

## VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

### Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC; Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE An ISO 9001:2015 Certified Institution

**SCHOOL OF ENGINEERING & TECHNOLOGY**

# B.Tech Programme: CSE

Course Title: Unsupervised Learning

Course Code: ML 465P

**Submitted By:**

**Name: Ishaan Jain**

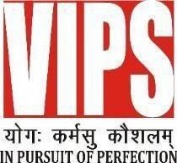
**Enrollment No: 06117702722**

**Branch & Section: CSE B**

# **Submitted To:**

**Ms. Khusboo Vijayran**

**Assistant Professor**



## VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

## MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building

capabilities to enable them to make significant contributions

**PROGRAM NO.-01**

**AIM-**

Implement and visualize various Distance Metrics in Unsupervised Learning on a synthetic dataset.

**BACKGROUND-**

In unsupervised learning, the notion of similarity and dissimilarity plays a crucial role in grouping and analyzing unlabeled data. Since the dataset lacks predefined class labels, clustering and other unsupervised methods rely on distance metrics to measure how close or far data points are from one another.

A distance metric is a mathematical function that quantifies the dissimilarity between two vectors in a feature space. Different distance measures can capture different relationships in the data, and the choice of metric can significantly impact the results of clustering or dimensionality reduction techniques. For instance, Euclidean distance emphasizes absolute differences in space, while Manhattan distance focuses on axis-aligned differences. Cosine similarity, on the other hand, measures orientation rather than magnitude, making it useful in text or high-dimensional data.

Synthetic datasets are often used for experimentation because they allow controlled scenarios where patterns are known beforehand. By applying and visualizing various distance metrics on such data, we can better understand how these measures influence clustering outcomes and the structure of the data.

This experiment demonstrates the implementation of different distance metrics and their effect on data visualization in unsupervised learning, providing insights into when and why a particular metric should be chosen.

**ALGORITHM-**

1. **Import Required Libraries**
   1. Import numpy, matplotlib, and sklearn.datasets for dataset generation.
   2. Import scipy.spatial.distance or sklearn.metrics for distance computations.
2. **Generate Synthetic Datasets**
   1. Use make\_blobs() to create a dataset with spherical clusters.
   2. Use make\_moons() to create a non-linear, crescent-shaped dataset.
3. **Preprocess Data (Optional)**
   1. Apply normalization or standardization if needed, to ensure fair distance comparisons.
4. **Define Distance Metrics:** Implement various distance measures such as Euclidean distance, Manhattan distance, Cosine similarity, etc.
5. **Compute Pairwise Distances:** Calculate pairwise distances between all data points for both datasets using the chosen metrics.
6. **Visualize Results**
   1. Plot the datasets (make\_blobs and make\_moons) with distance relationships.
   2. Use scatter plots, distance heatmaps, or color coding to show how distances vary with different metrics.
7. **Comparison and Analysis**
   1. Compare the effect of different metrics on both datasets.
   2. Note how distance metrics perform differently on linearly separable (make\_blobs) vs. non-linear (make\_moons) data.

**CODE-**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.metrics.pairwise import euclidean\_distances, manhattan\_distances, cosine\_similarity

import pandas as pd

# 1. Generate synthetic dataset (make\_blobs)

X, y = make\_blobs(n\_samples=100, centers=3, n\_features=2, random\_state=42)

# 2. Choose a reference point (first sample)

ref\_point = X[0].reshape(1, -1)

# 3. Compute pairwise matrices

edist\_matrix = euclidean\_distances(X, X)

mdist\_matrix = manhattan\_distances(X, X)

csim\_matrix = cosine\_similarity(X, X)

# Convert to DataFrame for readability

edist\_df = pd.DataFrame(edist\_matrix)

mdist\_df = pd.DataFrame(mdist\_matrix)

csim\_df = pd.DataFrame(csim\_matrix)

# Print only first 5x5 block of each matrix

print("Euclidean Distance Matrix (first 5x5):\n", edist\_df.iloc[:5, :5])

print("\nManhattan Distance Matrix (first 5x5):\n", mdist\_df.iloc[:5, :5])

print("\nCosine Similarity Matrix (first 5x5):\n", csim\_df.iloc[:5, :5])

