CS5691: Pattern Recognition and Machine Learning
Assignment #2

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

Teammate 1: (Sooraj Srinivasan) (50% of contribution) **Roll number:** CS20B075
Teammate 2: (Soham Tripathy) (50% of contribution) **Roll number:** CS20B073

- **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.**

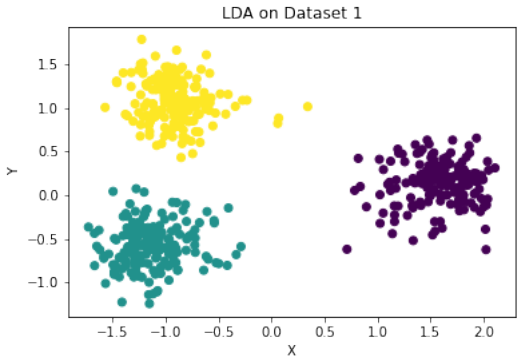
   (a) (2 marks) Use Linear Discriminant analysis (LDA) to convert dataset1 into the two-dimensional dataset and then visualize the obtained dataset. Also, perform an analysis on how results will change if we perform normalization (i.e., zero mean, unit variance normalization) on the initial dataset before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

   Solution:

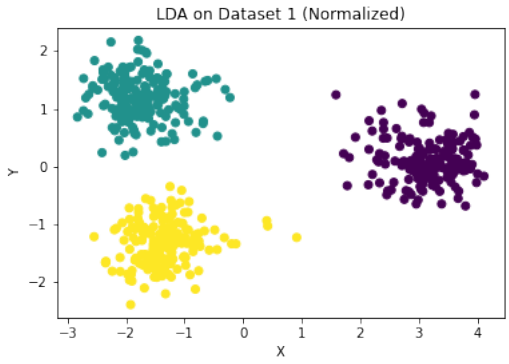
   To implement LDA, we used the eigenvalue method to calculate the weights. The procedure used was:

   - Calculate the scatter-within, scatter-between matrices of the given dataset.
   - Use that to calculate the eigenvalues and eigenvectors of $(S_w)^{-1}S_b$ where S_w is the scatter within matrix and S_b is the scatter between matrix.
   - Sort the eigenvectors based on the eigenvalues and take the top k (here 2).
   - The matrix formed by appending the eigenvectors is your weight matrix, which you can then multiply with the features to get the reduced features.

   Doing this process gave this as the output for Dataset 1:



   After normalising the data, we perform the LDA again. Now, we get:

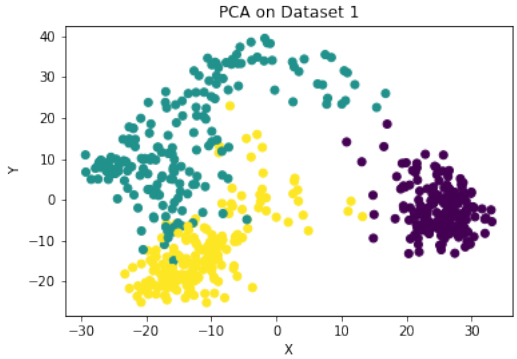


We see that there is no difference between the unnormalised data's LDA and the normalised data's LDA. This shows that normalisation has no effect on the LDA.

(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 `sklearn` to import `PCA` and perform the principal component analysis on the given dataset. On doing so, we get:



We see that the class separation in PCA is much worse than the class separation in LDA. This is to be expected as the LDA is calculated with the knowledge of the labels of the dataset and tries to increase the class separation through the $S_w^{-1}S_b$ which tries to increase the between class separation and decrease the within class separation.

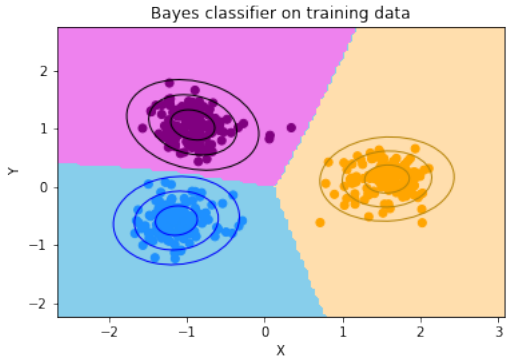
(c) (1.5 marks) Randomly shuffle and split the obtained dataset from part (a) into a training set (80%) and testing set (20%). Now build the Bayes classifier using the training set and report the following:

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

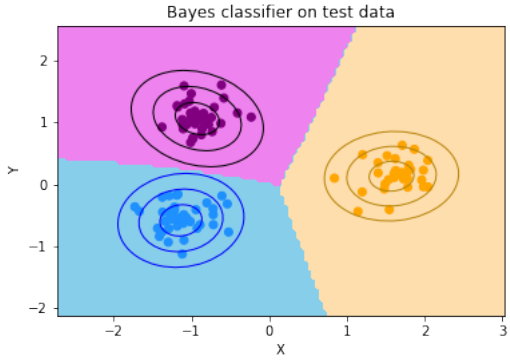
- confusion matrices on both train and test data.

Solution:

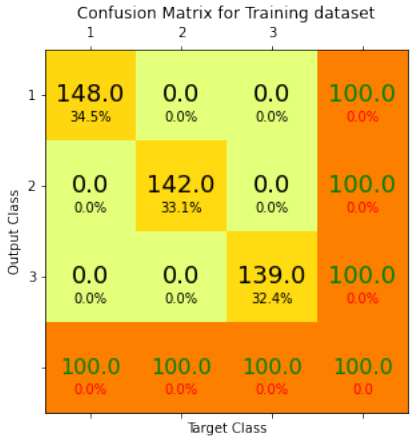
We randomly shuffle the dataset into training and test dataset with 80/20 split. Using this, we build a Naive Bayes classifier (with different covariances) to classify the data. We acheive 100% accuracy on both the train data and the test data. The decision boundary is shown below:



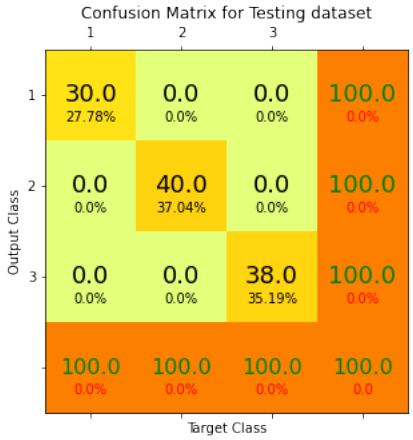
For the test data, we have the plot as this:



The confusion matrices for the train data is shown below:



The confusion matrix for the test data is shown below:

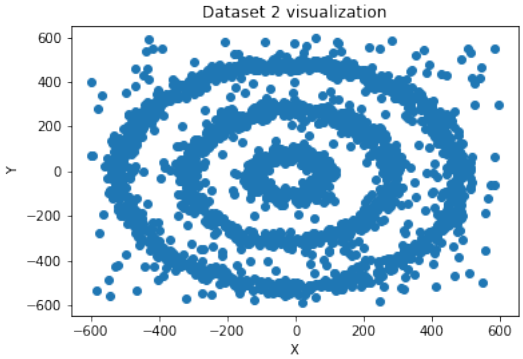


2. **[DBSCAN]** In this Question, you are supposed to implement **DBSCAN algorithm from scratch** on dataset2 provided here and dataset3 provided here. You also need to compare and contrast your observations from above with K-Means applied on both datasets. **However, you can use predefined libraries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

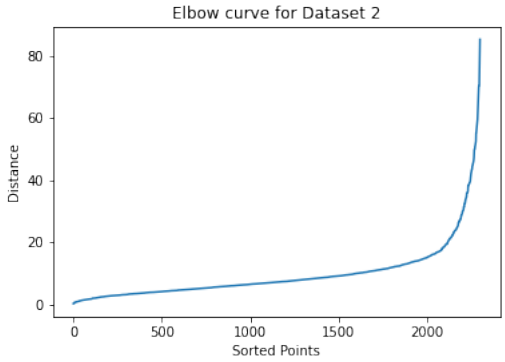
   (a) (1 mark) Visualize the data in dataset2. Then, find a suitable **range of values for epsilon** (a hyperparameter in DBSCAN algorithm) by using the 'Elbow Curve' of Datapoints plotted between K-Distance vs Epsilon. For simplicity, take only integer values for epsilon. **You can use predefined libraries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

Solution:

The visualisation for dataset 2 is shown below:



Using the `NearestNeighbour` class from `sklearn`, we were able to implement K-distance and plot the elbow curve. For dataset 2, the elbow curve is given below:



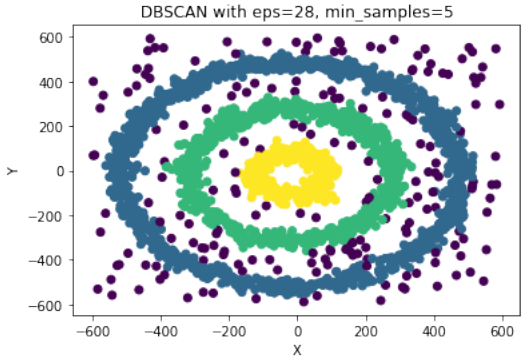
(b) (2 marks) Implement DBSCAN with the above suitable range of values of epsilon and detect the optimal value of epsilon, which gives the best clustering visually on the dataset. Show a visualization of the clusters formed for the best value of epsilon.

Solution:

We implement DBSCAN as follows:

- Maintain an array of visited nodes for the dataset.
- Find out the neighbours of a given node using the `eps` provided.
- If the number of neighbors is less than `minPts`, we know that it has to be a noise point or a border point, so we assign the label to the point as -1 and let other updates change it to a border point if necessary.
- If not, then we can generate the cluster. For each neighbor, we look at its neighbours and repeat the process. But this time, if the number of neighbors is less than `minPts`, then we assign it to the current cluster number.

From the elbow curve, we got the best values to be around 25-35. Using those values, we plot the DBSCAN clustering with `minPts=5` for those values. Out of these graphs, we chose the best one to the graph at `eps=28`.

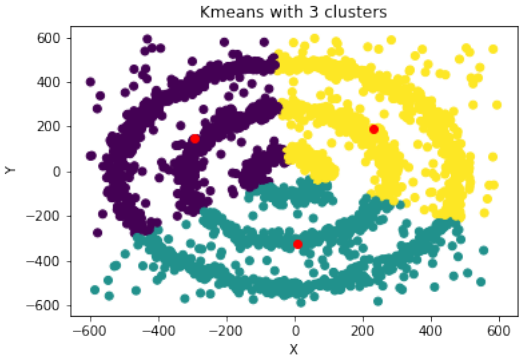


We see that the number of clusters here (excluding the noise "cluster") is 3.

(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:

We have implemented KMeans from scratch. In the previous sub part, we saw that the number of clusters in DBSCAN is 3. Using that as our value of `num_cluster`, we get this as our clustering:

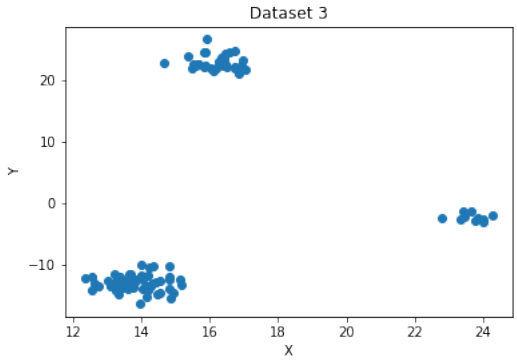


Here, the red points plotted are the means computed by the KMeans algorithm.

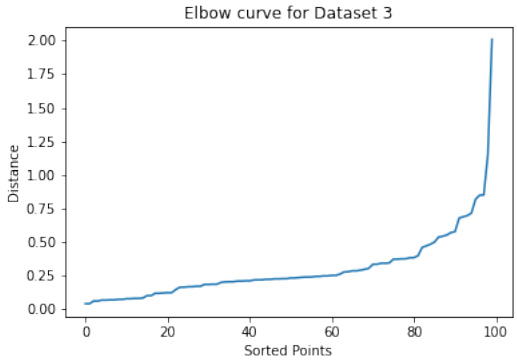
(d) (1.5 marks) Show a visualization of the data in dataset3. Use your implementation of DBSCAN with `minPts=15` on dataset3. Plot 'Elbow curve' to get an optimal range of values for `eps`. Detect the optimal value of epsilon which gives the best clustering visually on the dataset. Show a visualization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

Solution:

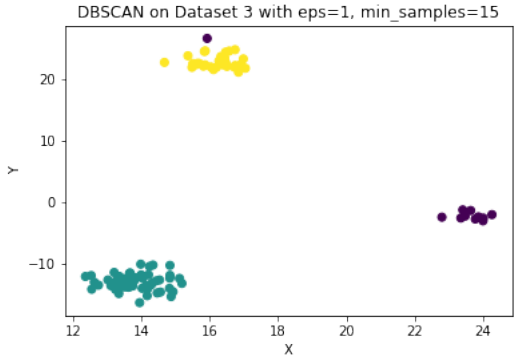
The visualisation of the dataset 3 is shown below:



Plotting the elbow curve for this dataset as described in part (a), we get:



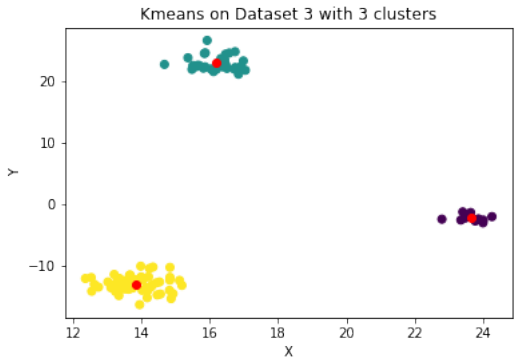
From the elbow curve, we see that the elbow of the curve lies at around 1.
Using DBSCAN with `eps=1`, we get this as the clustering:



(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:

Performing KMeans with K=3, we get this as our clustering:



The red points represent the means of the clusters.

We see that K-Means is much better at clustering this dataset as compared to the DBSCAN clustering. The number of clusters in DBSCAN is 2, which would not be the best fit for this dataset. Moreover, DBSCAN classifies some of the points as noise whereas K-Means does a better job of clustering as well.

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

Solution:

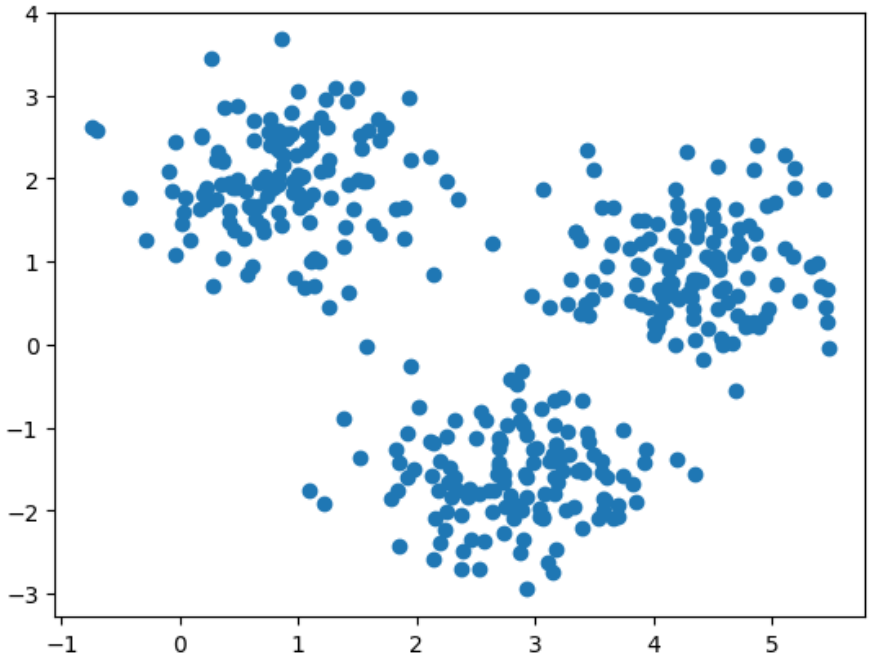
3. **[GMM]** In this question, you are supposed to implement the Expectation-Maximization algorithm for Gaussian mixture models on the given dataset4. The data can be found here.

