CS5691: Pattern Recognition and Machine Learning
Assignment #2

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

Teammate 1: (Karan Bardhan) (Q1 and Q3) **Roll number:** CS20B036
Teammate 2: (Smit Bagul) (Q2) **Roll number:** CS20B011

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.**

   Google colab notebook solution for Q1 : here

   (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:
Let's see the solution obtained by LDA on the dataset without any normalisation. Note that we have removed the columns with 0 variance across all the classes as they will cause a problem while calculating inverse of Scatter Matrix:

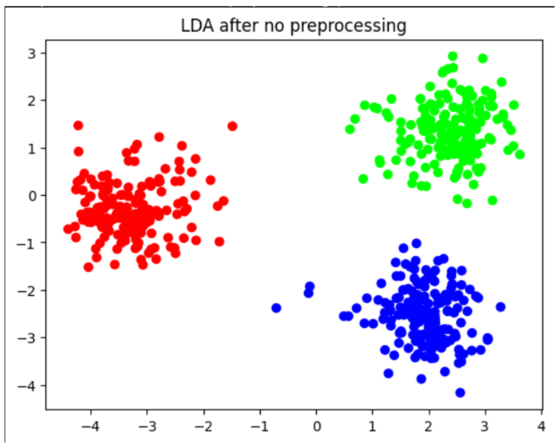


Figure 1: LDA without normalisation

Now let's see what happens when we pre-process the data, we have applied two different normalisation methods, namely -
1) Min-Max Scaling:

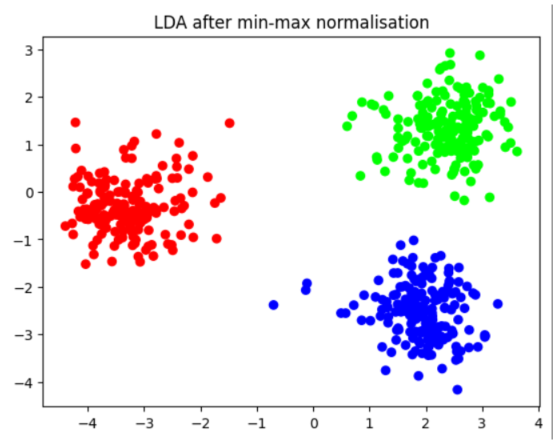


Figure 2: LDA with Min-Max Scaling

2) Zero Mean Unit Variance Normalisation:

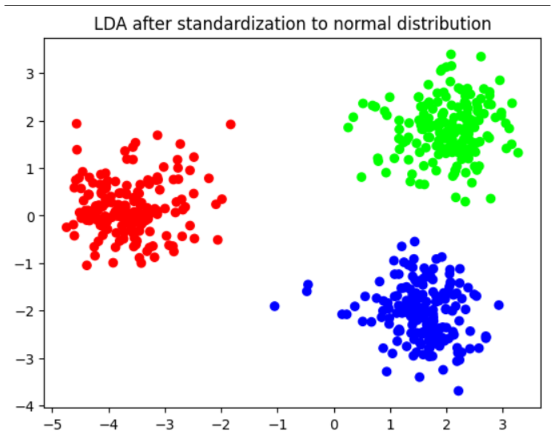


Figure 3: LDA with Zero Mean Unit Variance Normalisation

Surprisingly we notice that normalisation has no effect on the transformation done by LDA, i.e, results obtained applying LDA on both normalised and non-normalised datasets are exactly the same

(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:
We used the sklearn modules for implementing PCA and obtained the following results on the non normalised dataset and also the 0 mean 1 variance normalised dataset. Here as well we are removing features with 0 variance.
1) Non-normalised

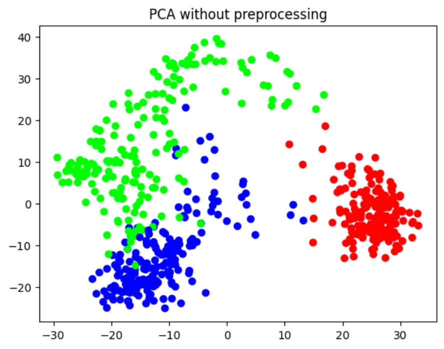


Figure 4: PCA without Normalisation

2) Normalised

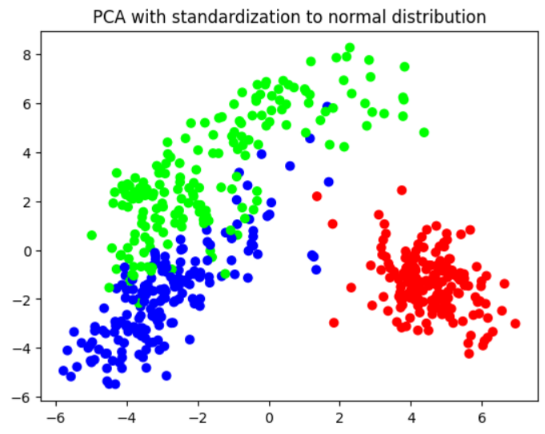


Figure 5: PCA with Zero Mean Unit Variance Normalisation

At first glance it looks like the figures are same, however on careful observtion, we notice that the normalisation actually affects the distribution obtained after PCA. Now let's compare between PCA and LDA

- LDA is insensitive to normalisation however normalisation leads to a better distinction in classes in PCA

- The classes obtained after applying LDA are more distinct than after applying PCA. This can be attributed to the fact that LDA is a supervised learning method whereas PCA is an unsupervised learning method

(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:
For this part of the solution, we have included three kinds of Bayes classifiers:
1) Cov = I for all classes:

- Accuracy:
  - Train : 100
  - Test : 100
- Classification Boundary:

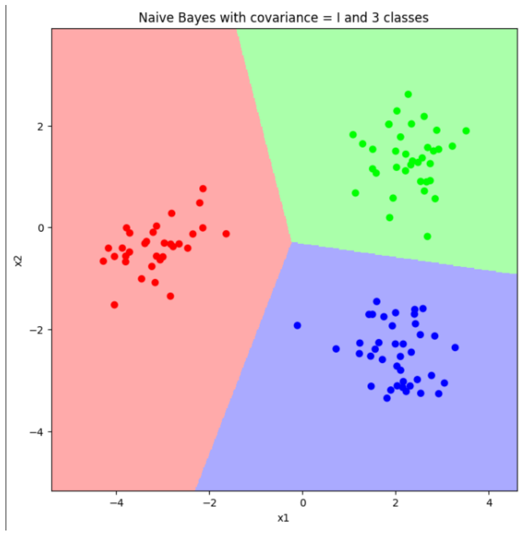


Figure 6: Classification Boundary for Cov = I

- Confusion Matrix:

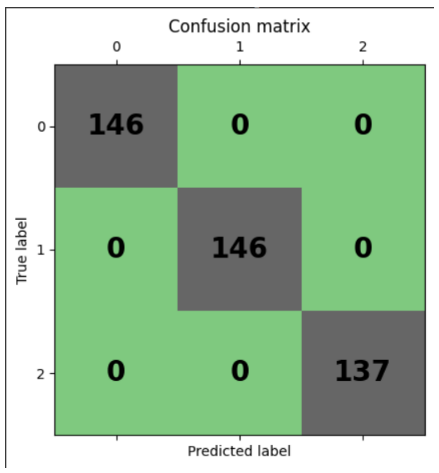


Figure 7: Confusion Matrix for Train Data for Cov = I

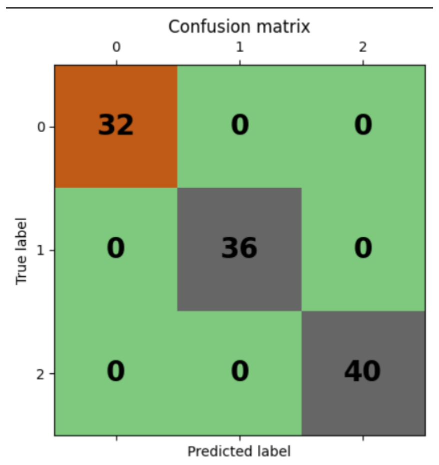


Figure 8: Confusion Matrix for Test Data for Cov = I

2) Cov $\neq$ I but same for all classes:

- Accuracy:
  - Train : 100
  - Test : 100
- Classification Boundary:

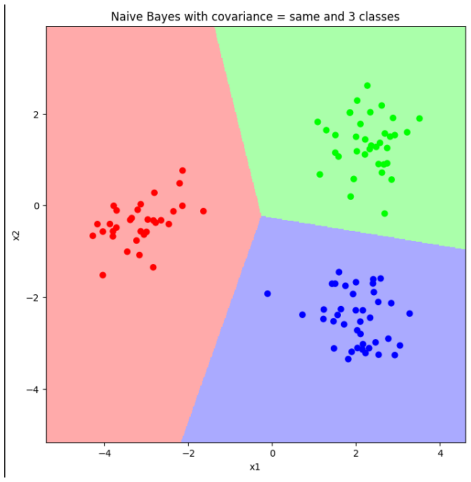
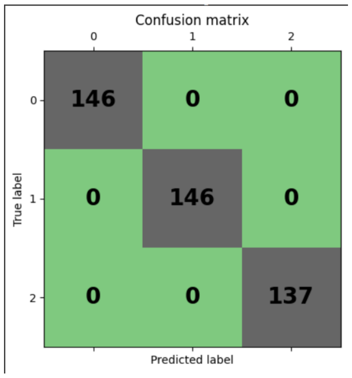


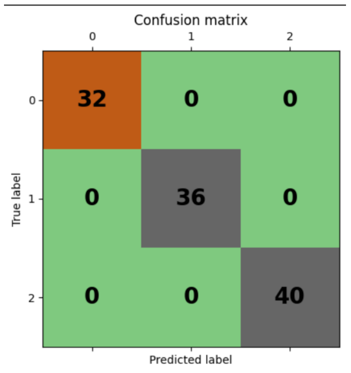
Figure 9: Classification Boundary for same Cov

- Confusion Matrix:


Confusion matrix
Predicted label
True label

	0	1	2
0	146	0	0
1	0	146	0
2	0	0	137

Figure 10: Confusion Matrix for Train Data for same Cov


Confusion matrix
Predicted label
True label

	0	1	2
0	32	0	0
1	0	36	0
2	0	0	40

Figure 11: Confusion Matrix for Test Data for same Cov

3) Cov is different for all classes:

- Accuracy:
  - Train : 100
  - Test : 100
- Classification Boundary:

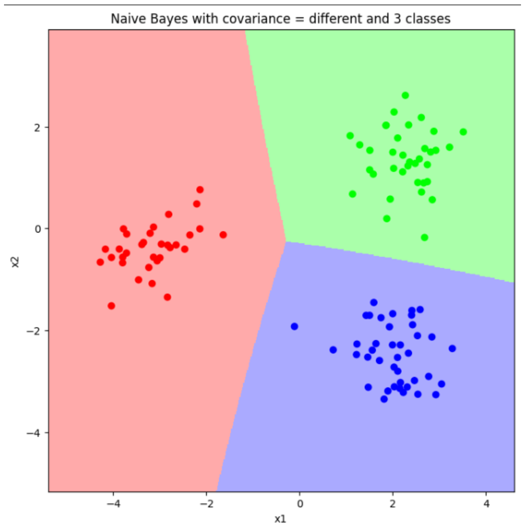


Figure 12: Classification Boundary for different Cov

- Confusion Matrix:

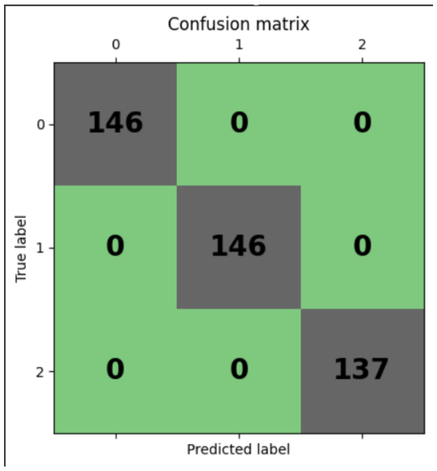


Figure 13: Confusion Matrix for Train Data for different Cov

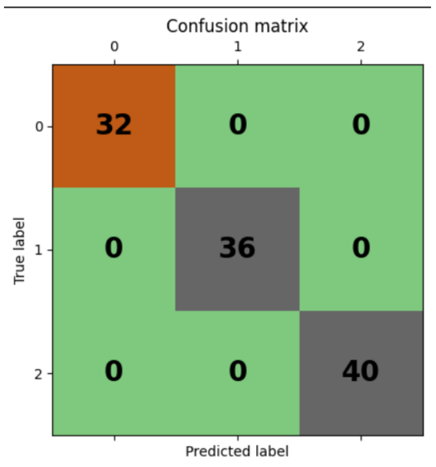


Figure 14: Confusion Matrix for Test Data for different Cov

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.**

   Google colab notebook solution for Q2 : here

   (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:
   Scatter plot for dataset 2:

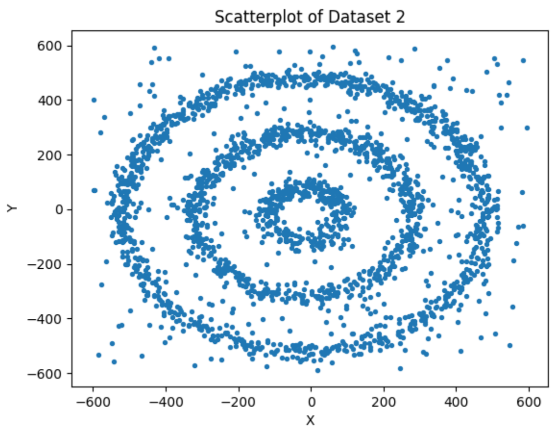


Figure 15: Scatter Plot for dataset 2

Elbow curve:

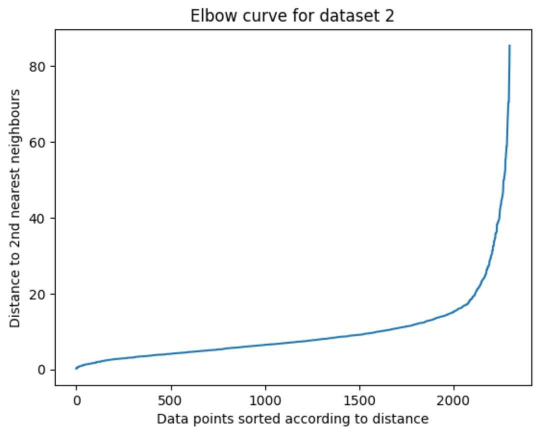


Figure 16: Elbow curve for dataset 2

As we can see in the elbow curve the suitable range of values for epsilon is between 20 and 40

(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:
We will plot the scatter plots with 2 different minpoints i.e 4 and 8 and compare accordingly for optimal epsilon
We plot the graph using min points = 4 and epsilion ranging from 20 to 40

