

Machine Learning-LAB (LAB 06)

**Objectives:**

* Understand how clustering algorithms work internally.
* Implement K-Means and K-Medoids from scratch.
* Use Scikit-Learn's KMeans implementation.
* Compare clustering results, inertia (cost function), and run-time.

**Note: Carefully read the following instructions (***Each instruction contains a weightage***)**

1. There must be a block of comments at start of every question's code by students; the block should contain brief description about functionality of code.
2. Comment on every function and about its functionality.
3. Use understandable name of variables.
4. Write a code in PYTHON language and you may use any of IDE or Notebook environment.
5. First think about the problems statements then you may start your programming.
6. At the end when you done your tasks, attach. ipynb files on google classroom. Paste your complete code in word file along with output **(Make sure your submission is completed)**. In case of missing any file, marks will be deducted.
7. Please submit your file in this format **19F1234\_L01**.
8. Email submission is not accepted.
9. Submission from home will not be accepted.
10. Use of GenerativeAI tools or copy code from any source otherwise you will be penalized with negative marks.
11. **YOUR MARKING WILL BE BASED ON PRIOR SUBMISSION OF YOUR CODE BEFORE DEADLINE AND VIVA.**

**Part 1: K-Means from Scratch**

**Implement K-Means manually:**

* Initialize K random centroids.
* Assign each point to the nearest centroid.
* Recompute centroids as the mean of assigned points.
* Repeat until convergence (no or minimal centroid movement).

**Part 2: K-Medoids from Scratch**

**Implement K-Medoids manually:**

* Initialize K random medoids (data points, not means).
* Assign points to nearest medoid.
* For each cluster, try swapping medoid with non-medoid points to minimize total distance.
* Stop after no improvement.

**Part 3: Using Scikit-Learn**

* Use from sklearn.cluster import KMeans.
* Fit on the same dataset.
* Report number of iterations, final inertia.

(For K-Medoids, use scikit-learn-extra if available:  
from sklearn\_extra.cluster import KMedoids, or just skip sklearn K-Medoids if unavailable.)

**Part 4: Comparison**

* Compare results:
  + Number of iterations to converge
  + Final clustering cost (inertia / total distance)
  + Visual comparison (scatter plots)
* Discuss differences:
  + Why might K-Medoids be more robust?

**Deliverables:**

* Complete Python script (.py) or Jupyter notebook (.ipynb).
* Must include:
  + Manual K-Means
  + Manual K-Medoids
  + scikit-learn KMeans
  + scikit-learn K-Moids
  + Comparisons
  + Short written answer for comparison observations.

**Output Files**

* lab-kmeans-kmedoids.ipynb or lab-kmeans-kmedoids.py
* Students' screenshots/plots + written discussion

I will also give you a **starter code with TODOs** which you can fill!

TASK 2:

* Apply multiple unsupervised clustering algorithms to real-world data.
* Visualize and compare cluster structures and performances.
* Analyze linkage strategies in Agglomerative Clustering.
* Use clustering metrics to evaluate performance without ground truth.
* Use ChatGPT to improve reasoning, find edge cases, and interpret results.

Use **Wholesale customers** dataset from UCI repository.

# Install package

!pip install ucimlrepo scikit-learn matplotlib seaborn scikit-learn-extra

# Load the dataset

from ucimlrepo import fetch\_ucirepo

wholesale\_customers = fetch\_ucirepo(id=292)

import pandas as pd

X = wholesale\_customers.data.features

**Part 1: Preprocessing**

1. Normalize the data using StandardScaler or MinMaxScaler.

**Part 2: Apply Clustering Algorithms**

1. K-Means
2. K Medoids
3. Agglomerative Clustering
   1. 'single'
   2. 'complete'
   3. 'average'

**Part 3: Visualization**

**Part 4: Evaluation (Even Without Ground Truth)**

Use internal metrics:

* Silhouette Score

**Part 5: Provide answer to following questions with justification.**

Q1. Which clustering algorithm gave the best silhouette score?

Q2. Compare the shapes of the clusters in all visualizations. Which method seems more sensitive to outliers?

Q3. Which linkage in Agglomerative Clustering works better for this dataset? Why?

Q4. Between K-Means and K-Medoids, which one is more robust to noise/outliers?