K-Means Clustering from Scratch (NumPy Only)
Project Report

1. Project Objective

The target of this project is to develop the **K-Means clustering algorithm from scratch**, and the numerical computations are handled using the **NumPy library**. The **K-Means algorithm** implementation will not make any use of **Scikit-learn's K-Means class**. This project is meant to improve the knowledge of the iterative optimization process used in the **K-Means algorithm.**

2. Dataset Generation

Generate a **synthetic two-dimensional dataset using NumPy**; the dataset should be such that it contains several salient, distinct clusters-ideally between three to five clusters-generated by taking a sample of points from **Gaussian distributions** and shifting them to different centers. This makes **the cluster structure visually clear and suitable for evaluation.**

3. K-Means Algorithm Implementation (From Scratch)

The **K-Means algorithm** is implemented through the following steps

- **Initialization:** Choose K unique points from the data set randomly as initial centroids.
- **Assignment Step:** To perform the assignment step, the appropriate data point needs to be assigned to the nearest centroid based on the data.
- **Update Step:** Recompute each centroid as the mean of all points that are assigned to that cluster.
- **Iteration and Convergence:** Iterate the assignment and centroids calculations until convergence or a desired tolerance level, or the specified number of iterations, is reached.
- **Empty Cluster Handling (recommended):** In case an empty cluster occurs, reinitialize its centroid to an arbitrary data point.

4. Experiments with Different Values of K

Once the algorithm has been implemented, the **K-Means algorithm** can be used to test the **dataset for different values of K**, **i.e.,** for **K=2, 3, 4, 5, and 6.** This is done to see how the **value of K** impacts the overall partitioning of the data.

5. Model Evaluation using Silhouette Score

Each **value of K** is subsequently assessed for **clustering performance** using the **Silhouette Score**, which reflects the extent to which an object is similar to objects within its cluster relative to objects within other clusters-in terms of **intra-cluster cohesion and inter-cluster separation.**

The calculation of the **Silhouette Score** can be done **manually or computed by using sklearn.metrics.silhouette_score,** but **Scikit-learn** should be used only for **evaluation, not for the K-Means clustering logic.**

6. Visualizations

Such plots are generated in the project.

- **Scatter plot** of the generated synthetic dataset
- **Line plot** of Silhouette Score vs K (K = 2 to K = 6).
- The **scatter plot** of the final clustering result for the optimal K, including centroid markers

7. Interpretation and Discussion

The results are evaluated, and it can be done using the Silhouette Score and the **visual representation of clusters for different values of K**. When **K is less than the appropriate number,** the model will be less fitted, meaning it will not be able to form different classes into different clusters. When **K is very large**, it will be considered as if the model is **over-segmenting classes, or the data points are being split**. Therefore, the **optimal value of K** will be the one that produces the **highest Silhouette Score** and visually makes the most sense in terms of **natural groupings.**

8. Expected Deliverables

- A **Jupyter Notebook (.ipynb)** containing the complete code for implementing the K-Means algorithm
- **Silhouette Scores printed for K = 2 to 6.**
- All required plots (**dataset, silhouette vs K, best K clustering result**).
- A written interpretation of how **K** affects **the clustering and Silhouette Scores**.