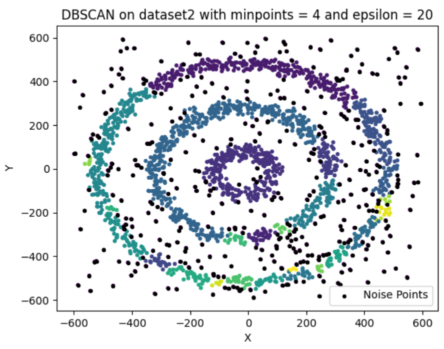


Figure 17: Scatter Plot with min points = 4 and epsilon = 20

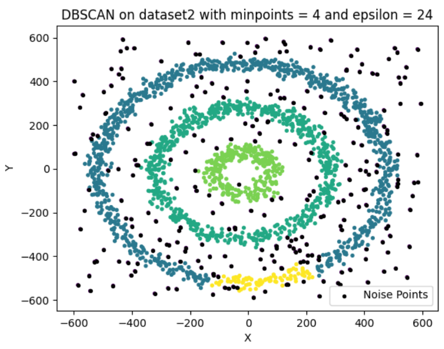


Figure 18: Scatter Plot with min points = 4 and epsilon = 24

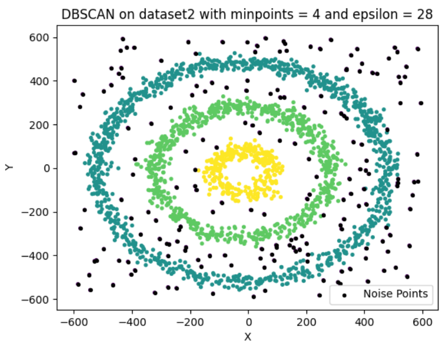


Figure 19: Scatter Plot with min points = 4 and epsilon = 28

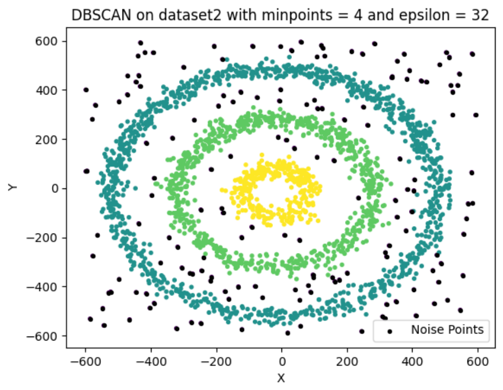


Figure 20: Scatter Plot with min points = 4 and epsilon = 32

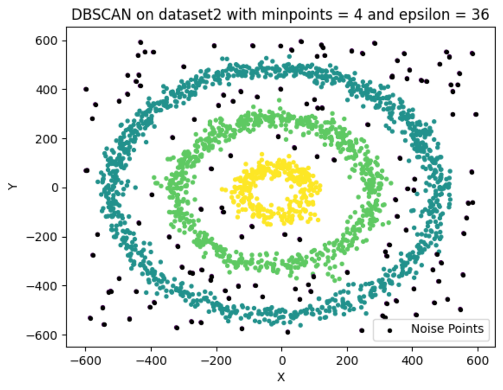


Figure 21: Scatter Plot with min points = 4 and epsilon = 36

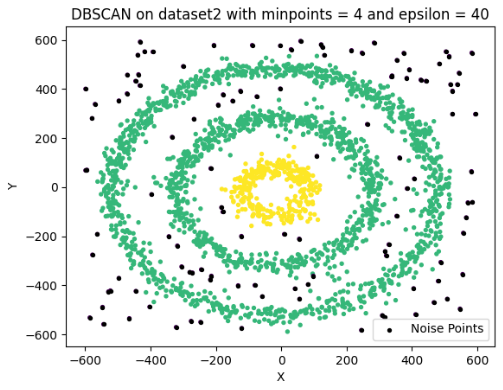


Figure 22: Scatter Plot with min points = 4 and epsilon = 40

As seen in the figures we get visually good plots for min points = 4 and epsilon ranging from 28 to 36.

We plot the graph using min points = 8 and epsilon ranging from 20 to 40

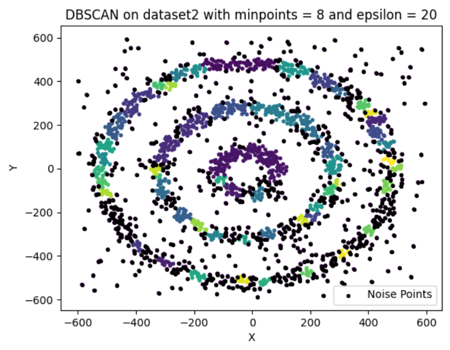


Figure 23: Scatter Plot with min points = 8 and epsilon = 20

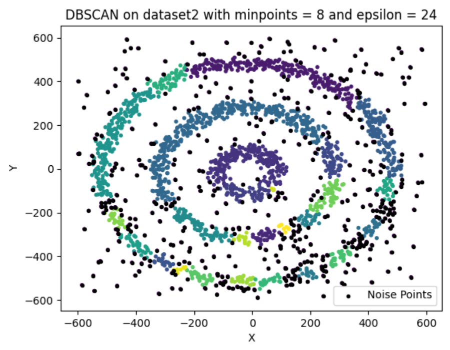


Figure 24: Scatter Plot with min points = 8 and epsilon = 24

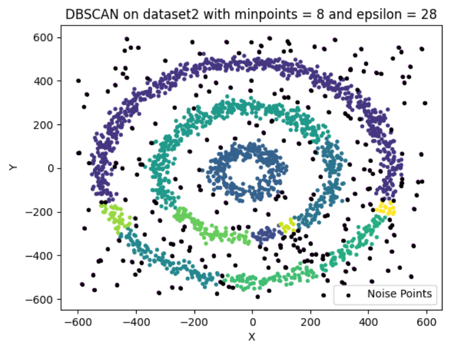


Figure 25: Scatter Plot with min points = 8 and epsilon = 28

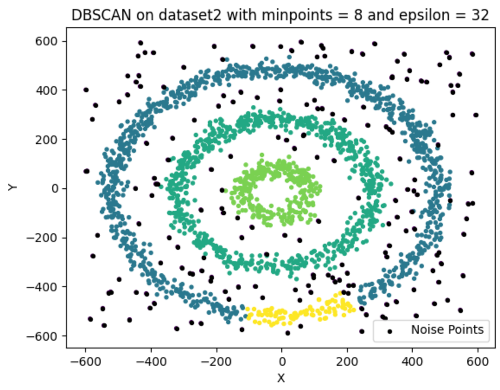


Figure 26: Scatter Plot with min points = 8 and epsilon = 32

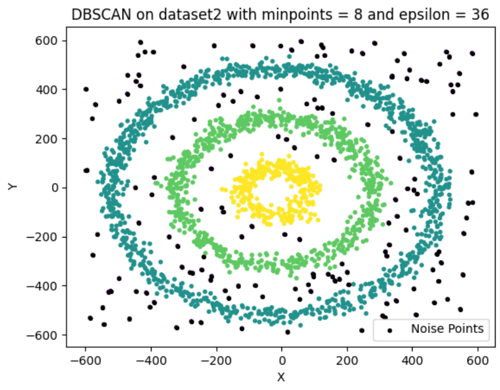


Figure 27: Scatter Plot with min points = 8 and epsilon = 36

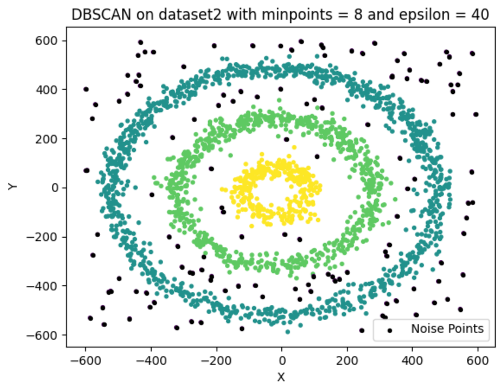


Figure 28: Scatter Plot with min points = 8 and epsilon = 40

As seen in the figures we get visually good plots for min points = 8 and epsilon ranging from 36 to 40.

Hence,we can infer the optimal value of epsilon to be 36.

