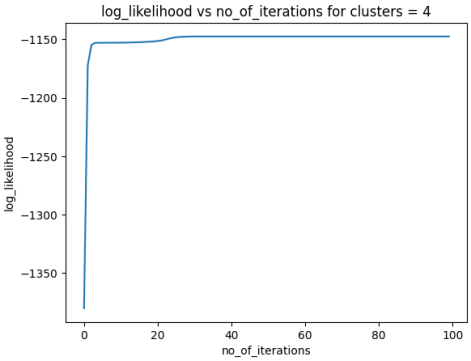


GMM for no of Gaussians($\mathbf{K} = 3$) is as follows
And log-likelihood vs no of iterations for no of clusters = 3 is as follows

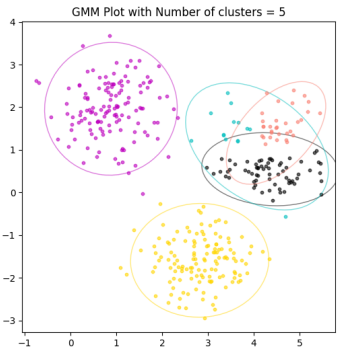


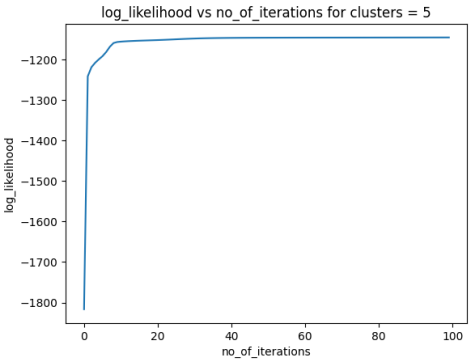


GMM for no of Gaussians($\mathbf{K} = 4$) is as follows
And log-likelihood vs no of iterations for no of clusters = 4 is as follows

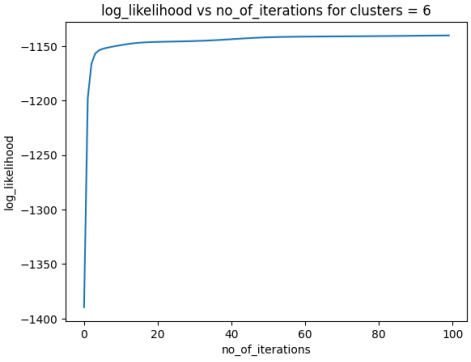


GMM for no of Gaussians($\mathbf{K} = 5$) is as follows
log-likelihood vs no of iterations for no of clusters = 5 is as follows





GMM for no of Gaussians($\mathbf{K} = 6$) is as follows
log-likelihood vs no of iterations for no of clusters = 6 is as follows



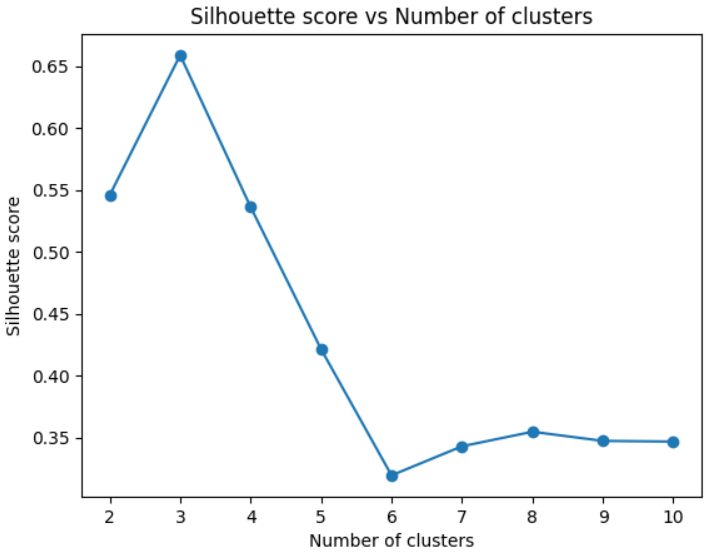
REPORTING OBSERVATION:
CASE1 : When K is low
when K is low GMM is not able to represent the underlying data because it is having the low components (which is not good), i.e low perfomance

CASE2 : When K is high
Whenn K is high GMM is overfitting and becoming very closely and having more no of components
i.e when an training data is given it may not able to detect the new data because they are very close
implies leads to low prefomance which is not good

(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:
METHOD1:
finding the optimal value of K using the **Silhouette score** method

Silhouette score Vs Number of clusters graph is as follows

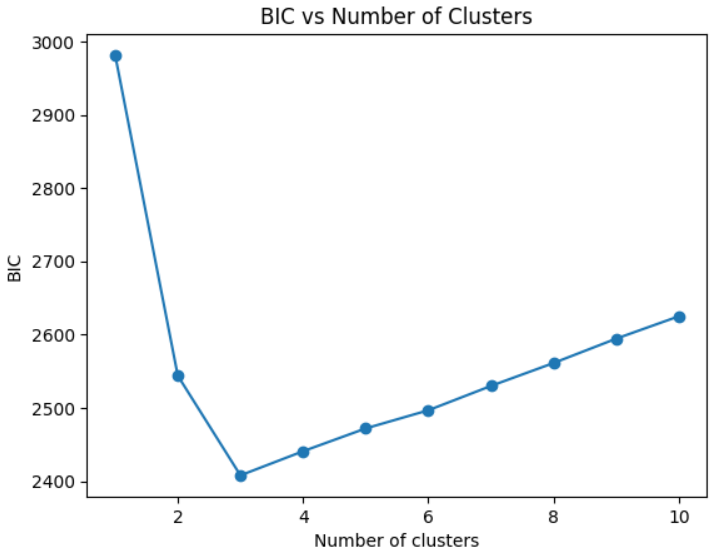


from the above graph we can observe that we are getting the max score at no of clusters = 3

Scatter plot of data based on Silhouette Score (i.e for no of clusters = 3)

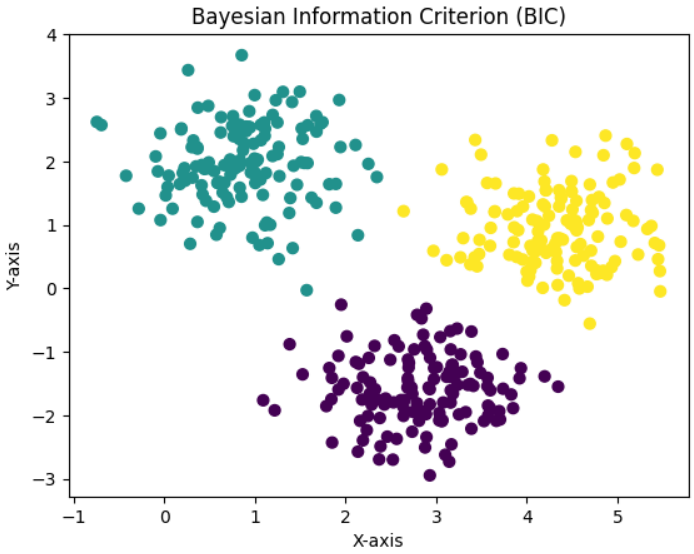
METHOD2:
finding the optimal value of K using the **BIC** method

BIC Vs Number of clusters graph is as follows



YES, from this graph also we can observer that the optimal value of K we are getting is 3

Scatter plot of data based on BIC Score (i.e for no of clusters = 3)



Which is similar to the graph we get in the Silhouttee method