# 4. Compute distances from reference point for visualization

edist = euclidean\_distances(ref\_point, X).flatten()

mdist = manhattan\_distances(ref\_point, X).flatten()

csim = cosine\_similarity(ref\_point, X).flatten()

# Normalize cosine similarity for visualization (0-1 range)

csim\_norm = (csim - csim.min()) / (csim.max() - csim.min())

# 5. Plot function

def plot\_metric(values, title, cmap):

    plt.scatter(X[:,0], X[:,1], c=values, cmap=cmap, s=80, edgecolor="k")

    plt.scatter(ref\_point[:,0], ref\_point[:,1], c="red", marker="X", s=200, label="Reference Point")

    plt.colorbar(label=title)

    plt.title(title)

    plt.legend()

    plt.show()

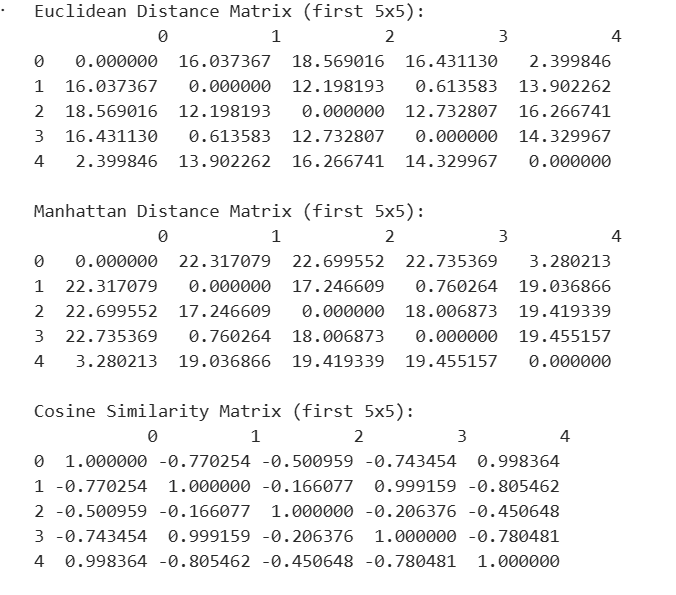
# 6. Visualizations

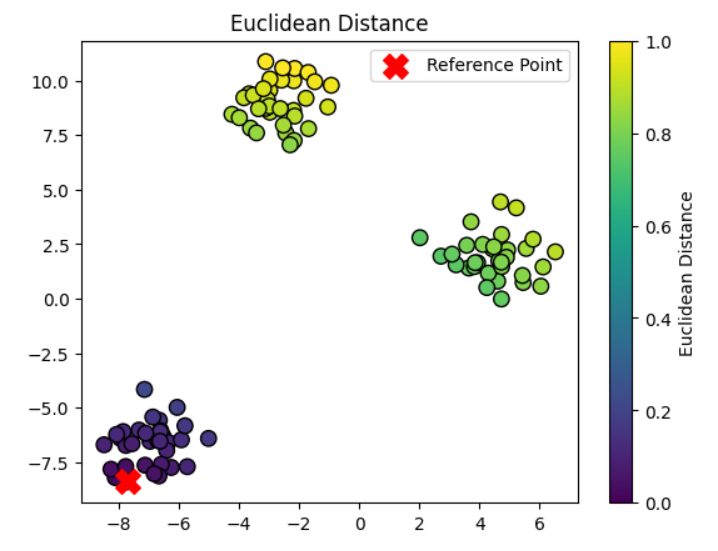
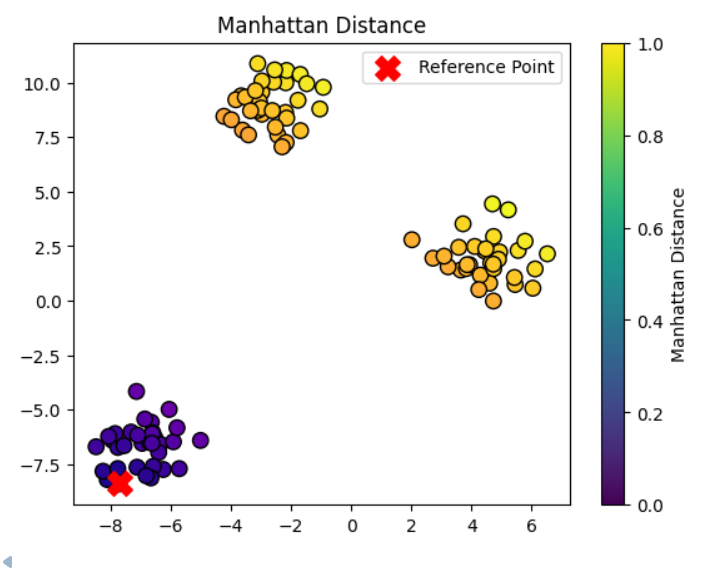
plot\_metric(edist, "Euclidean Distance", "viridis")