(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:
The optimal value of K will be 3
K means using K = 3

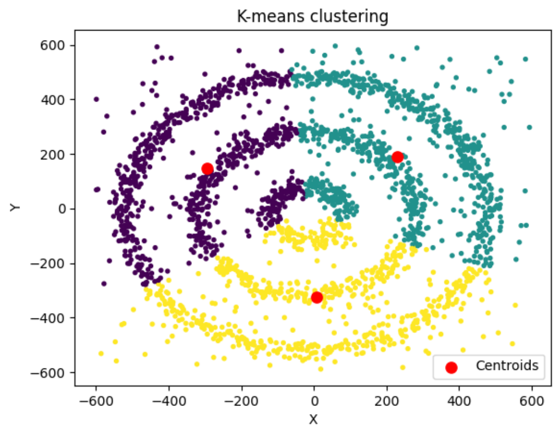


Figure 29: K means on dataset 2 using k=3

To improve clustering by K-means:

(a) We can choose different number of clusters (change value of K).

(b) Use a different metric in K-means instead of using Euclidean distance.

(c) Reduce the dimensions from 2D to 1D using a common dimensionality reduction method and apply KNN to the reduced dataset.

(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:
Scatter plot for dataset 3:

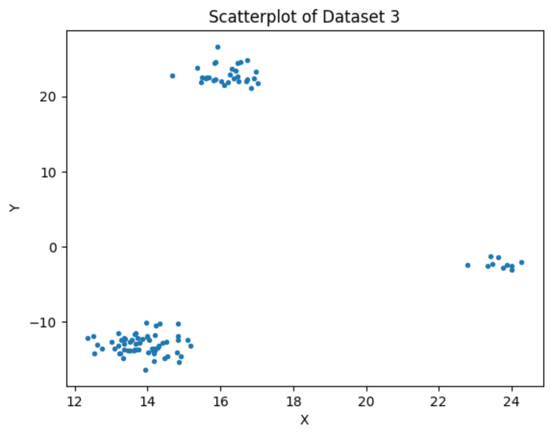


Figure 30: Scatter Plot for dataset 3

Elbow curve:

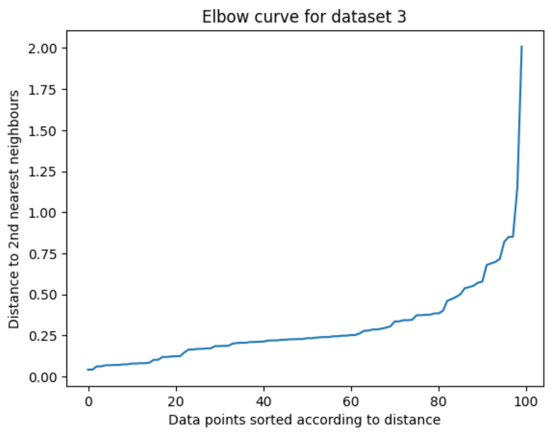


Figure 31: Elbow curve for dataset 3

As we can see in the elbow curve the suitable range of values for epsilon is between 0.8 and 3

