# Lab Assignment-4

**Q.1** Implement fuzzy_c_means (X, c, $m$=2.0, max_iter=100, tol=1e-5) from scratch in Python (do NOT use external libraries like fcmeans).

Algorithm steps:

1. Initialize a random membership matrix $U \in \mathbb{R}^{N \times c}$, ensuring each row sums to 1.
2. Update cluster centers $V$ using the weighted average data points.
3. Update membership values based on distances and fuzzifier $m$.
4. Stop when the maximum change in $U$ is less than tol or max_iter is reached.
5. Return the final cluster centers, the membership matrix, and the number of iterations used. Test the function on a 2-D blobs dataset (sklearn.datasets.make_blobs, 3 clusters, 300 points) and plot:
    - The data points colored by their highest membership
    - The Cluster centers

**Q.2** Apply your fuzzy_c_means() implementation to a grayscale image segmentation task.

Steps:

1. Load a grayscale image (e.g., from Pillow) and flatten it to a 1-D array.
2. Apply FCM with 3 clusters.
3. Assign each pixel to the cluster with highest membership and reconstruct the segmented image.
4. Display the original and segmented images side-by-side.
5. Report the runtime and mean squared difference between original and segmented image.

**Q.3** Investigate the effect of fuzzifier m on clustering.

For a fixed dataset (make_blobs with 3 clusters):

1. Run your FCM for m = 1.5, 2.0, 2.5, 3.0.
2. For each, report:
    - Final objective function value
    - Number of iterations
    - Plot of data colored by dominant membership

Write a short analysis: how does increasing m affect membership sharpness and convergence?

**Q.4** Use Partition Coefficient (PC) and Partition Entropy (PE) as validity indices.

Implement:

1. partition_coefficient(U) = $\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} u_{ij}^{2}$
2. partition_entropy (U) = $\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} u_{ij} \log u_{ij}$

For k = 2 … 6, run FCM on the Iris dataset (load_iris()) and plot PC and PE vs. k on a single plot (one curve increasing, one decreasing).
Select and report the best k based on both indices.

**Q.5** Perform document clustering with FCM using TF-IDF vectors.
Steps:

1. Load up to 3,000 documents from sklearn.datasets.fetch_20newsgroups (select “rec.autos”, “sci.space”, “talk.politics.misc”).
2. Compute TF-IDF vectors (TfidfVectorizer) and apply dimensionality reduction (TruncatedSVD, 50 components).
3. Run FCM with c = 3.
4. Assign cluster labels by highest membership and compare against true categories using Adjusted Rand Index (ARI).
5. Report ARI and display a bar chart of average membership per cluster.