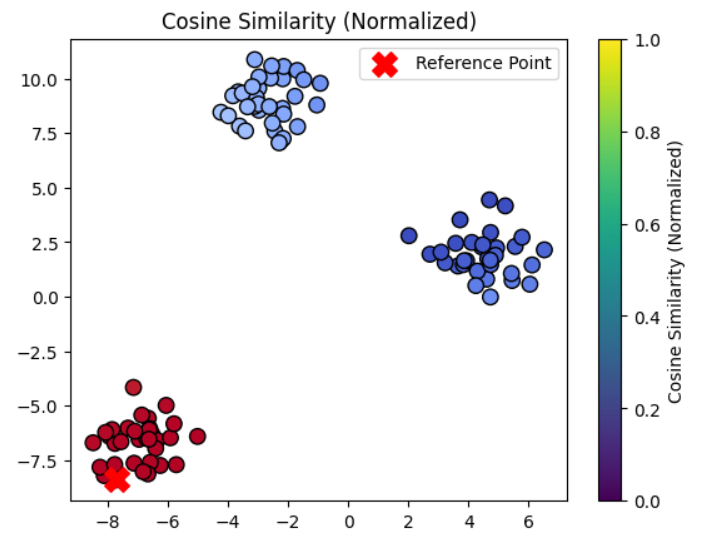
plot\_metric(mdist, "Manhattan Distance", "plasma")

plot\_metric(csim\_norm, "Cosine Similarity (Normalized)", "coolwarm")

**OUTPUT-**





**LEARNING OUTCOMES-**

**PROGRAM NO.-01**

**AIM-**

Write a program for clustering quality comparison k-means vs EM gaussian mixture models.

**BACKGROUND-**

In unsupervised learning, the notion of similarity and dissimilarity plays a crucial role in grouping and analyzing unlabeled data. Since the dataset lacks predefined class labels, clustering and other unsupervised methods rely on distance metrics to measure how close or far data points are from one another.

A distance metric is a mathematical function that quantifies the dissimilarity between two vectors in a feature space. Different distance measures can capture different relationships in the data, and the choice of metric can significantly impact the results of clustering or dimensionality reduction techniques. For instance, Euclidean distance emphasizes absolute differences in space, while Manhattan distance focuses on axis-aligned differences. Cosine similarity, on the other hand, measures orientation rather than magnitude, making it useful in text or high-dimensional data.

Synthetic datasets are often used for experimentation because they allow controlled scenarios where patterns are known beforehand. By applying and visualizing various distance metrics on such data, we can better understand how these measures influence clustering outcomes and the structure of the data.

This experiment demonstrates the implementation of different distance metrics and their effect on data visualization in unsupervised learning, providing insights into when and why a particular metric should be chosen.

**ALGORITHM-**

**1. K-Means Clustering Algorithm**

* Initialize the number of clusters kkk and randomly select kkk initial centroids.
* Assign each data point to the nearest centroid using a distance metric (typically Euclidean distance).
* Update the centroids by computing the mean of all data points assigned to each cluster.
* Repeat steps 2 and 3 until convergence (centroids no longer change or maximum iterations reached).
* Output the final cluster assignments and centroids.