We plot graph with min points = 15 and epsilon between 0.8 and 3

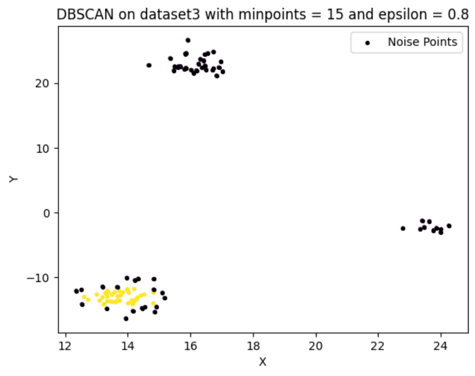


Figure 32: Scatter Plot with min points = 15 and epsilon = 0.8

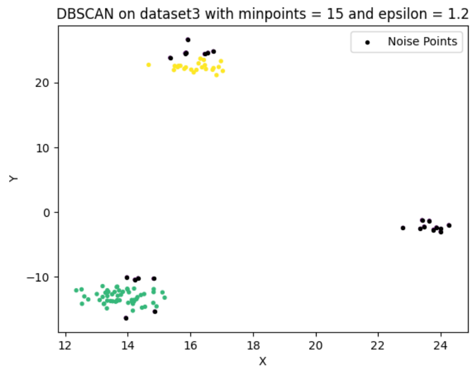


Figure 33: Scatter Plot with min points = 15 and epsilon = 1.2

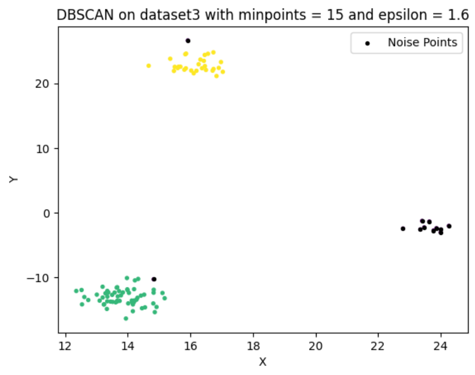


Figure 34: Scatter Plot with min points = 15 and epsilon = 1.6

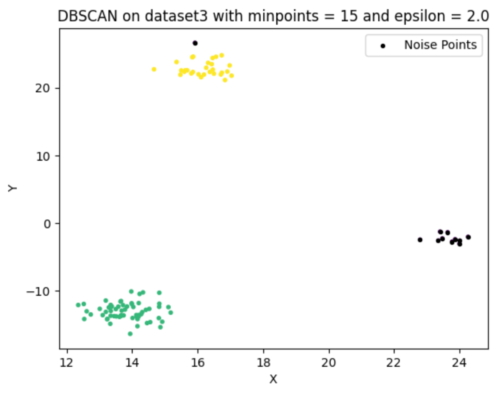


Figure 35: Scatter Plot with min points = 15 and epsilon = 2.0

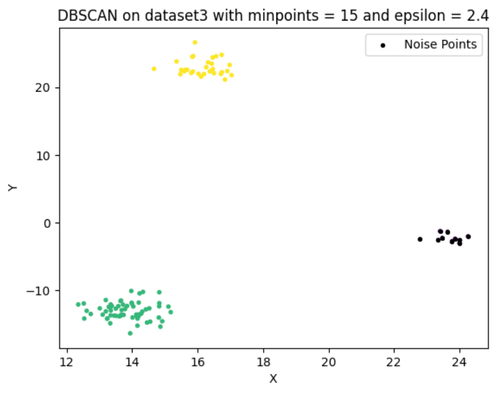


Figure 36: Scatter Plot with min points = 15 and epsilon = 2.4

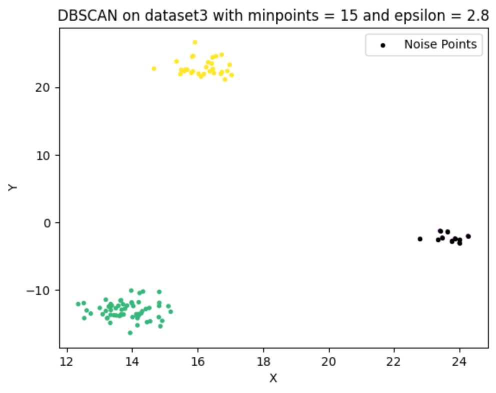


Figure 37: Scatter Plot with min points = 15 and epsilon = 2.8

We can see that we get visually good plots for epsilion between 2.4 and 2.8. Hence we can infer the optimal value of epsilon to be around 2.5

(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:

Scatter plot for KMeans with K=3:

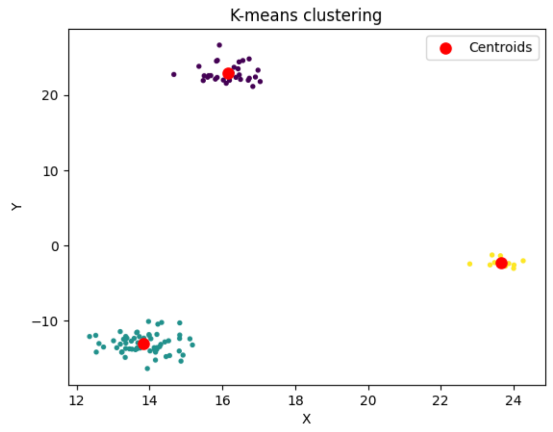


Figure 38: Scatter Plot for K means with K=3

Observation:

(a) As we can see in part (d), using DBSCAN, we get only 2 clusters and the rest points are marked as noise points when min points is equal to 15. This scatter plot is not optimal as the third class is all marked as noise points. This is due to very high min points value. Hence, we can also infer that the given initialization values were bad and minpoints should be set to a lower value.

(b) In part (e), using K means we get 3 clusters as required and the classification of data in these clusters is perfect. Hence, K-means using 3 classes is very good for classification of current data.

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

Solution:

(a) In DBSCAN, if we set min points to a high value a lot of points may get classified as noise points. Whereas in K-means we can get better classification for same if we know the number of classes the data can be classified into.

(b) In K-means we need to set the number of classes beforehand, whereas in DBSCAN, the algorithm decides the optimal number of classes.

(c) DBSCAN can handle noisy data and outliers more effectively as compared to K-means. K-means can be sensitive to placement of outliers.

(d) K-means works well when data is well-separated and clusters are roughly spherical and of similar size, whereas DBSCAN can identify clusters of any size and shape.

(e) DBSCAN required setting parameters such as the radius and minimum number of points for a cluster, which can be difficult to tune.

Conclusion:

Overall, K-means is a good choice for well-separated data and spherical clusters, while DBSCAN is a better choice for identifying clusters of arbitrary shape and size, handling noisy data and outliers, and when the number of clusters is not known in advance.

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.

   Google colab notebook solution for Q3 : here

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

Solution: For this part we applying GMM with a k-value of 2. Let's visualise the working of GMM for this value of k.

Scatter plot for dataset 4:

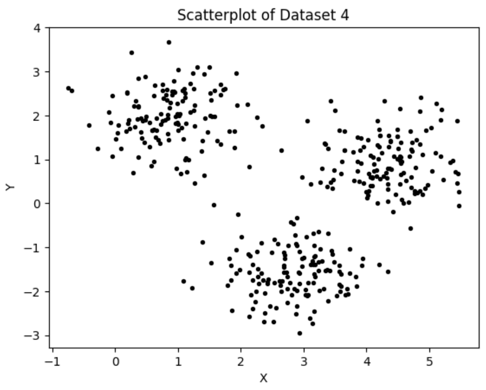


Figure 39: Initial scatter plot

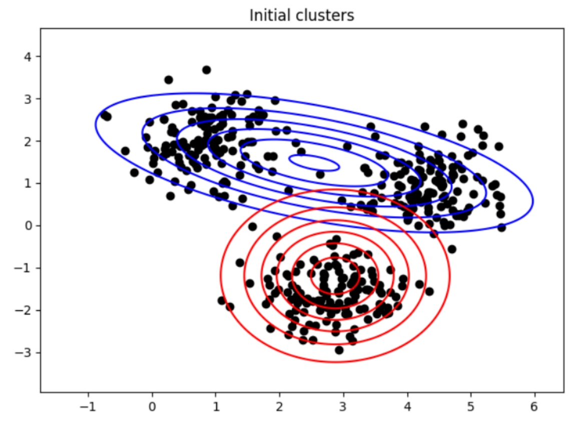


Figure 40: Initial clusters - 2

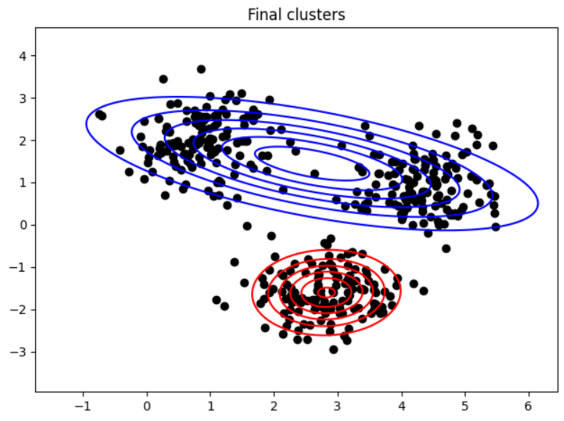


Figure 41: Final clusters - 2

As we can see, the classification is becoming more distinct from the initial to final clustering. Hence intuitively we deduce that the log-likelihood values must increase to a less negative value. Let's see if the deduction is correct.

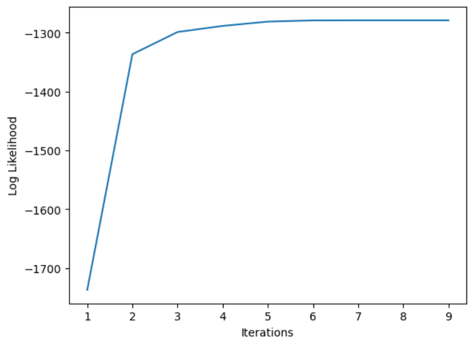


Figure 42: Log-Likelihood v/s n_iters - 2

As we imagined the log-likelihood increases with n_iters.

(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: First we see the visualisations for final clustering of all the values of k form 2 - 6. Scatter plot for dataset 4:

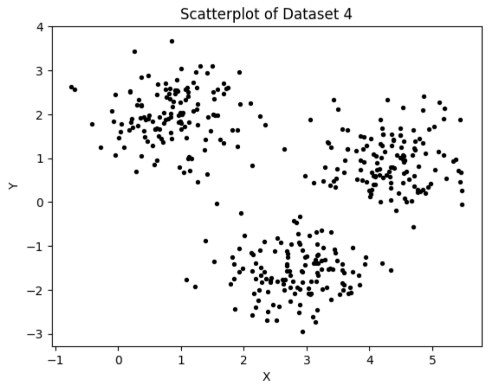


Figure 43: Initial scatter plot

- k = 2

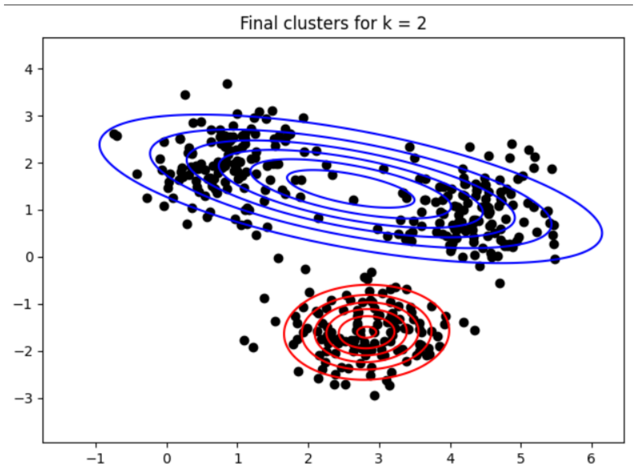


Figure 44: Final Clusters for k = 2

- k = 3

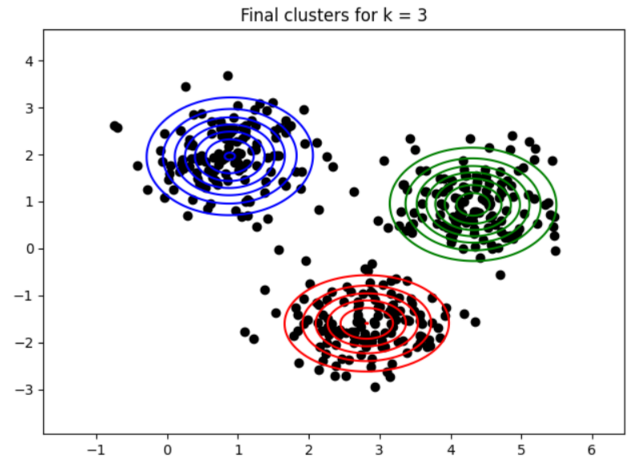


Figure 45: Final Clusters for k = 3

- k = 4

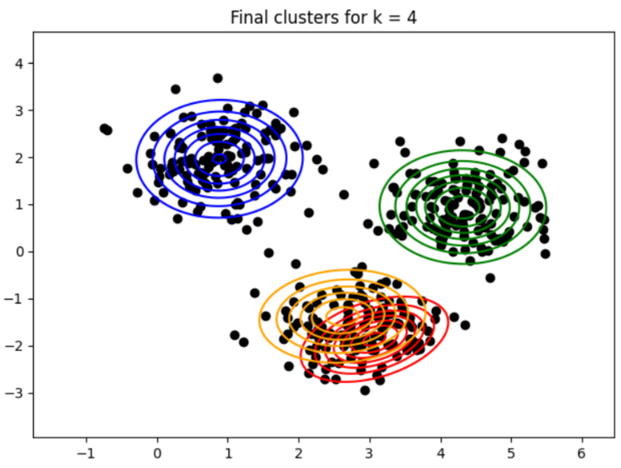


Figure 46: Final Clusters for k = 4

- k = 5

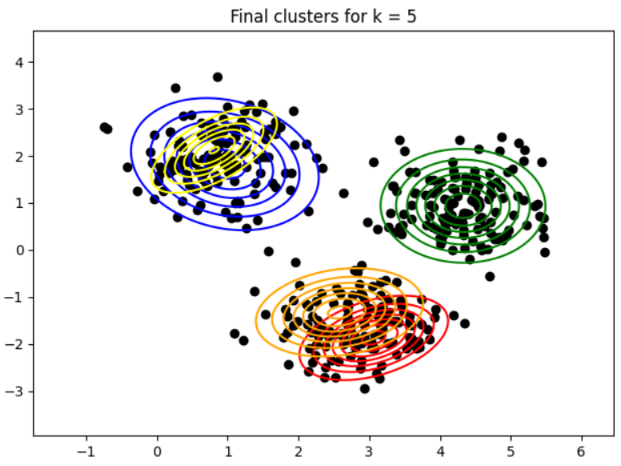


Figure 47: Final Clusters for k = 5

- k = 6

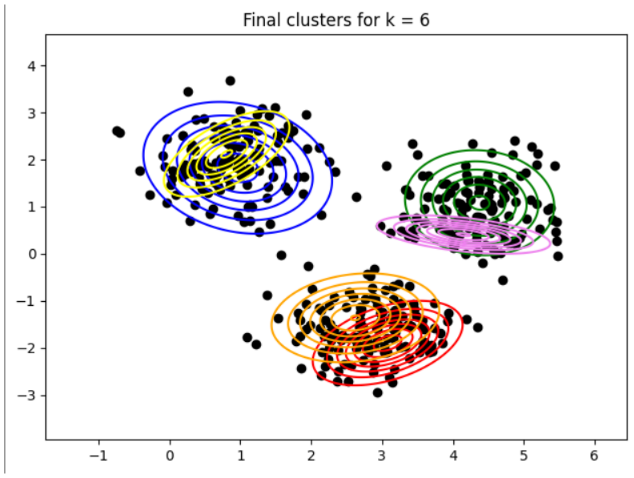


Figure 48: Final Clusters for k = 6

Now let's see the effect of increasing k on the log-likelihood values

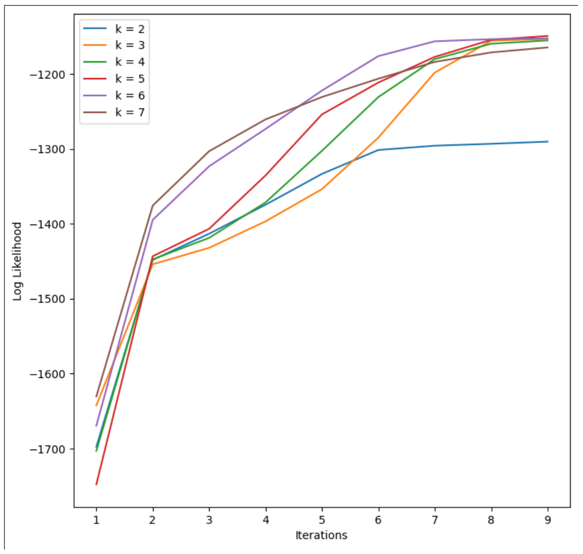


Figure 49: Log-Likelihood v/s n_iters as k increases

For our purposes of inference, we have only included iters till 9 as beyond that the rate of change of log-likelihood is negligible. Let's discuss our inferences from the graph given:

- The jump in log-likelihood is really big from 2 - 3, however after that the jump gets smaller and smaller.

- The log-likelihood at k = 3 and iter = 9 is greater than other values of k and it is actually lower for higher values of k

These inferences can be explained because after k = 3, as we increase the value of k, we begin to overfit the model more and more. Hence, the log-likelihood score gets worse.

(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: We have already discussed a subpart of this in the above part. Now we present more evidence to show that the optimal k value is 3.
1) AIC-Score and BIC-Score - These are very solid methods to score and select a model. For both evaluation criteron, the lower the better.

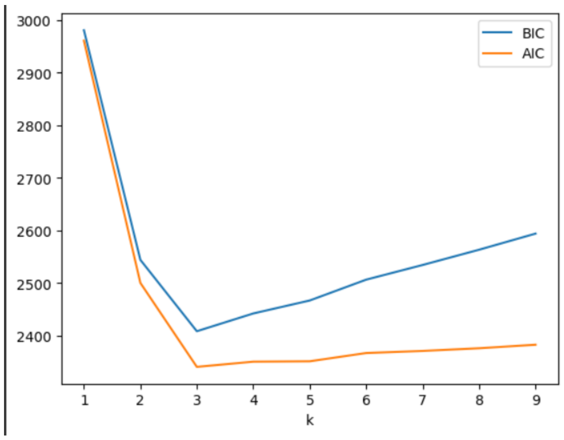


Figure 50: AIC, BIC v/s k

We can clearly see that k = 3 has the lowest measure in both AIC and BIC
2) Silhouette Score - It is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. The higher the score, the better.

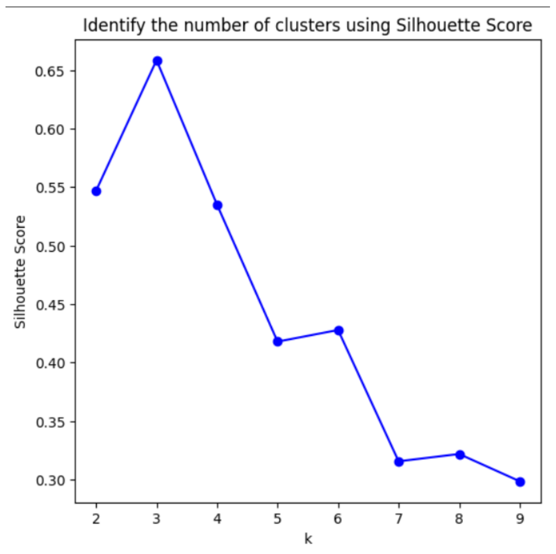


Figure 51: silhouette score v/s k
Again we see that k = 3 has the best measure in this metric. Hence we can safely conclude that k = 3 is the optimal value for k.