Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	**TE**	**Semester:**	**V**
Course Code:	**CSC504**	**Course Name:**	**Data Warehousing and Mining**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**51**
Experiment No.:	**08**
Title of the Experiment:	**Implementation of any one clustering algorithm using languages like JAVA/ python.**
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Meet Expect Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Neha Raut

Signature :

Date:

Aim: To Study and Implement K-Medoids algorithm

Objective: Understand the working of K-Medoids algorithm and its implementation using Python.

Theory:
K-Medoids is a clustering algorithm that is very similar to K-Means. However, instead of choosing means (centroids) as the central point for each cluster, it chooses actual data points (medoids) to minimize the sum of dissimilarities between the data points and the medoids. K-Medoids is more robust to noise and outliers compared to K-Means because it uses actual data points as cluster centers.

Input:

- K: Number of clusters
- D: Dataset containing n objects

Output:

- A set of k clusters

Given k, the K-Medoids algorithm is implemented in 5 steps:

1. Step 1: Arbitrarily choose k objects from D as the initial cluster centers (medoids).
2. Step 2: Find the dissimilarity (e.g., Euclidean distance) between each object in the dataset and the medoids.
3. Step 3: Assign each object to the cluster with the nearest medoid.
4. Step 4: Update the medoids by minimizing the sum of the dissimilarities between all objects in a cluster and the medoid. For each cluster, the object that minimizes this sum becomes the new medoid.
5. Step 5: Repeat the process until there is no change in the medoids.

Example:
Let the dataset D = {2, 4, 10, 12, 3, 20, 30, 11, 25}, and k = 2 clusters.

1. **Randomly assign initial medoids**: Assume m1 = 3 and m2 = 20.
    - Cluster 1 (k1) = {2, 3, 4},
    - Cluster 2 (k2) = {10, 12, 20, 30, 11, 25}.
2. **Calculate distances and reassign medoids**:
    - After calculating the dissimilarities, update medoids to m1 = 4 and m2 = 25.
    - Cluster 1 (k1) = {2, 3, 4, 10, 12, 11},
    - Cluster 2 (k2) = {20, 30, 25}.
3. **Update medoids and repeat**:
    - Continue updating medoids and clusters until the medoids stop changing.
4. **Final Medoids and Clusters**:
    - Cluster 1 (k1) = {2, 3, 4, 10, 12, 11},
    - Cluster 2 (k2) = {20, 30, 25}.

CODE:

```
from sklearn_extra.cluster import KMedoids

import numpy as np



# Example dataset

data = np.array([[2], [4], [10], [12], [3], [20], [30], [11], [25]])
```

```
# Apply K-Medoids with k=2

kmedoids = KMedoids(n_clusters=2, random_state=0)

kmedoids.fit(data)


# Print results

print("Final Medoids:", kmedoids.cluster_centers_)

print("Cluster Labels:", kmedoids.labels_)
```

OUTPUT:

CONCLUSION:

The K-Medoids algorithm is a robust clustering method, especially in the presence of noise and outliers, because it uses actual points from the dataset as the medoids. It effectively partitions data into k clusters based on minimizing the sum of dissimilarities between points and their assigned medoids.

What types of data preprocessing are necessary before applying the K-Medoids algorithm?

- Handling missing values (remove or impute them).
- Scaling/normalization (since distance-based methods are sensitive to magnitude).
- Removing duplicates/outliers if they distort clustering.
- Selecting appropriate features to improve cluster quality