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

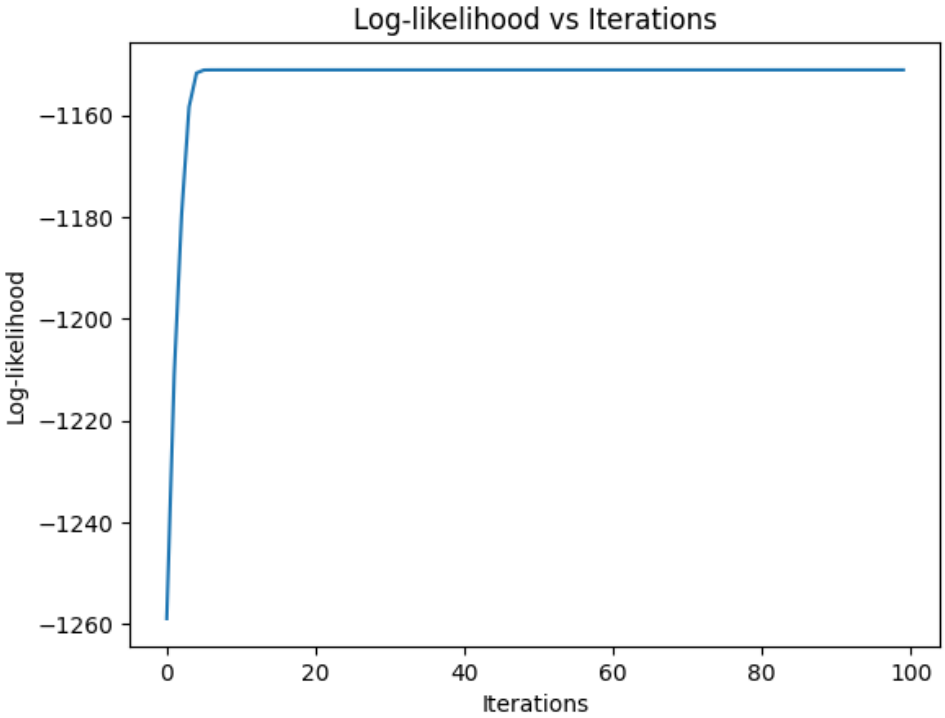
Solution:

GMM Implementation - Shared as Q3.ipynb file

The scatter-plot of the given data points is given below:



The log-likelihood (for K = 3) as a function of iterations is given below:

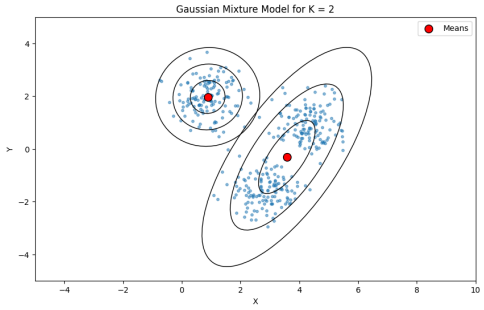


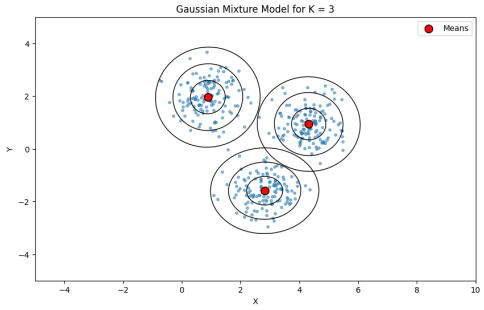
(b) (2 marks) Run EM for different numbers of Gaussians (k)(Try 2,3,4,5,6). Plot figures that can help in visualization and also log likelihood as a function of iteration for different

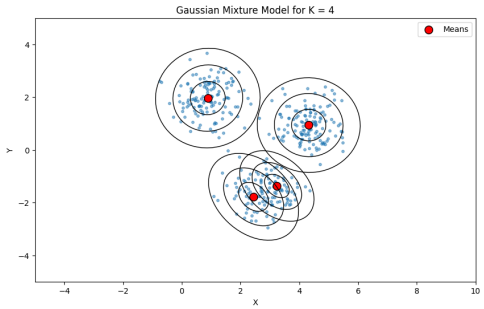
values of k. Report the observations.