**2. Gaussian Mixture Model (GMM) using Expectation-Maximization (EM) Algorithm**

* Initialize the number of clusters kkk, mean vectors, covariance matrices, and mixing coefficients for each Gaussian component.
* Expectation (E-step): Compute the probability (responsibility) that each data point belongs to each Gaussian component.
* Maximization (M-step): Update the parameters (mean, covariance, and mixing coefficients) of each Gaussian using the responsibilities calculated in the E-step.
* Repeat steps 2 and 3 until convergence (log-likelihood stabilizes or maximum iterations reached).
* Output the final cluster assignments based on the highest probability and the Gaussian parameters.

**CODE-**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

from sklearn.metrics import silhouette\_score, davies\_bouldin\_score

X, y\_true = make\_blobs(n\_samples=500, centers=4, cluster\_std=1.0, random\_state=42)

kmeans = KMeans(n\_clusters=4, random\_state=42)

kmeans\_labels = kmeans.fit\_predict(X)

kmeans\_silhouette = silhouette\_score(X, kmeans\_labels)

kmeans\_db = davies\_bouldin\_score(X, kmeans\_labels)

print("K-Means Silhouette Score:", kmeans\_silhouette)

print("K-Means Davies-Bouldin Score:", kmeans\_db)

gmm = GaussianMixture(n\_components=4, random\_state=42)

gmm\_labels = gmm.fit\_predict(X)

gmm\_silhouette = silhouette\_score(X, gmm\_labels)

gmm\_db = davies\_bouldin\_score(X, gmm\_labels)

print("GMM Silhouette Score:", gmm\_silhouette)

print("GMM Davies-Bouldin Score:", gmm\_db)

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

axs[0].scatter(X[:, 0], X[:, 1], c=kmeans\_labels, cmap='viridis', s=50)

axs[0].scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color='red', marker='x', s=200)

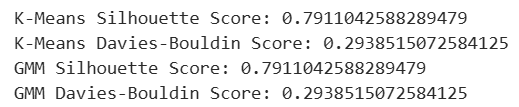
axs[0].set\_title("K-Means Clustering")

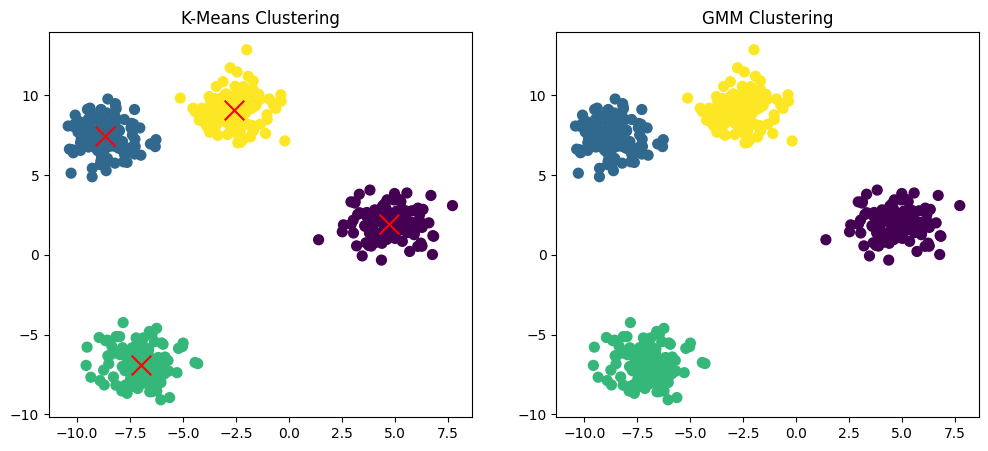
axs[1].scatter(X[:, 0], X[:, 1], c=gmm\_labels, cmap='viridis', s=50)

axs[1].set\_title("GMM Clustering")

plt.show()

**OUTPUT-**





**LEARNING OUTCOMES-**