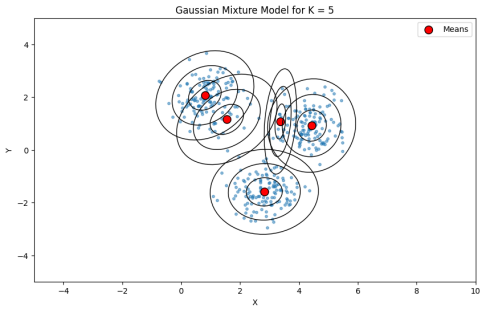
Solution:

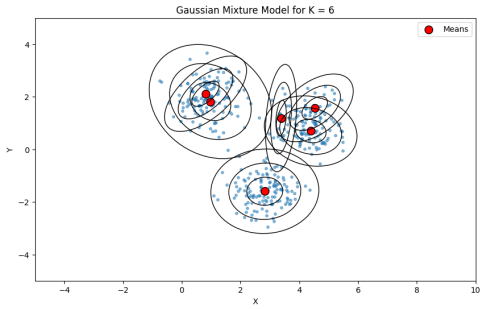
We ran the EM for K from 2 to 9 and obtained the following results.

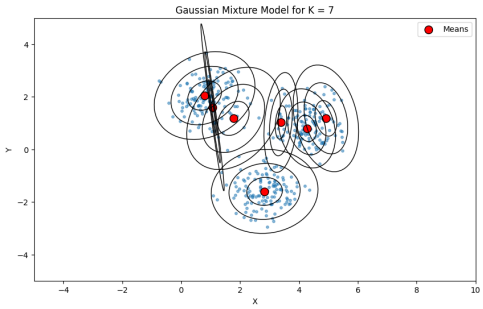


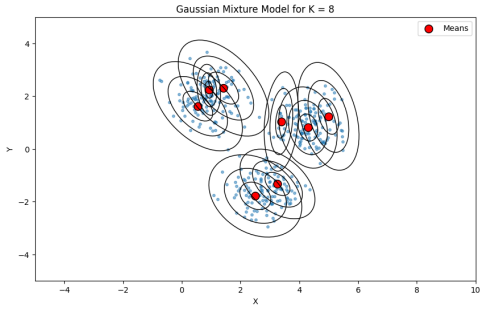


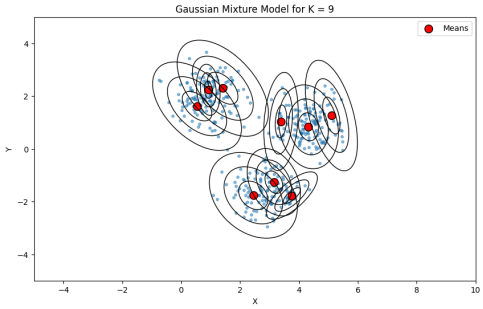




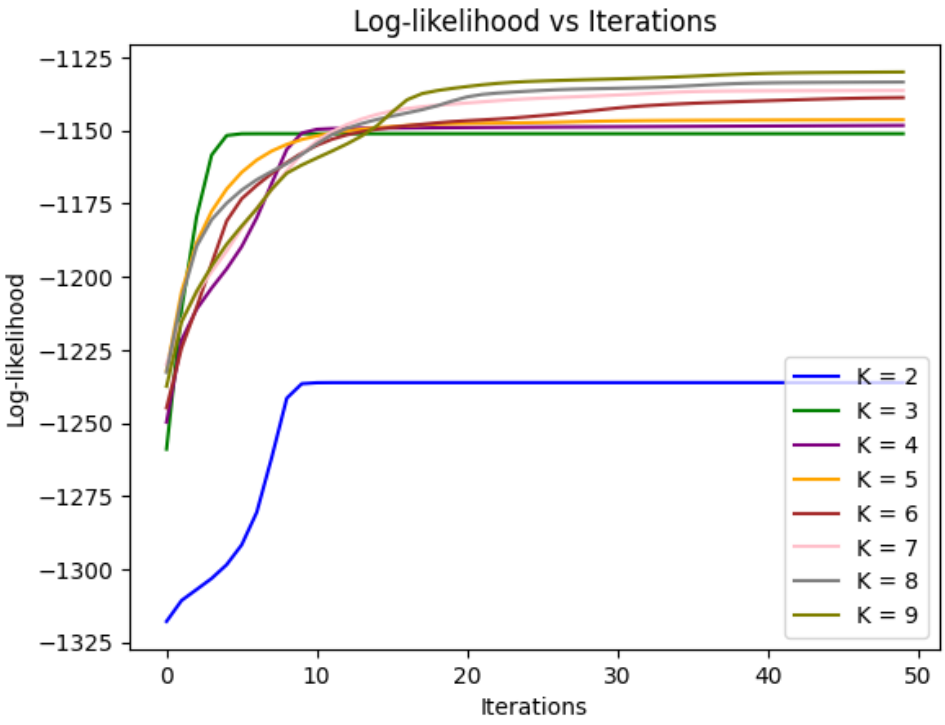








The log-likelihood function v/s iterations graph for various values of K is given below.



Observations

1) Convergence: The log-likelihood should increase with each iteration of the EM algorithm until it converges to a local maximum. This is indicated by a plateau in the log-likelihood curve, where the slope becomes small or negligible. The number of iterations required for convergence depends on the complexity of the GMM and the size of the dataset.

2) Initialization: The choice of initialization for the GMM parameters can affect the convergence of the algorithm and the quality of the final solution. Random initialization can sometimes result in the algorithm converging to a suboptimal solution, while a better initialization can help the algorithm converge faster and to a better local maximum.

3) Log-likelihood: The log-likelihood vs iterations graph can be useful for comparing different GMMs with different numbers of components. As we increase the value of K in our GMM model, it converges to a higher log-likelihood value, indicating a better fit to the data. However, it is essential to balance model complexity with goodness-of-fit and avoid overfitting. This phenomenon can be seen in the graph.

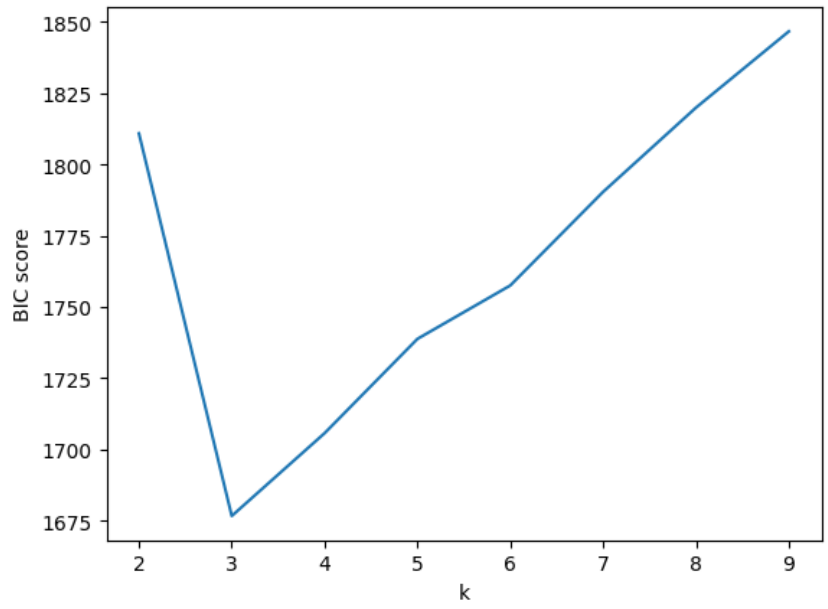
(c) (2 marks) Find the optimal k. There are several metrics like Silhouette score, Distance between GMMs, and Bayesian information criterion (BIC), or even you can use log-likelihood from the last question to infer. Give a clear explanation for your decision.
Note: **You can use third-party libraries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

Solution:

SILHOUETTE SCORE:

The Silhouette score measures the similarity of each data point to its assigned cluster, and ranges from -1 to 1, with higher scores indicating better clustering. A high Silhouette score indicates that the data points within each cluster are tightly packed, and well separated from other clusters. In the case where K=3, the Silhouette score for each data point is relatively high, indicating that each cluster is well-defined and compact.

BIC SCORE:



On the other hand, the BIC score measures the tradeoff between model complexity and goodness-of-fit to the data. Lower BIC scores indicate a better tradeoff between these two factors. A GMM model with three components provides a good balance between capturing the underlying structure of the data and avoiding overfitting.

Therefore, based on the Silhouette and BIC scores, a GMM model with three components is the optimal choice for this dataset. This means that a GMM model with three components provides the best balance between model complexity and goodness-of-fit to the data.